Learning to detect named entities in bilingual code-mixed open speech corpora

by

Yihong Theis

M.S., Kansas State University, 2016

A THESIS

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

College of Engineering

Kansas State University

Manhattan, Kansas

2019

Approved by:

Major Professor

Dr. William Hsu

Copyright

Abstract

This research addresses the problem of code-mixing in speech-based cognitive services, and the subtasks of language identification in multilingual speech commands, search, and named entity recognition. According to the American Community Survey (ACS) published by the United States Census Bureau, more than 20 percent of U.S. residents speak a language other than English at home. Many bilingual speakers habitually and even subconsciously switch languages in mid-sentence and mix them in successive sentences. For example, this happens when a user wants to listen to popular music by artists from different countries and use the native pronunciation of the artist's name. Misrecognition of these embedded named entities by an automatic speech recognition (ASR) system can lead to wrong search results. For instance, when a user wants to play songs by Chinese singers on Spotify, home assistants frequently play the wrong songs because they only recognize English. When callers leave voicemail messages on Google Voice that are transcribed to text, specific named entities (people, places, and things) and the surrounding context of messages are often misinterpreted. Malfunctions of this kind are inconvenient and detract from the holistic user experience for home assistant users.

To develop a machine learning-driven approach towards coping with such usability issues, I developed a research test bed centered around code-mixed bilingual sentences. We collected voice recordings from 40 individual participants for multiple commands, multiple streaming music service names, and about 100 Chinese names. We segmented and recombined these samples automatically using sound editing software to combinatorically enumerate a set of utterances, each of which is a short command phrase. Instead of traditional ways to use Hidden Markov models (HMMS), I used a deep learning model which is part of the Baidu *DeepSpeech* Project and developed by contributors to the Mozilla *DeepSpeech* open source repository on GitHub. This narrows the focus of our code-mixing task, and the associated supervised learning task, to language identification and segmentation of utterances in different languages at the phrase level. This facilitates development of a prototype web application through which users can contribute their voice data to improve the system. In current and continuing work, I am improving the phrasal model using deep learning to develop a working prototype that integrates with cognitive service APIs (e.g., Amazon Alexa, Google Home) for Chinese/English music search.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my major professor, Dr. William Hsu, for his patience and guidance. Without his help, I could not have narrowed down my broad topic, and I want specifically to thank him for his support to go many conferences and present my work to 2019 Midwest Speech and Language Days. He did everything to support me, allowing me to chase my dream, which means a lot to me. Without his support, I could not reach the level that I have, and without his guidance, I might still not know how machine learning can help many people in society. I also want to thank him for his patience in helping me to improve my writing skills while finishing the report.

I also would like to thank the other members of my committee, Dr. Doina Caragea and Dr. Mitchell Neilsen for their guidance, and time they spent participating in my committee.

I also would thank to my friends and my peers who gave me suggestions and help me collecting the corpora for this thesis. Throughout my studies in computer science over the past three years, I have made lots of friends who have given me support and encouragement to help me pursue my continuing goals. Specifically, I would like to thank my friends Huichen Yang and Lei Luo. They gave me lots of good design suggestions to help me get through the process of completing my project. I am thankful to them for listening to me talk about the challenges I faced and giving me their opinions about my research topic.

Finally, I would like to thank my husband, my parents, and my parents-in-law. My parents and my husband have and continue to work hard to support my tuition and life expenses. Without their support, I could not finish my second master's program. They have allowed me to focus my studies, without worrying about daily stresses. I want to specifically express my love and thanks to my husband, who supports me through everything. Without his encouragement, I would never have finished my degree.

Dedication

This thesis is dedicated to my husband, Thomas Allen Theis; thank you for supporting and encourage me in finishing the program.

This thesis is also dedicated to my parents, Ying Yan and Jing Fan; thank you for helping me to distribute my survey and getting recordings.

Chapter 1 Introduction

This chapter will present the problem that I address in this thesis, the motivation for studying it, a formalization of the problem statement, and the objectives and significance of my approach.

1.1. Overview

According to the web site Languages.org, 40% of the present world population consists of individuals who speak only one language, while 43% of the world population can use two languages with equal fluency and more than 16% of the world population can use three languages or more. [1] As a new generation of users grows up, more multilingual speakers are emerging, generating increased interest in mixed language in speech communications. Mixed-language speech, which contains more than one language within an utterance, can also be called *code-mixing*. This thesis deals with the problem of automatic speech recognition with code mixing, which is often used by multilingual speakers and is a naturally emergent phenomenon among multilingual users of cognitive services which defined as using a set of machine learning algorithms to solve problems in the field of artificial intelligence and signal processing, encompassing machine self-learning and human-computer interaction, such as the skills of home assistants

Today there are many home assistant products on the market, such as the Google Home and Amazon Echo. The purpose of such home assistants include home automation is supplying users' convenience; for example, users can turn on their bedroom lights by giving spoken commands to home assistant devices placed throughout their homes, or you can play music by speaking the appropriate commands and the titles of specific works and albums, or names of artists. However, there are also open technical challenges in the design and implementation of cognitive services for home assistants; for example, if a multilingual speaker wants to listen to a song from a non-English-speaking country, home assistants have a much higher failure rate in practice. [2] This is because the commands that users give to assistive devices in such use cases is often code-mixed speech, and the cognitive services they employ do not support the ability to recognize such speech, specifically Chinese, for it has many different dialects. This AI phenomenon fails to

provide a proper method to incorporate bilingual users and defies the primary achievement of worldwide automation. Therefore, it is important to do research that aims to improve current speech recognition systems that can also be embedded in code-mixing speech.

1.2. Motivation

Code-mixed speech recognition faces many of challenges, which include the lack of training data for language modeling [3], co-articulation effects which denotes instances where two successive sounds were articulated together [4]from one language that may influence the other [5], and the need for expert linguistic knowledge. Moreover, language models need to combine with Hidden Markov Model (HMM) which is statistically Markov model with unobservable states, so they can be able to predict transition points between two languages [6]. One main challenge is that the size of the vocabulary for an automatic speech recognition (ASR) task may be up to twice that of a monolingual system, and the predominant language of the system (e.g., that used to give commands) might be very different from the client language (e.g., that used only to specify embedded named entities). One such language pair, which this thesis focuses on, is Chinese and English. In written Chinese, complex ideogrammic characters form the words of the lexicon, while English uses alphabetic characters to form words. This results in morphological differences such as in the formation of homonyms in the spoken language. There are also many differences in speech units, some of which result from the pronunciation of words and some of which are based on other syntactic differences between Chinese and English.

Named entities are often translated in ASR tasks in order to simplify the task domain from multilingual to monolingual. When this is done, however, some regional information may be lost; for example, accents and dialects within the native language might cause synonymous names to be spuriously distinguished. One example of this is 杨千嬅, the last character of which can be read *Yè* or *Huà*, so that some speakers might read the name as either *Yáng Qiānhuà* (the canonical dictionary reading) or *Yáng Qiānyè* (a more popular alternate reading). Since individuals' dialects and accents affect their way that they pronounce words, phrases, and specifically named entities, this entails a need for contextualized ASR. However, there is a general lack of large speech corpora for different language pairs, presenting another challenge for code-mixed ASR, where embeddings of named entities and other phrases and lexical units is

commonplace. This is one reason it is difficult to develop a good ASR system that can handle code-mixing in human speech in general and particularly in cognitive services.

These challenges are reflected in our main motivating use case: cognitive services for playing music. An example of the limitations of existing work on multilingual cognitive services is seen in services for the Amazon Alexa virtual assistant, which drives the Amazon Echo line of devices. Alexa is unable to properly recognize commands to play songs by Chinese singers, such as: "Play 杨千嬅 [*Yáng Qiānhuà*] on Spotify". Similar examples are also abundant among Google Voice transcripts of voicemails as noted by Hsu (2019). [7] When Chinese speakers call his Google Voice number and leave voicemail messages, the text transcripts produced using Google Voice's speech-to-text service admit a high error rate. For example, if a person says "Yihong will pick you up later", Google might mis-transcribe that statement as "U-Haul will pick you up later". Hsu reports that this is a very frequent occurrence with bilingual students in our lab and bilingual family members. In my initial informal survey of the state of the field, I found that this is a common problem that needs to be studied more. [8] [9]

This led me to formulate a project idea that centers around detection of Chinese names embedded in English speech. From a literature review on this topic, I found some research on the problem of detecting code-mixed speech and properly recognizing the words of its constituent languages. This task has been studied for English words in Chinese sentences [8] [9] [10]; however, there has not been much research on recognizing Chinese names in English sentences. [11] [12] This raises the following questions regarding how to approach the problem and formulate concrete tasks that can advance the state of code-mixed ASR in multilingual cognitive services:

1. **How will I construct our corpora? (On what home assistant skill, and underlying ASR task or tasks, will it be based?)**
2. **Should the corpus be automatically synthesized from isolated word recordings or collected by having human participants (training and validation speakers) read pre-enumerated phrases?**
3. **Can I use open source corpora?**
4. **How many Chinese names should I use in a sentence?**

5. **How can I prepare transcripts that human participants can read aloud to represent an appropriate range of task instances?**
6. **How can I prepare an interest survey to recruit participants, inviting them to record their voices?**
7. **To achieve the objective of training a sufficiently accurate model, how many participants suffice to attain a desired threshold Word Error Rate (WER)?**

The biggest challenge of this project is getting the appropriate data, as it is hard to find more than 100 hours of code-mixed conversation from any source, for any task. The Linguistic Data Consortium (LDC) has conversation recordings of nearly all-Chinese sentences interspersed with infrequent English words or phrases. However, I could not find any sources that consist of mostly English mixed with Chinese named entities. Therefore, I decided to build our own speech corpus by synthesizing custom phrases and collecting recordings of multiple speakers reading these phrases. These speakers are volunteer participants, recruited using a survey that asks for code-mixed bilingual Chinese and English speakers to help produce training data to improve an experimental bilingual cognitive service.

The traditional approach to code-mixing is to use Hidden Markov Models (HMM) to detect transitions from one language to another, then use an automatic speech recognition (ASR) pipeline to convert speech to text. The ASR pipeline usually breaks the problem into several key components, which extract features from an audio recording (e.g., a Microsoft wave file) and then use an acoustic model and a language model to decode the original audio into units of text, such as words, phrases, or sentences. The acoustic model usually builds a sequence of *phonemes* which are primitive units of differentiable sound in a language and can be mapped to the symbols of a phonetic alphabet. For example, “hello” can be represented as “HH AH L OW” in the International Phonetic Alphabet (IPA). Mandarin Chinese has at least four different phonemic tones, 21 initials (consonants) and 35 finals (vowels and vowel-nasal pairs); this adds complexity due to the need to translate many more phonemes in training and applying a language model. With this added complexity, a larger training data set is needed to insure the error rate is reduced to usable levels, perhaps Baidu *DeepSpeech* used thousands of hours of data to train [13].

Another approach applies end-to-end [1]deep learning [2]which is an idea of outputting complex data types from raw features, using recurrent neural networks, which takes advantage of the capacity of deep learning systems to learn from large datasets to improve overall performance [13]. Baidu has created a model that is called *DeepSpeech*, which is trained end-to-end to produce transcriptions from code-mixed. With enough data and computing power, it can also learn to filter noise and improve multilingual speaker-adaptive recognition [13]. This approach does not use traditional phonemes; instead, the model contains a merged multilingual character set (e.g., Roman alphabet and Hanzi) that are included in the training data. This approach should result in an increase in accuracy rate than other methods. We study this model to investigate whether I can modify it to develop code-mixed language models.

1.3. Problem Statement

As stated above, I usually approach the problem by creating a phoneme table for a multilingual language model; however, Chinese proper nouns (e.g., personal names) are not as redundant as most English ones. For instance, there are many common given names in English, such as "*Tommy*" or "*Mary*", *Tommy* is a diminutive form of *Tom*, which in turn is a short form of *Thomas*, and thus the formation of names is governed by etymological rules that do not have analogues in pictographic Asian languages. Because Chinese names are constructed from thousands of characters based on very little structure or rules, different people can have names consisting of combinations of (usually 2-4) individual characters each chosen from among thousands. For example, by contrast with the family name (e.g., "*Wang*"), a given name can be any character that exists in Chinese, or in some cases a pair of characters, which makes personal names more individualized and sometimes unique. This is a part of why current speech recognition systems fail to recognize Chinese names. When most collections of phonemes are created, many of the phonemes needed to translate Chinese names are left out because of the large number needed as well as their added complexity. An ASR corpus consists of both audio recording of speech and the ground truth text transcript (which is usually produced by hand

[1] End-to-end means that a machine learning model provides the full sequence of operations convert an input data into an output, subsuming or bypassing the intermediate stages that are differentiable and accessible in a traditional pipeline [101].
[2] Deep learning is part of a broader family of machine learning methods based on artificial neural networks, learning can be supervised, semi-supervised, or unsupervised [102] [103] [42].

annotation). In multilingual ASR, it may be important to recognize embedded words in their native language rather than transliterate them. Therefore, I need to seek a better approach to isolate Chinese characters from English words in the text transcript. Mozilla has created open speech corpora based on Baidu *DeepSpeech*. However, the model has not been trained using code-mixed corpora; all their lower word error rates are based on pretrained English models. Therefore, there currently exists no research on, or related source for, code-mixed training data for ASR systems such as *DeepSpeech* that are built as capable of multilingual transfer learning[3]. The training of recent models is based on English, French, and Chinese; even though *DeepSpeech* includes a transfer learning module, no one has researched code-mixed speech recognition training. We will attempt to use code-mixed speech **without** segmenting or translating words *in situ*. Part of my central hypothesis is that this naturalistic treatment of speech corpora will help improve the current process of model creation, in terms of representation and learning, especially for multilingual cognitive services. Traditional model creation uses a pronouncing dictionary of existing words, but personal names are among named entities that such dictionaries rarely include. To cope with this problem, I take a deep learning-based approach of aligning voice patterns with a text transcript of speech, with the aim of decreasing loss due to named entity detection failures.

1.4. Objectives and Significance of the Thesis

This section talks about the objective of the thesis and its significance, while introducing the overall goal, technical objectives, central hypothesis, and desiderata of this work.

1.4.1. Objectives

Code-mixed speech recognition is intrinsically more complex than typical monolingual speech recognition, or even multilingual use cases of ASR where language changes are at the sentence level or higher. It adds complexity by needing to detect when the speaker switches language. Code-mixed systems that represent, reason with, and learn any kind of grammar model also needs to be able to switch between grammars for each language and change context in recognizing phrase and sentence structure. In a traditional use case, researchers might ask around

[3] Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem [105].

15 people to have a conversation and freely mix languages during this conversation. During such a conversation, each person has their own way of speaking, and I cannot guarantee that these 15 people would represent other bilingual speakers in a learnable or generalizable way. Some bilingual speakers will use a loan word from another language instead of a longer word in their native language. Some speakers will use a foreign word or phrase in a sentence because they know that is the best one they know to express their meaning. To start to improve solutions to the problem of recognizing and understanding code-mixed speech, I want to take a closer look at when users naturally tend to use Chinese names in English sentences. These are mostly monolingual utterances that any bilingual speaker would use in their everyday life. For cognitive services, these utterances would tend to be commands or queries. In cognitive services, there are about 25,000 commands that a system such as Amazon Alexa can recognize [14]; meanwhile, the complexity of a conversational transcription system can be measured in the number of hours of training data for a task, which can number in the thousands [15]. Because current monolingual cognitive assistant services fail to detect the right Chinese name, the primary goal of this thesis is to try to improve detection of Chinese names in English sentences. Unlike other research that focuses on full code-mixed conversations, I developed our own corpus by asking volunteers to record themselves reading fixed content. Our goal is not to create a model to encompass a broad sampling of conversational Chinese and English code-mixed sentences, but to analyze the phonetic accents and transitions observed when reading aloud a representative set of names in English sentences used to operate a cognitive service for a particular skill – in this case, playing music on different services. I also intend to compare existing and new approaches to this problem, with an emphasis on improving measures of quality for code-mixed speech recognition, such as import a different, more appropriate language model in order to achieve a lower word error rate for the task.

1.4.2 Significance

In this thesis, I did not use traditional LDC data, but instead prepared imperative command sequences with variable named entities in English and Chinese, instead of asking people to hold a conversation. These sentences were of the form ***action entity*** **on** ***service***, where *action* ranged over terms such as “play”, “search”, “find”, and “who is”. In this way, I was able to focus a controlled experiment on raising the number of different speakers and automatically

generating combinations of these words (especially actions and entities) to form specific sentences. This test bed, consisting of sample text to be read by human speakers, phrase and unigram recordings, scripts for segmenting and recombining words, and the specifications used to produce experimental corpora, can help future researchers to do further work on named entity recognition in bilingual speech. We also conducted different experiments using a variety of evaluation criteria and methods to test what would be an effective way to detect name entities in bilingual speech. For example, one approach is to use machine learning to treat the Chinese singer name as an English name, which can get complicated because of the different pitch and tones used in Chinese.

Because the novel contribution of this research is in improving multilingual code-mixed recognition in an ASR-based cognitive service rather than advancing the state of ASR itself, I needed to find an open speech model to serve as a basis for developing a mixed speech recognition system and test bed. I explored CMU *PocketSphinx*, the Google Cloud Speech API and IBM *Watson*. These systems are very inflexible for multilingual uses: *PocketSphinx* only allows developers to edit the phonemes used, and not to modify the actual speech recognition system; moreover, Google Cloud Speech and Watson do not allow researchers to augment or modify phoneme sets at all. Therefore, I need to develop a new and extensible experimental framework to help us easily connect two or more language systems using a unified or hybrid model. For this purpose, I found that the Mozilla *DeepSpeech* platform is a suitably flexible one for developing my framework and implementing a prototype system within it. The significance of this thesis is to investigate how deep learning and a corpus-based approach can meet the representation, learning, and inference (decoding) needs of a code-mixed cognitive service task, and help future researchers advance research on code-mixed speech recognition in general.

Chapter 2 Background

This chapter talks about a paradigm shift in speech recognition from conventional ASR to the deep learning methodology that is currently prevalent. It consists of two sections: the first, on conventional ASR, including Hidden Markov Models, phonemes, and language models; and the second, on deep learning ASR, which briefly discusses deep neural networks, recurrent neural networks, and transfer learning.

2.1. Conventional ASR

Conventional ASR using Hidden Markov Models (HMM) was first explored in the 1970s [16], leading to the back-off family of models that allowed language models to use multiple length n-grams in tandem with HMM to recognize large-vocabulary, speaker-independent, continuous speech (the original Sphinx system) by Lee, Hon, and Reddy in 1987 [17, 18]. Then 1990s, the CMU Sphinx-II system was developed by Reddy's former student Huang [17]. This section will talk about how HMMs, language models, and phoneme recognition work in conventional ASR.

2.1.1. Hidden Markov Models

A Hidden Markov Model (HMM) is a stochastic process model, represented by a directed graph whose vertices represent states that are not directly observable [14], and whose edges represent transitions that are governed by the *Markov property* on state sequences ($n \geq 1, x_1, \dots, x_n$) [19]:

$$P\ (Y_n \in A\ |X_1 =\ x_1, \dots, X_n = x_n) = P\ (Y_n \in A|X_n = x_n). \tag{2.1}$$

HMM edges are decorated with output distributions, which allows them to represent *filtering*[4], *smoothing*[5], and *prediction*[6] tasks [20]. HMMs provide a simple and effective framework for modeling time-varying spectral vector sequences. They can be used in a large-vocabulary continuous speech recognition system [18]. Since speech has temporal structure and

[4] *Filtering* is the task of computing the given model's parameters, a sequence of observations, and the distribution over hidden states of the last latent variable at the end of the sequence [20].
[5] *Smoothing* is similar to filtering, but it is aimed at estimating the distribution of a latent variable somewhere in the middle of a sequence [20].
[6] *Prediction* is the task of predict a sequence of unknown (hidden) variables from a set of observed variables [20].

can be encoded as sequence of spectral vectors spanning the audio frequency range, HMMs can provide a natural framework for solving models for the forward (matching)[7], backward (decoding)[8], and forward-backward (learning) problems[9]. States in an HMM for ASR represent units of speech for the language to be recognized, such as *phonemes*[10] [21].

HMMs form the core of a basic framework for modern speech recognition because they provide a representation for the inference and learning tasks above and thereby allow a variety of Bayesian inference methods and specific algorithms to be applied [21]. For example, the forward-backward algorithm which is used to find the unknown parameters of an HMM model. Let X_t be a discrete hidden random variable with N possible values; in an HMM, it is assumed that $P(X_t|X_{t-1})$ is independent of time t, which can be a definition of the time-independent stochastic transition matrix [22]:

$$A = \{a_{ij}\} = P(X_t = j\ |X_{t-1} = i). \tag{2.2}$$

The initial state distribution as given in [22] is:

$$\pi_i = P(X_1 = i). \tag{2.3}$$

The observation variables Y_t can take one of K possible values, assuming that the "hidden state" is time-independent. The probability of a certain observation y_i at time t for state $X_t = j$ as given in [22] is:

$$b_j(y_j) = P(Y_t = y_i|X_{t-1} = i). \tag{2.4}$$

An observation sequence is given by $Y = (Y_1 = y_1, Y_2 = y_2, \dots, Y_T = y_T)$. The forward stochastic process $\alpha_i(t)$ is the probability of seeing the observations $y_1, y_2, \dots, y_t$ and being in

[7] The *forward algorithm* is used to calculate a 'belief state': the probability of a state at a certain time, given the history of evidence [106].
[8] The *backward algorithm* is used to calculate the probability that observation x_i in the sequence came from state k [106].
[9] The *forward-backward algorithm* is an inference algorithm for hidden Markov models which computes the posterior marginals of all hidden state variables given a sequence of observations [106].
[10] A *phoneme* is one of the units of sound that distinguish one word from another in a particular language.

state i at time t [22]. The backward stochastic process $\beta_i(t)$ is the probability of the ending partial sequence $y_{t+1}, \ldots, y_T$,continuing from state i at time t [22].

The architecture of an HMM-based ASR system is illustrated in Figure 2.1, which shows the input, an audio waveform which is in the time-amplitude domain where the independent axis is time, the dependent axis is amplitude, and oscillations occur within a fixed window of time. By contrast, audio data from a microphone has undergone a Fourier transform into frequency-amplitude domain, which consists of a histogram of frequencies observed in the waveform. All audio features can be extracted from different methods; for example, a short-term Fourier transform can be used to convert sound described in one domain to the other domain [23]. This process, called *feature extraction* , converts input into a sequence of fixed-size acoustic vectors $\boldsymbol{Y}_{\mathbf{1:T}} = y_1, \ldots, y_T$, where y_i are observation variables and t is the time as shown in Figure 2.2 HMM-based phone model. These feature vectors use a function called a *decoder* to try to find the sequence of words $w_{1:L} = w_1, .., w_L$ which is generated from Y,

$$\breve{\mathrm{w}} = \arg_{\mathrm{w}} \max\{\mathrm{P}(\mathrm{w}|\mathrm{Y})\}. \tag{2.5}$$

However, because the entire conditional $P(w|Y)$ is difficult[11] to model directly by HMMs, which are generative models[12], a Bayesian approach to inference and learning is taken that exploits conditional independence - specifically, the Markov property given above [21]. This provides a way to update beliefs based on the arrival of new, relevant pieces of evidence to refactor (2.5) into the following equation:

$$\breve{w} = arg_w \max\{p(Y|w)P(w)\}. \tag{2.6}$$

As shown in Figure 2.1, an *acoustic model* can determine (2.1) $p(Y|w)$, and the prior $P(w)$ can be determined by a *language model*, where the acoustic model is not normalized, the language model is often scaled by an empirically determined constant, and

[11] Some systems can be based on discriminative models [108] where $p(w|Y)$ is modelled directly, rather than generative models, such as HMMs, where the observation sequence $p(Y|w)$ is modelled. Generative algorithms make some kind of structural assumptions on the model which might influence the independence of the features.
[12] A generative model is a powerful way of learning any kind of data distribution suing unsupervised learning and it a static model of the joint probability distribution [107]

a word insertion penalty is added. In the decoder structure, the pronunciation dictionary consists of each distinct unit, or *phone*[13], that can be represented by the acoustic model. For instance, the word "*bat*" is composed of three phones */b/ /ae/ /t/* [21]. There are about 40 such phones in English. In summary, given any word w, the acoustic model concatenate phone models to form words as defined by the pronunciation dictionary. The phone models include training data consisting of speech waveforms and their corresponding text transcripts. Language models are typically N-gram models which include the probability of each word conditioned only on its $N-1$ predecessors [21]. The N-gram parameters are estimated by counting N-tuples from each of waveform transcripts [21]. In the end, when the decoder searches through all possible word sequences using pruning to remove unlikely hypothesizes thereby keeping the search tractable" and the end of the utterance is reached, the most likely corresponding text (recognized speech) is generated (Figure 2.1). [21]

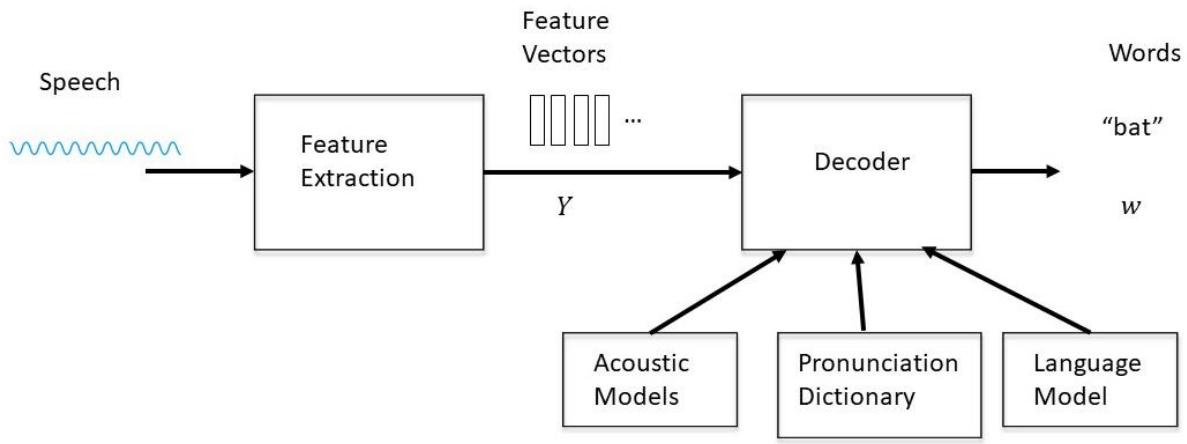

Figure 2.1 architecture of an HMM-based recognizer [21]

The objective of feature extraction is to provide a compact representation of the speech waveform, preserving information that helps discriminate between words such that acoustic units of speech map well to recognizable features. For example, many ASR systems use a variant of clustering called vector quantization to map packets of speech waveform data to their closest

[13] A *phone* is a speech sound considered as a physical event without regard to its place in the sound system of a langauge [109].

associated phoneme [21]. Feature vectors can be encoded by mel-frequency cepstral [14]coefficients (MFCC), one of the simplest and most widely-used encoding schemes [24]. They were introduced by Davis and Mermelstein in the 1980s and remain the state of the art to the present day. Before MFCCs were introduced, Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs) were the main feature type for automatic speech recognition (ASR), especially with HMM classifiers [24]. MFCCs use short windows for the digital signal, then calculate the periodogram estimate, an estimation of the spectral density of a signal. The mel scale is used for non-linear frequencies to provide smooth spectral estimates; it applies a truncated discrete cosine transformation (DCT), which expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies [21]. The formula for converting from frequency to mel scale is:

$$M(f) = 1125 \ln\left(1 + \frac{f}{700}\right). \tag{2.7}$$

An HMM consists of states $q_1, q_2, .., q_n$ with associated transition probabilities $\{a_{ij}\}$,forming the linguistic component of the model, and output emission probabilities $\{b_{ij}(k)\}$ with k ranging over elements from the codebook (VQ output) [15]defined above. The linguistic model is based on a sequence of basic states (phones) $q(t)$, each occurrence of which comes from the decomposition of a sequence of spoken words. Figure 2.2 illustrates how each phone q corresponds to a state in a continuous density HMM model, the probability parameters $\{a_{ij}\}$ correspond to successive pairs related by a transition, the output observation distribution $\{b_{ij}(k)\}$ corresponds to the set of possible ways in which a transition can generate or be interpreted as, a sound $y(t)$. The underlying assumption that is enforced as a constraint throughout the HMM is that the states are conditionally independent of all other states given the previous state, and each

[14] Mel-frequency cepstra are a representation of the shot-term power spectrum of a sound, based on a linear cosine transform of a log power on a nonlinear mel scale of frequency.

[15] Vector quantization (VQ) can be used in signal compression, speech and image coding, and it is based on standard clustering algorithms. The individual cluster centroids are called codewords, and set of cluster centroids is called a codework. Among the most common and basic VQ algorithms is K-means clustering [110].

observation is conditionally independent of all other observations given the state that generated it [21].

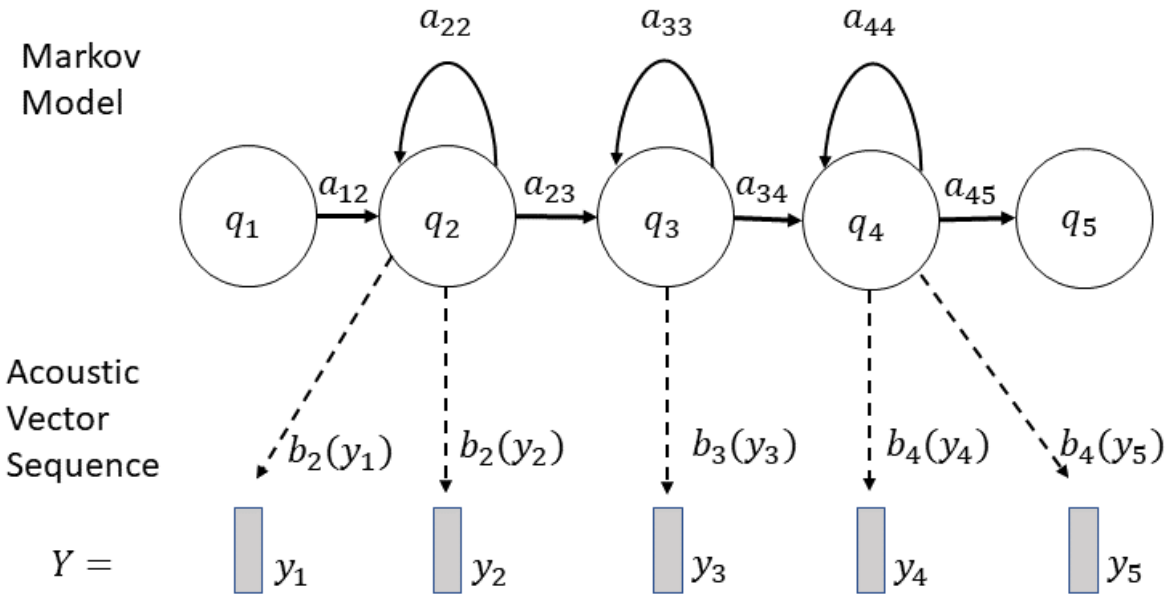

Figure 2.2 HMM-based phone model [21]

2.1.2. Phonemes

As discussed above, many modern speech recognition systems are based on HMMs, which must be trained using labeled corpora as with other supervised learning models., This requires a high volume of data to train each parameter to increase the probability of recognizing the right phoneme when the trained model is applied to new speech input. These parameters $\{a_{ij}\}$and$\{b_{ij}(k)\}$ correspond to inter-phoneme transition and per-transition output emission probabilities. Because different speakers have different fundamental speaking frequencies and noise within each audio recording may influence the recognition of phonemes, the learning phase of ASR system development may require on the order of hundreds of hours of recorded speech (on the order of one hour of speech per training speaker) [25]. For instance, the phoneme [s], which occurs in the word "*sing*" or "*song*" (Figure 2.3), also occurs in many other words. For example, there are about 50 phones in English, and the possible count of triphone could be in the order of 50^3, which is a very large high number and causing recognition failure [26].

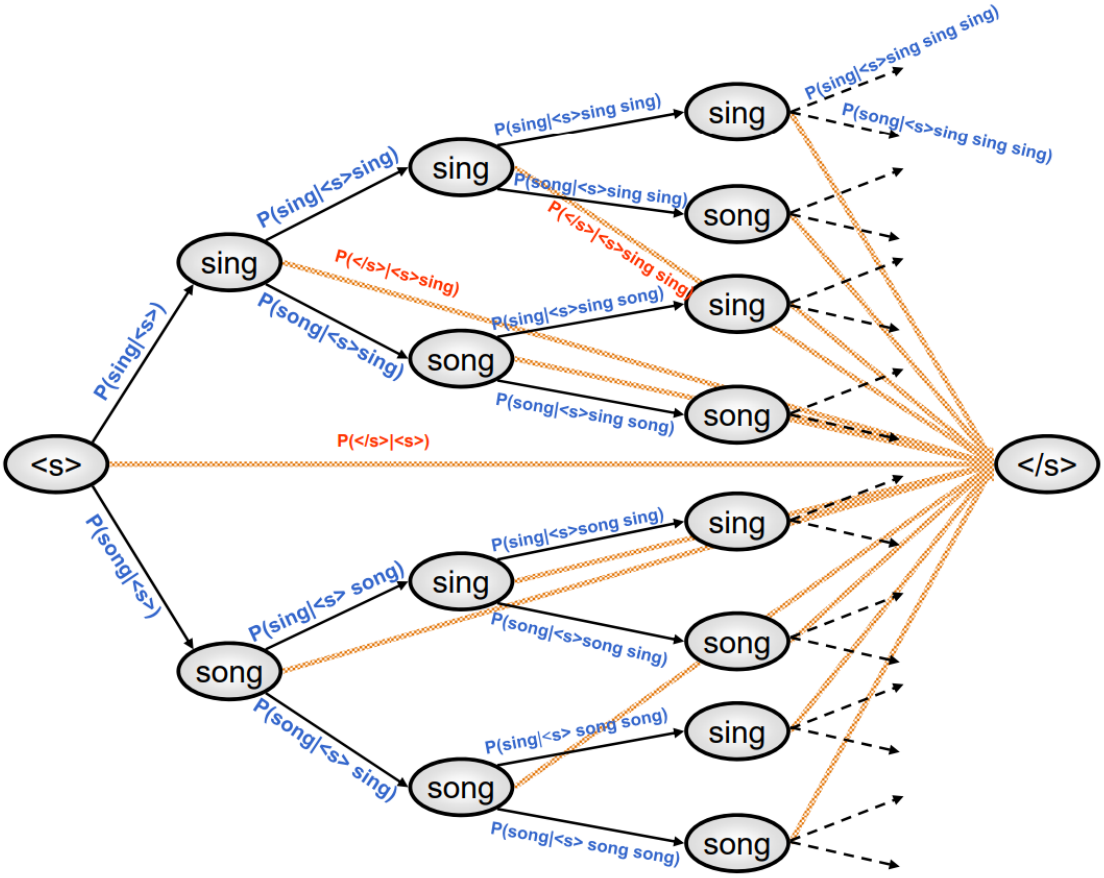

Figure 2.3: a two-word vocabulary: "sing" and "song" [25]

In this case, a combination of Gaussian Mixture Models [16](GMM) can be used to find the closest phoneme after clustering similar sounds based on observed inter-phoneme transitions and emitted outputs. As Figure 2.4 shows, "the GMM can be viewed as a hybrid model between parametric[17] and non-parametric[18]" [27]. The GMM density can be defined as the sum of weights of Gaussian densities given the Gaussian component of the graphical model and corresponding component probabilities [28]. The GMM-HMM has the ability to find the joint maximum probability among all possible reference words W given the observation sequence O

[16] A *Gaussian Mixture Model (GMM)* is a parametric probability density function represented as a weighted sum of Gaussian component densities.
[17] A parametric model captures all information of the data within the parameters, and it is predicting a future data value from the current state of the model by its parameters, such as knowing two parameters will enable to predict a new value [25].
[18] A non-parametric model captures more subtle aspects of the data, and it allows more information to pass the current state to predict the further data [25].

[28]. These likelihood values are passed to the decision block, where they can be transformed into the new combined likelihood $L'(W)$:

$$L'(w) = \big(1 - x(W)\big)LiGMM + x(W)LiHMM \tag{2.8}$$

where LiGMM is the production of i^{th} word-independent GMM (for $i \in \{1, 2, \dots, W\}$ where W is the size of the lexicon), and LiHMM is the production of ith word-independent HMM. The $x(W)$ denotes a weighted coefficient [28]. "GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as vocal-tract related spectral features in a speaker recognition system" [29]. Figure 2.5 shows the variations in the vowel space for a set of distinct speakers (in this case, three).

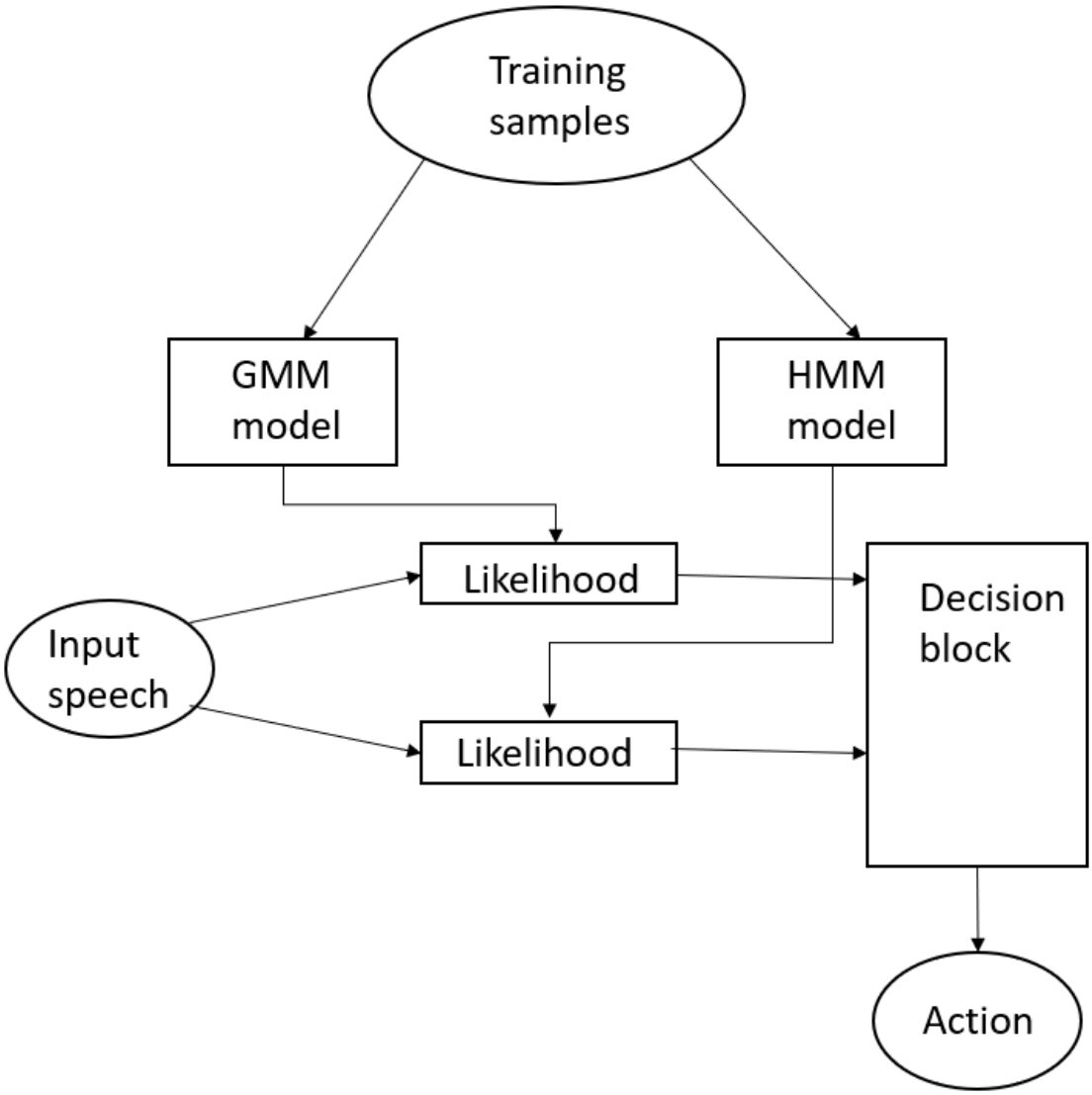

Figure 2.4:GMM-HMM model [28]

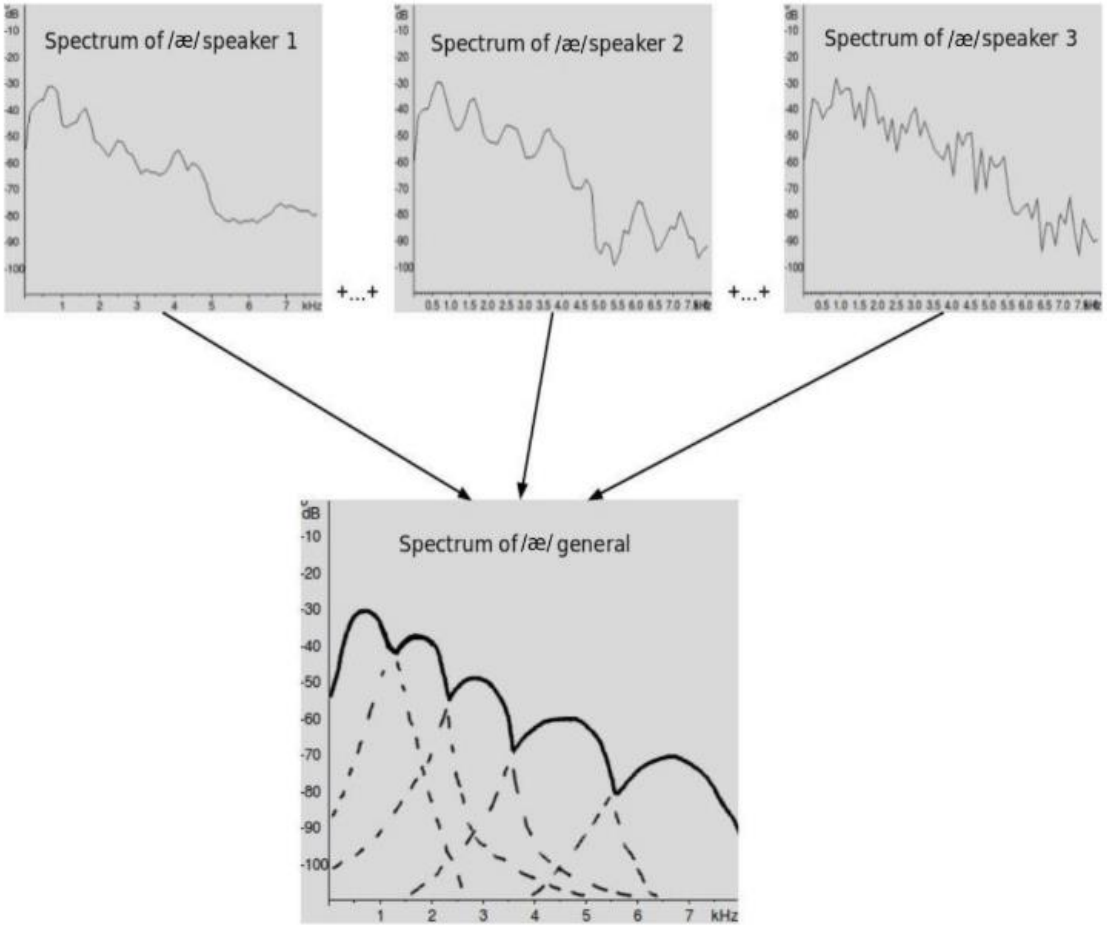

Figure 2.5: Modeling phoneme [ae] using multiple occurrences of phoneme [ae]. [25]

The general definition of a *phoneme* is "any of the perceptually distinct units of sounds in specified language that distinguish one word from another", Examples include *p, b, d* and *t* in the English words *pad*, *pat*, *bad*, and *bat*. [30] There are 26 letters corresponding to approximately 44 unique sounds, which are categorized as consonants (Table 2.1:) and vowels (Table 2.2) in English. [31]

Phoneme	**IPA Symbol**	**Graphemes**	**Examples**
1	b	b, bb	bug, bubble
2	d	d, dd, ed	dad, add, milled
3	f	f, ff, ph, gh, if, ft	fat, cliff, phone, enough, half, often

Table 2.1: Example of English consonants [31]

Phoneme	IPA Symbol	Graphemes	Examples
25	æ	a, ai, au	cat, plaid, laugh
26	eɪ	a, ai, eigh, aigh, ay, er, et, ei, au, a_e, ea, ey	bay, maid, weigh, straight, pay, foyer, filet, eight, gauge, mate, break, they
27	e	e, ea, u, ie, ai, a, eo, ei, ae	end, bread, bury, friend, said, many, leopard, heifer, aesthetic

Table 2.2: Example of English vowels [31]

Mandarin Chinese, however, has 7 vowel phonemes in, short (2-3 phone) sequences of which form generally monosyllabic words whose structure in Mandarin consists of an optional initial phone, followed by a vowel (accompanied by tone) and, for some words, an optional final consonant (n or ng) (Table 2.3). Mandarin consists of 22 consonant phonemes (Table 2.4) and 4 different tones that determine pronunciation. The government of the People's Republic of China adopted the romanization system called ***Pīnyīn*** which represents tones by diacritical marks over vowels (Table 2.5) [32].

	Front		Central	Back
	Unrounded	**Rounded**	**Unrounded**	**Rounded**
Close	i	y		u
Mild	e		ə	o
Open			a	

Table 2.3: Mandarin Vowels [32]

		Bilabial	Labiodental	Alveolar	Retroflex	Alveolopalatal	Velar
Stops	unaspirated	p		t			k
	aspirated	p^h		t^h			k^h
Fricatives			f	s	ʂ	ɕ	
Affricates	unaspirated			ts	tʂ	tɕ	
	aspirated			ts^h	$tʂ^h$	$tɕ^h$	

Nasals		m		n		ŋ	
Lateral				l			
Approximants		w			ɻ		

Table 2.4: Mandarin 22 consonant phonemes [32]

1st tone	high-level	mā	'mother'
2nd tone	rising	má	'hemp'
3rd tone	Falling-rising	mǎ	'horse'
4th tone	falling	mà	'scold'

Table 2.5: Mandarin different tones [32]

Modern research studies mostly use the pronouncing dictionary published by Carnegie Mellon University (CMU) as part of their language model [33]; it consists of over 134,000 words and their pronunciations [34]. The current set has 39 phonemes and is based on the *ARPAbet* symbol set [19]for speech recognition. Table 2.6 shows how the CMU pronunciation dictionary is represented. Because Chinese and English pronunciation are distinct, some systems define a subtask of tagging contiguous word sequences by language in code-mixed sentences. These sequences are then treated using different pronunciation dictionaries; for example, THCH30 is an open source Chinese speech database published by the Center for Speech and Language Technology (CSLT) at Tsinghua University [35]. Table 2.7 shows three examples of how Chinese phonemes are represented in THCH30 dictionary.

Phoneme	**Example**	**Translation**
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T

Table 2.6: Example of CMU phoneme dictionary

Phrases	**Translation**
爱祖国 (love country)	aa ai4 z u3 g uo2
奥地利 (Austria)	aa ao4 d i4 l i4

[19] ARPAbet symbol set is a set of phonetic transcription codes developed by Advanced Research Projects Agency (ARPA) as a part of their Speech Understanding Research project in the 1970s

八月 (August)	b a1 vv ve4

Table 2.7: Chinese pronunciation dictionary [35]

2.1.3. N-Gram Language models

Language models (LMs) are functions that allow one sequence of language units to be recognized from another parallel sequence. For example, translation models map words and phrases in a source language to those of a target language. In ASR the goal is to map sounds to units of speech, from phones to syllables, to words (unigrams), to phrases, and finally to full sentences. Probabilistic LMs are among the most important types; they use conditional probability functions to represent possible sequence-to-sequence mappings, for example $P(next\ word \mid previous\ sequence)$. An N-gram is a sequence of N words, 2-gram is a two-word sequence of words like "please turn", "turn your", or "your homework", and a 3-gram is a three-word sequence of words like "please turn your", or "turn your homework" [36]. LMs are important to speech recognition because that they represent the order of words in a sentence in order to support inference (reasoning) tasks predicting the next word in a sequence of arbitrary length [37]. Furthermore, a fundamental task in developing an LM is learning: choosing or iteratively refining the architecture of the LM and training using an appropriate, prepared corpus.

The main goal of LMs in speech recognition is to recognize spontaneous speech by creating general models for some specified domain of recognition, comprising the group of users, languages and vocabulary of words recognized, and context of speech that defines the recognition task. This is divided into a learning process that results in a trained LM, and a continuing use case where this trained LM is used to perform the inference task of ASR. For instance, the backward LM is that finding the probability of each word by estimating on the following words, and the backward LM can be estimated on the same training texts as the forward LM, but with the words in each sentence in reverse order [38].

In order to prepare training data for the most common kinds of LMs, short n-grams (2-grams or bigrams, 3-grams or trigrams, and up to 4 or 5--grams) are generally used. The n-gram

language model represents every possible sequence[20] of units that occurs in a sequence. Based on Equation (2.6) above, the n -gram assumption would be,

$$P(w_K|w_1, w_2, w_3, \dots w_{K-1}) = P(w_K|w_{K-(N-1)}, w_{2K-(N-2)}, \dots w_{K-1}) \tag{2.9}$$

To explain the details of how n-grams helped LMs define unit probabilities, take an example sentence such as "*the little girl is so beautiful*", for which I need to compute the probability of a word *w* given some history *h* (a preceding sequence). n-gram frequencies can be used to estimate for probability that the word "*that*" will occur next in the sentence:

$$P\ (\text{that}|\ \text{the little girl is so beautiful}). \tag{2.10}$$

Because we wish to find the most probable next word (i.e., compute the maximum likelihood prediction), the system needs a large enough corpus to estimate the frequency (count) of word sequences ending in *that* and get the estimate the probability as a frequency, and hence likelihood, ratio:

$$P(\text{that} \mid \text{the little girl is so beautiful}) = \frac{\text{C(the little girl is so beautiful that)}}{\text{C(the little girl is so beautiful)}} \tag{2.11}$$

In order to predict further words in a sequence, I use a *sliding window*: the current n-gram context, which is advanced as next-word candidates are enumerated, in order to estimate desired probabilities using a frequency model compiled from our existing corpus. This model estimates the probability of a next-word candidate by treating that word as a random variable X_i,, e.g.,$P\ (Xi\ =\ "that")$, abbreviated P(that) *[36]*. The sequence of N words is denoted $w_1 \dots w_n$ or $\boldsymbol{w}_1^n$ (where the expression w_1^{n-1} means the string $w_1, w_2, \dots, w_{n-1}$) *[36]*. $P\ (w_1, w_2, \dots, w_n)$ represents the joint probability of each word in a sequence having a value $P\ (X\ =\ w_1, Y\ =\ w_2, Z\ =\ w_3, \dots, W\ =\ w_n)$:

$$\begin{aligned} P(X_1 \dots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_1^2)..P(X_n|X_1^{n-1}) \\ &= \prod_{k=1}^{n} P(X_k|X_1^{k-1}) \end{aligned} \tag{2.12}$$

Bayes' rule of conditioning allows us to compute estimates of the conditional probability of a word given previous words from frequency estimates of the joint probability of sequences. By

[20] The *possible sequences* includes zero-frequency *N*-grams and what the default or contextually inferred probabilities are, while the *observed sequences* are those which generate the frequencies.

using it with sliding windows, I can estimate the joint probability of an entire sequence of words as a product of conditional probabilities as shown in Equation (2.12), an application of the chain rule[21] on word sequences using a Markov LM [36].

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)..P(w_n|w_1^{n-1}) = \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \quad (2.13)$$

Markov language models are defined as a formalism for reasoning about states over time [39]. They employ two assumptions in order to tractably reason about time series: one is the *limited horizon* assumption which is that the state at time t depends only on the state at time $t-1$, and the other one is the *stationary process* assumption which states that the conditional distribution over the next state given the current state dues not change over time [39]. For example, for bigram LMs estimate the probability of each word given all possible previous words $P(w_1^n|w_1^{n-1})$ by using only the conditional probability of the preceding word $P(w_1|w_{n-1})$. In this case, bigrams are used to generate n-gram by looking n-1 words into the past. Thus, the bigram approximation of the conditional probability of the nest word in a sequence is:

$$P(w_1^n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1}) \quad (2.14)$$

2.2. ASR in Deep Learning

Deep learning is a family of machine learning methods based on artificial neural networks, which can be supervised, semi-supervised or unsupervised [40] [41] [42]. Deep learning architectures such as deep neural networks (including the major subcategories of recurrent neural networks and convolutional neural networks) and deep belief networks, have been applied in many fields such as speech recognition and natural language processing [43].

[21] The *chain rule* permits the calculation of any member of the joint distribution of a set of random variables using only conditional probabilities.
Random two events for the chain rule:

$$P(A \cap B) = P(A|B) \cdot P(B)$$

More than two events:

$$P(A_n \cap ... \ A_1) = P(A_n|A_{n-1} \cap ... \cap A_1) \cdot P(A_{n-1} \cap ... \cap A_1)$$

which by induction turned into: $P(A_n \cap ... \ A_1) = \prod_{k=1}^{n} P(A_k| \bigcap_{j-1}^{k-1} A_j)$ [117]

2.2.1 Deep Neural Network (DNN)

As mentioned above, HMMs are used in some modern speech recognition systems to deal with speech variability and GMMs, as defined above in Section 2.1.2, in turn are used at each state of an HMM to fit a frame[22] or short window of frames for the acoustic input. With the rise of deep learning in recent years, deep neural networks were introduced into speech recognition that have many hidden layers (often more than two) and are trained using new methods that outperform HMMs with GMMs on a variety of speech recognition tasks. Hinton et al. (2012) compared the performance of HMM-based models and DNN-based models and found that monophone DNN-based models achieve a better phone accuracy rate than HMM-based models [44].

As Yu et. al proposed in their paper, "A deep neural network (DNN) is a feed-forward, artificial neural network that has more than one layer of hidden units between its inputs and its output" [44]. This means DNNs allow data to flow from the input layer to the output layer without looping back. Figure 2.6 from the book by Yu et al. on ASR deep learning approaches, depicts a DNN with a total of five layers, which include input layer and output layer as well as three hidden layers.

[22] A *frame* is an acoustic feature vector estimated on a 20-30ms signal window [115]

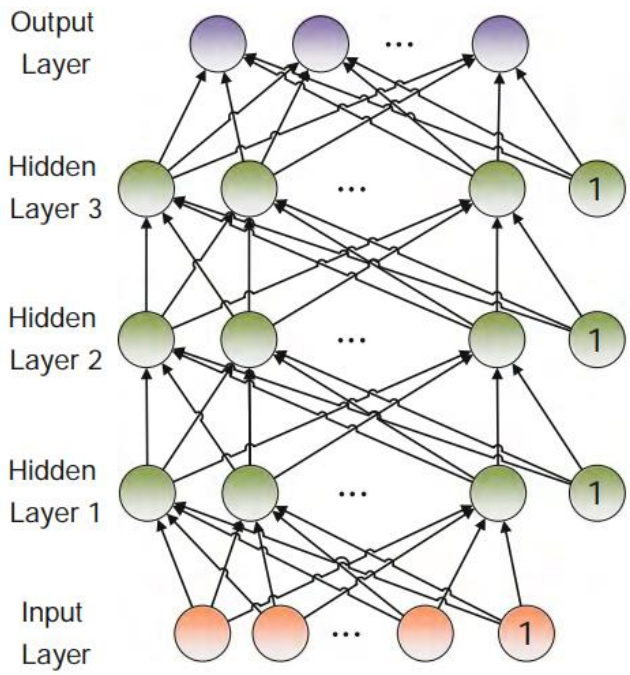

Figure 2.6: An example deep neural network with an input layer, three hidden layers, and an output layer [45]

The three hidden layers consist of non-output units which have trainable weights associated with connections that synapse upon them; these originate with the input units or with preceding hidden layers. The input layer, whose input consists of the incoming data vectors, is needs treated as layer 0, making the output layer the L[th] for an L+1-layer network [45]. The first L layers can be represented as follows:

$$\mathbf{v}^{l} = f(z^{l}) = f(W^{l}\mathbf{v}^{l-1} + b^{l}), \text{for } 0 < l < L, \tag{2.15}$$

where $z^l = W^l \boldsymbol{v}^{l-1} + b^l \in \mathbb{R}^{N_l \times 1}$, $v^l \in \mathbb{R}^{N_l \times 1}$, $W^l \in \mathbb{R}^{N_l \times N_{l-1}}$, $b^l \in \mathbb{R}^{N_l \times 1}$, and $N^l \in \mathbb{R}$, which represents: the excitation vector z^l; the activation vector v^l whose components are the activations v_j^l; the weight matrix W; the bias vector b^l whose elements b_j^l express the bias of the j^{th} neuron in the l^{th} layer; and l, the number of neurons at layer L [45]. The purpose of a neural network is to map an input vector into an output vector, and the non-linear units' "neurons", which are organized into one more layer. The interested reader is referred to Upadhyay [46] for background on DNNs feedforward networks used to approximate a target function, in which the flow of information is forward from inputs to network-approximated outputs, as x is used to calculate some intermediate function in the hidden layer which in turn is used to calculate y.

Linear combinations of weighted inputs pass through an activation function within each hidden unit (hidden layer neuron), thus determining the overall activation that is transmitted to all neurons to which that hidden unit is connected. The weight matrix represents the DNN's map of artificial[23] neurons and assigns random numerical values to the parameters associated with each interconnection, initializing the weight matrix. Furthermore, DNNs use the error backpropagation (EBP or backprop) algorithm to compute the gradient of the cost function [47]. EBP is a gradient descent algorithm used to update the neural network's weights and biases given a loss function computed using the current network's approximating output and the target output values in supervised training data. The primary beneficial feature of EBP is its iterative, recursive, and efficient method for calculating weight updates to improve the network until it is able to perform the task for which it is being trained [48].

The training criterion of a DNN is based upon multiple design parameters, such as the network architecture (expressed by the number of layers, number of units per layer, interconnection density, etc.), learning rate, and distribution of initial weights (which may be specified by hyperparameters of a prior) [49]. Typically, DNNs are more efficient at speeding up computation using small batches [24]of input data for a large training set [49]. The advantage of using small batches as opposed to the full dataset goes back the fundamental idea of stochastic gradient descent. The full dataset will take lots of memory to do when computed the gradient over the entire dataset. In this case, DNN can have the best performance on mini-batch that can reduce the noises when updating the parameters and running efficient [48].

DNN can also interface with deep belief networks (DBN). DBNs are a class of deep neural network, which is composed of multiple layers of hidden units with full interconnections between successive layers but between units within layers. DBNs help DNNs scale well because they can be trained greedily, one layer at a time, speeding up the training process. However, it is difficult to use the parallelism of cluster systems effectively when training on DBN-DNN. Furthermore, DNNs use the EBP algorithm and are hence susceptible to which might cause features with low weights to be spuriously eliminated in hidden layers given small labeled training data sets (a form of underfitting) [44]; DBNs can help mitigate this problem by

[23] All DNNs are **artificial** neural networks (ANNs), as opposed to NNs, which by default are biological. ANNs simulate some aspects of the behavior of NNs.
[24] Batch means a group of training samples.

reformulating these features from layer to layer using autoencoders. An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner [50], which can also denoise the input and reduce the dimensionality of output logits [51]. Figure 2.7, a reproduction of Hinton et al.'s deep neural network paper, shows the stepwise process of training DBN-DNN models; first, a stack of models is trained which has one hidden layer; second, they are composed into a single DBN; third and finally, outputs are added and a full DNN is trained using backpropagation.

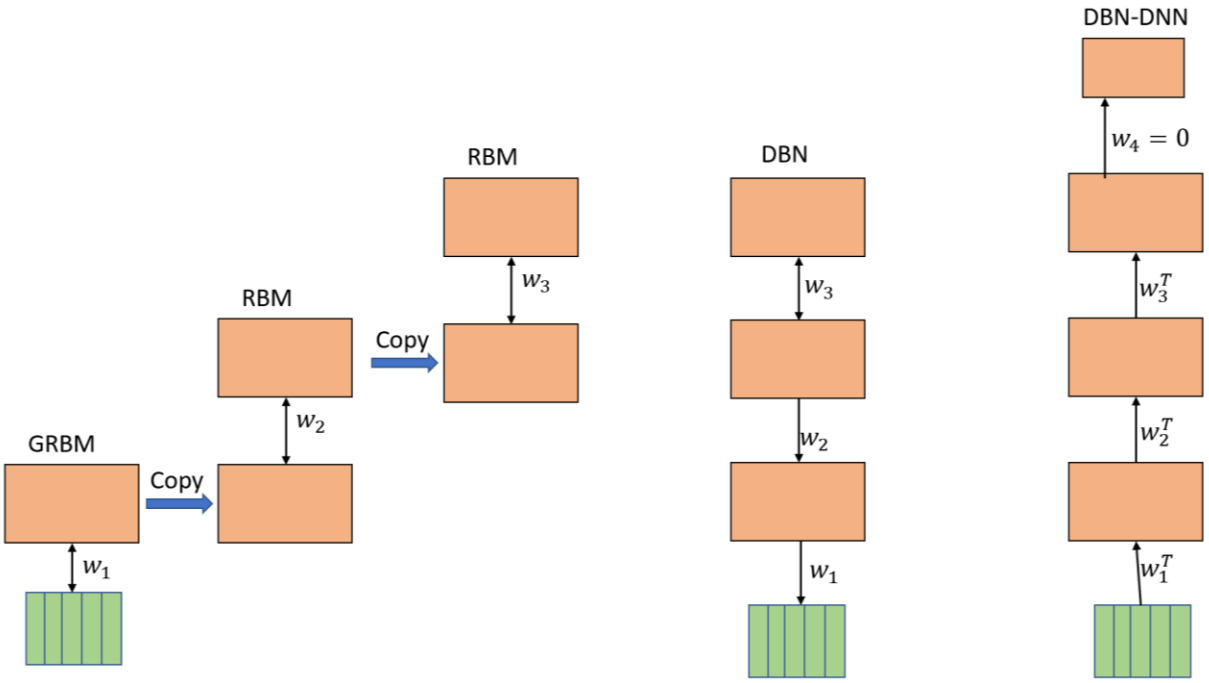

Figure 2.7: How DBN connects DNN [44]

The general benefit of using a DNN is that it learns intermediate representations (by training autoencoders) that are useful to prediction of the overall supervised target (an HMM state sequence) from a sufficiently large corpus of sequences (training speech). The implicit tradeoff of this approach is that the intermediate representation does not necessarily align naturally with the target sequence. A deep belief network-based deep neural network (DBN-DNN) might learn a discriminant[25] that predicts HMM states corresponding to the central frame

[25] Discriminant function is a statistical procedure that classifies unknown individuals and the probability of their classification into a certain group [116]

of each input window in a forced alignment. A less constrained representation is needed that supplies a mapping to the HMM states.

Toward this objective, it can interface a generic DNN with HMMs, replacing only the GMM to form a GMM-HMM hybrid with longer mel-frequency cepstral coefficient (MFCC) windows, omitting the DBN transformation [44]. The difference between a DNN-HMM and a GMM-HMM is the use of DNN instead of a GMM to estimate observation probabilities. Each output neuron of a DNN is trained to estimate the posterior probability distribution over the states of a continuous-density HMM, given acoustic observations. Using this approach, our HMM deals with the dynamic characteristics of speech signal and the DNN is used to model the observation probability [52]. The tradeoff of GMM-HMM is that it only fits a frame or a short window and lacks flexibility; however, DNNs can use many hidden layers to solve the HMM monophone training problem, and DBN-DNNs can use a larger vocabulary than HMMs. When trained on a data-parallel architecture such as a GPU, the clustering component of system does not perform efficiently or scale well. DNN-HMMs have been used for code-mixed ASR: they use their HMM component to detect language boundaries and then use their DNN component and phoneme dictionary to align sequences of utterances with transcripts [53]; however, this makes the DNN a black box, which only can detect in the phoneme dictionary word, not words that occur outside the dictionary.

2.2.2 Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) RNN

Recurrent neural networks (RNN) are another type of artificial neural network (ANN) which represents parameters as interconnection weights between nodes from a directed graph whose input consists of vectors in a temporal sequence. This allows it to take time and sequence into account. Unlike DNN, which are feed-forward neural networks that do not have a notion of order in time, recurrent neural networks (RNNs) remember the past and its inferences, such as classification and prediction outputs, are computed based on what it has learned from the past. In other words, RNNs can remember temporal patterns learned from prior sequential input which is fed through them to calculate outputs and a loss function for supervised training. Furthermore, it also can retain contextual representation in the form of hidden state vectors. Figure 2.8 is a diagram that shows how RNNs with hidden layers work, the outputs are $o^{(t)}$, the hidden layer

activations are $h^{(t)}$, the targets are $y^{(t)}$, and the loss is $L^{(t)}$; under each time step t, it aligns parallel sequences with three inputs x and three target y at each hidden layer; in addition, it stored the trained output to o and o is the only information that is allowed to send to the future [48].

In detail, let t be the time index, x_t the K × 1 vector of inputs for that time unit, h_t the $N \times 1$ vector of hidden state values, and $y_t$be the parallel L × 1 vector of outputs inputs aligned with x_t. Forward propagation begins with a specification of the initial state $h^{(0)}$, then for each time step from $t = 1$ and to $t = \tau$, then it can apply the equations as follows:

$$a^{(t)} = \boldsymbol{b} + \boldsymbol{W}h^{(t-1)} + \boldsymbol{U}x^{t} \tag{2.16}$$

$$h^{(t)} = \tanh(a^{(t)}) \tag{2.17}$$

$$o^{(t)} = \boldsymbol{c} + \boldsymbol{V}h^{(t)} \tag{2.18}$$

$$\mathrm{h_t} = \mathrm{f}\,(\mathbf{W}_{\mathrm{xh}}\mathrm{x_t} + \mathbf{W}_{\mathrm{hh}}\mathrm{h_{t-1}}) \tag{2.19}$$

$$\mathrm{y_t} = \mathrm{g}(\mathbf{W}_{\mathrm{hy}}\mathrm{h_t}) \tag{2.20}$$

where $\boldsymbol{b}$ and $\boldsymbol{c}$ are the bias vectors that along with the weight matrices $\boldsymbol{U}$,$\boldsymbol{V}$ and $\boldsymbol{W}$; in other words, for input-to-hidden, hidden-to-output, and hidden-to-hidden connections [48]. W_{hy} represents the matrix $L \times N$ of weights connecting the hidden units to the corresponding L outputs at every time step [44]. Because next outputs can aggregate from previous the sate vector, the state equation becomes

$$\mathrm{h_t} = \mathrm{f}\,(\mathrm{W_{xh}x_t} + \mathrm{W_{hh}h_{t-1}} + \mathrm{W_{yh}y_{t-1}}) \tag{2.21}$$

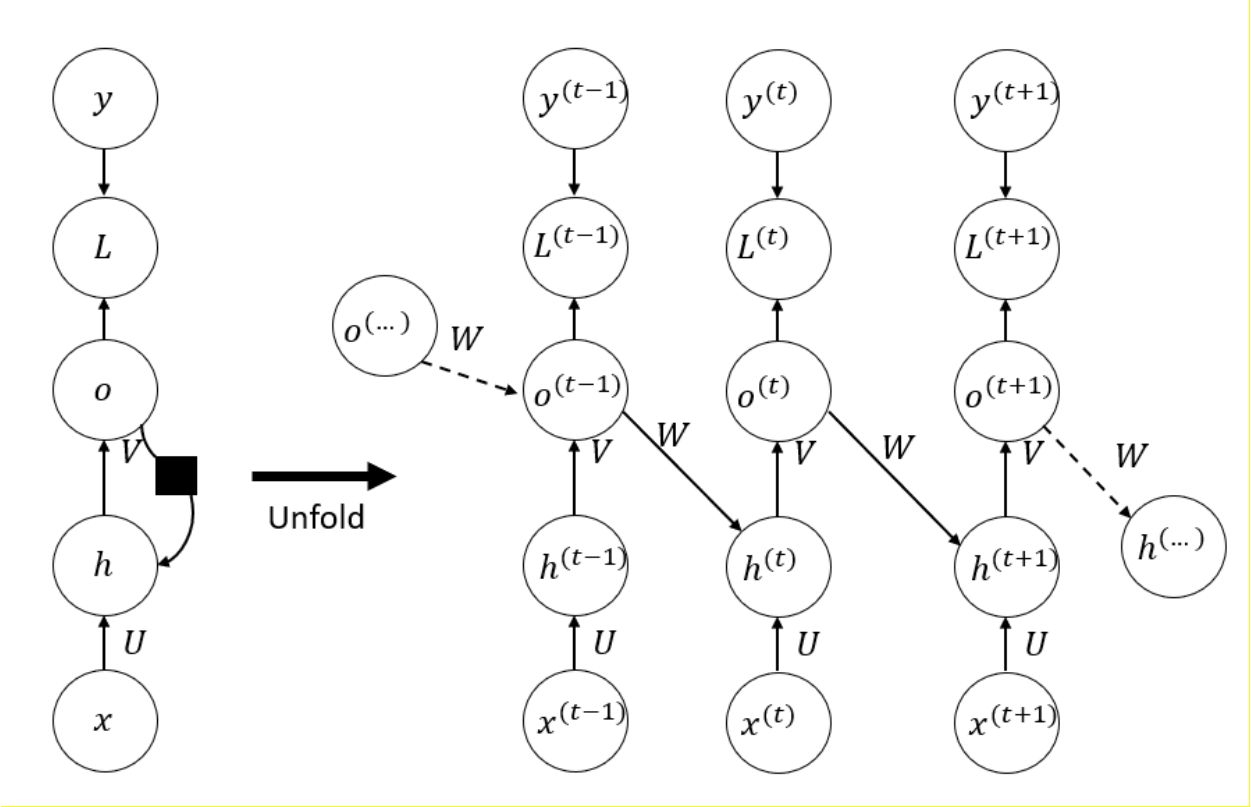

Figure 2.8: A Recurrent Neural Network, with a hidden state that is meant to carry pertinent information from one input item in the series to others. [48]

Even though RNNs go beyond DNNs to provide a solution to the time representation problem, one problem of RNN is that they sometimes cannot capture the information needed to model long-term dependencies, such as in the sentence "I grew up in China… I speak fluent *Chinese.*". RNN could not predict the final word in this sequence because it is the name of a language, so the prediction model needs to look back at multiple preceding words in the sentence. RNN do not represent sufficient *depth* (length of the preceding sequence) [48] when the gap between the relevant information and the point where it is needed to become very large. Toward this goal, long short-term memory networks (LSTMs) are a specialized type of RNN adapted from the basic model described thus far in this thesis, which are capable of learning long-term dependencies. Unlike basic RNNs, LSTM-RNNs are better at learning mappings between aligned input sequences when they are related by very long-time lags of unknown lengths [44].

2.2.3 Transfer Learning

Transfer learning is a machine learning method which was developed for scenarios where one task is reused as the starting point for a model on a second task. Transfer learning involves the concepts of a domain and a task. Task can be defined that a model is asked to perform many tasks that related to but not the same as the task it was trained for [54]. A domain D consists of a feature space X and a marginal probability distribution $P(X)$ over the feature space, where $X = x_1, \dots, x_n \in X$ [54]. The knowledge contained in a program, which based on viewing computer programs as rational gents at a knowledge level [54]. The purpose of the knowledge level is to provide a toll for describing systems and predicting their behavior [54]. Procedure knowledge refered to the knowledge of how to perform a specific skill or task, and is considered knowledge realted wo methods, procedures, or operatino of equipment [55]. Morover, declarative knowledge defined as the facts or information stored in the memory,which is considered static in nature [55]. Task-to-task learning defined as the model for each joint independently rather than the whole body ar one time to edffectively transfer knowledge between tasks [56]. Predictive model is a process that use data mining and probability to forecast outcomes [57].

The general idea of transfer learning is inspired by the process by which human learners reuse and adapt source knowledge based on prior experience of one task or domain to accelerate or otherwise improve learning of a related task or domain. This idea motivates the development of algorithms to adapt knowledge between tasks or domains, which is expressed in procedural (policy) or declarative form. This thesis focuses on task-to-task transfer using declarative knowledge in the form of predictive models, learned using supervised and unsupervised methods for one or more source tasks. [58] Figure 2.9 shows how transfer learning uses given data and a source task to learn to perform a target task.

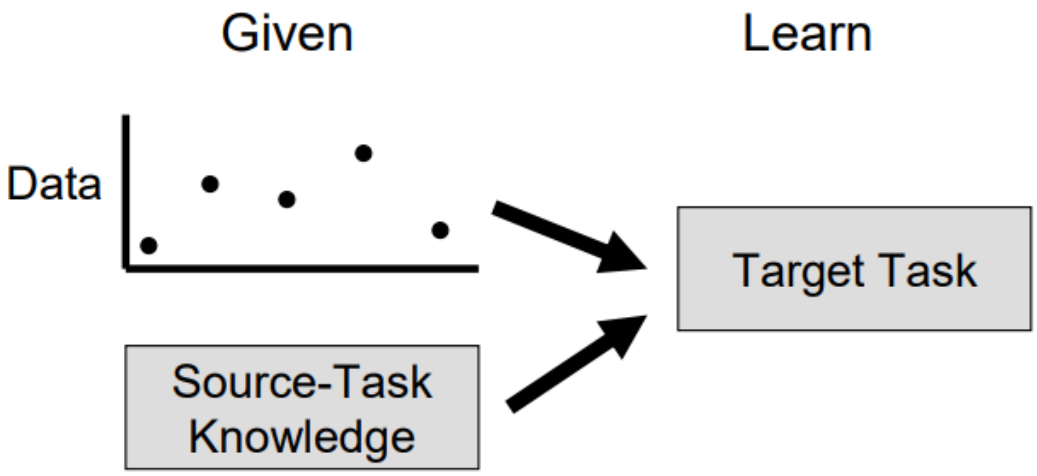

Figure 2.9: Transfer learning used data and source task to perform [58]

Transfer learning can be used in many fields, such as speech recognition, where it is used to adapt a trained acoustic model for one language to another language with little or no re-training [59]. Because deep learning has been used to achieve significant gains in the flexibility of learning representation for speech and language processing [51] [60] [61] [62], it can also be applied to the formulation of methods of transfer learning speech, especially in cross-lingual and multilingual transfer. Wang et al. (2015) mention that DNN-based multilingual approaches using English to code-mixed English/Chinese transfer learning on speech from speakers identified as having British/Chinese accents achieves greater improvement than just using DNN models on only English/Chinese speech [59].

An abstract depiction of this adaptation process is shown in Figure 2.10, where y_1 and y_2 represent two languages, and some data x_2 has been provided for the target task y_2 . The model is adapted in a language-merging phase from M_2, which is learned for an original distribution $P_2(x)$, to M_2', which is based on new data following a different distribution $P_2'(x)$ [36]. Wang et al. demonstrated a language pair (English/Chinese) for which features for recognizing code-mixed speech can be better learned using multiligual data, and for which each languge, training only the langugage-specific part is much easier than training the entire network.

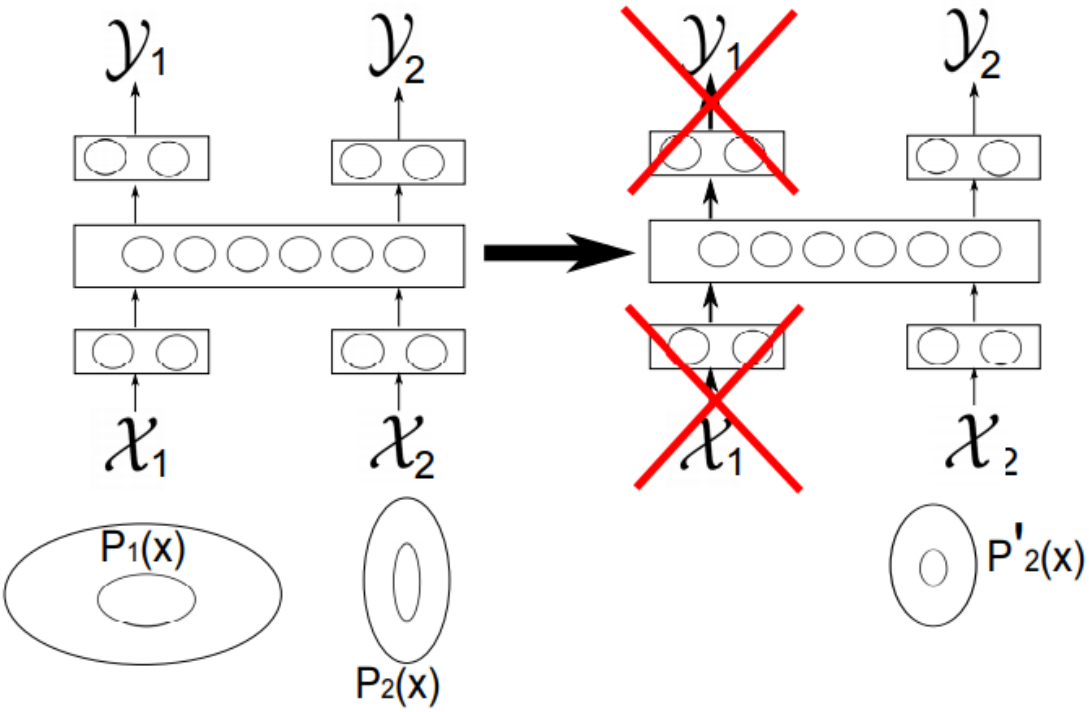

Figure 2.10:Transfer learning on multilingual or cross-lingual speech [59]

Chapter 3 Related Work

This section surveys directly relevant research on the central task of this thesis, code-mixed automatic speech recognition (ASR) for cognitive services. It is divided into two sections: one on approaches that adapt traditional multilingual ASR and use language identification (LID) as a subtask, and one on modern deep learning approaches.

3.1. Conventional Code-mixed ASR

Code-mixed speech recognition has been studied in the context of modeling acoustics, languages (text sequences), and pronunciation. As Section 2.1mentions, the most commonly used traditional approach to ASR is based on HMMs, GMMs, and probabilistic language models (e.g., Kalman filters based on text n-gram frequency). Thus, traditional ASR approaches to code-mixed speech recognition usually fall into two categories: multi-pass and one-pass recognition [63]. Multi-pass ASR performs the language identification (LID) task on code-mixed utterances to determine the language of each speech unit (and the boundaries of contiguous-language subsequences) before passing the result on to a language-dependent speech recognition module. Lyu et al. (1988) used a three-stage, data-driven phone clustering method to train an acoustic model for Mandarin-Taiwanese code-mixed speech [63]. Figure 3.1 shows how LID uses language boundary detection to tag Mandarin-Taiwanese utterances and dispatch each one to the either the Mandarin ASR module or the Taiwanese one to obtain the text of recognized speech. Later, Chan et al. (2005) used CMU's phone dictionary to detect English words and a separate Cantonese phone dictionary to detect Cantonese words, then concatenated contiguous words in the same language into phrases to mark the language boundaries between them [64]. Another approach using LID extracts features from code-mixed utterances, then builds a language-dependent recognizer for each language, and finally uses those two acoustic models to build a bigram language model [65]. In [66], Zhang (2012) uses LID information and acoustic information during the decoding process.

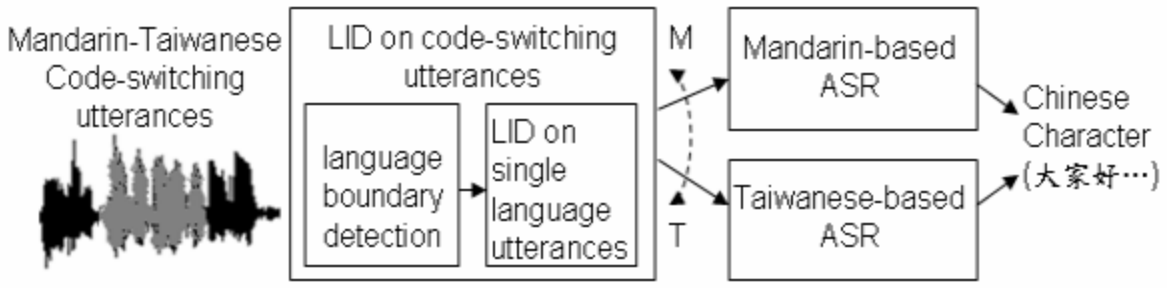

Figure 3.1: ASR for code-mixed speech in a multi-pass approach. [63]

One-pass ASR ignores the distinction between two languages, and instead uses lexical items between two languages. Considering the same use case from Lyu et.al of recognizing Mandarin-Taiwanese code-mixed speech, Figure 3.2 shows that the ASR takes Mandarin-Taiwanese utterances as input, then trains a bilingual acoustic model, a pronunciation model and a Chinese character-based data structure for search [63]. Moreover, Hindi-English code-mixed speech recognition also uses one-pass ASR that builds a model based on a language toolkit developed by CMU; they do not need to segment the speech by language [67], because their task objective is to express all words in English rather than as bilingual text. In [68], Sunit et al. used a phone-merging method which merged phone sets and matched similar sounds in Hindi and English words.

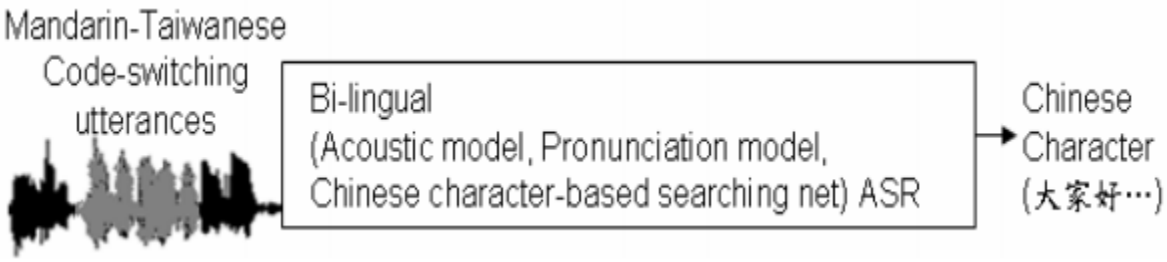

Figure 3.2: One-pass ASR speech recognition [63]

3.2. Deep Learning Code-mixed ASR

Code-mixed speech can also be trained using deep learning methods; for example, Yilmaz et al. use a DNN-HMM hybrid, as documented in Section 2.2.1, to train Dutch sentences mixed with Frisian words. They first trained and applied a GMM-HMM model to obtain the alignments for DNN training; they then use the DNN component to train on features obtained by applying MFCC features, which were trained using a GMM-HMM system, to language-

dependent phones [69]. The other approach is one that Wu et al. use to construct cross-lingual articulatory features (AF) that are incorporated into the estimation of the distance triphone in the training corpus. These AFs are extracted to construct an acoustic triphone HMM, then passed into a DNN that extracts the feature from them. Zeng et al. improve the N-gram language model by training a DNN with Mandarin sentences mixed with embedded English phrases [70]. Another relevant DNN architecture is the multilingual DNN (MDNN), which uses resources from other languages to develop acoustic models for an under-resourced language[26], as shown in Figure 3.3 [71].

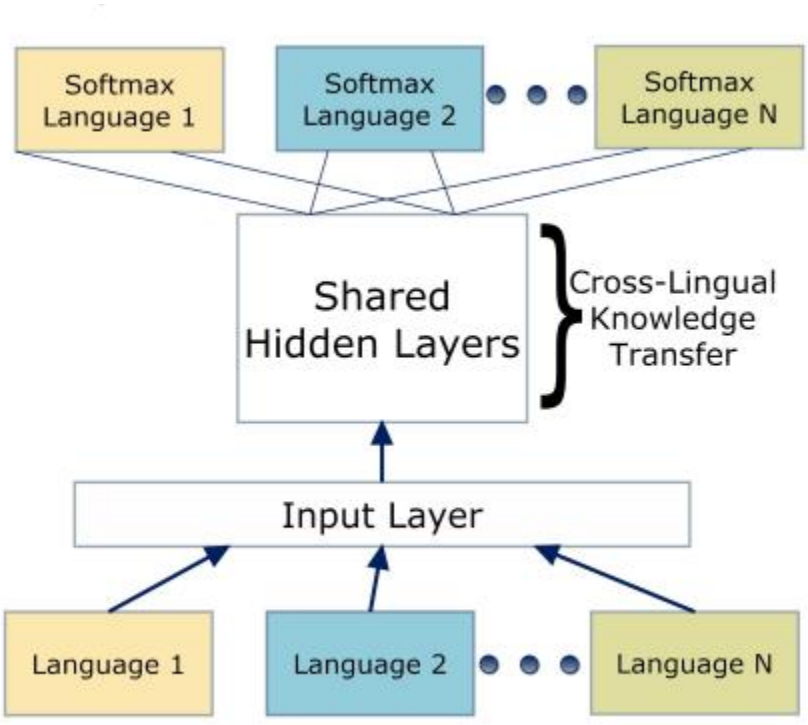

Figure 3.3: General framework for Multilingual DNN acoustic modeling [71]

RNNs are another approach for improving code-mixed speech recognition; Adel et al. [72] use a system consisting of an N-gram LM, RNN, and Part of Speech (POS) tagger, each trained on Mandarin-English code-mixed speech. Scultz and Vu integrate part-of-speech (POS) and language identification (LID) tags into RNN language models (RNN LMs) [27]components

[26] Under-resourced languages are defined as those suffering from an availability of services or resources that is less significant than that of other languages, for tasks such as text processing, speech processing, translation, and compilation of bilingual dictionaries, and usability dictionaries [113].
[27] RNN LMs are the result of integrating language identification and part-of-speech tagging over word sequences to predict not only the next word, but also its language [73].

each trained on Mandarin-English [73]. Furthermore, Hannun et al. proposed a new approach for speech recognition, *DeepSpeech*, which uses an end-to-end RNN. This RNN needs sufficiently large training labeled sets to get good results, and this method surpasses more complicated traditional methods [13]. They later extended this method in *DeepSpeech*, one of whose innovations is a Connectionist Temporal Classification (CTC) loss function that measures the quality of speech transcriptions from audio [15]. The CTC loss function is shown to raise model precision on the prediction task [28]for English and Mandarin speech recognition. A goal of the open-source Mozilla *DeepSpeech* project is to facilitate the development of new ASR systems for training many languages using new corpora and transfer learning. Third-party developers have used DeepSpeech to train models for code-mixed speech recognition tasks. For example, in unpublished communications (Mozilla forum posts), *DeepSpeech* users "Hamza" [74] and "Alam" [75] reported that they used the *DeepSpeech* code library to train ASR models for on Hindi-English or English-Urdu code-mixed speech using transfer learning.

[28] CTC network "predicts only the sequence of phonemes (typically as a series of spikes, separated by 'blanks', or null predictions)" [114].

Chapter 4 Methodology

This chapter presents the underlying methodology used in this thesis research, and is divided into three sections documenting: the language model, the acoustic model, and the deep learning architecture.

4.1. Language Model

As discussed in Section 2.1.3, language models (LMs) are used to choose which sequence of words make the most sense for a given use case. In Bayesian methods for multilingual ASR and other cognitive service and NLP tasks, the decoding and learning tasks are framed in terms of conditional probability as the basis of sense-making. In statistical language modeling, probabilities are estimated by calculating sequence likelihood, whether using generative or discriminative models[29]. In conventional ASR, GMM-HMM used discriminative model, while *DeepSpeech* uses generative models which is an audio file input rendered on 16kHz [76].

Because this thesis uses Mozilla *DeepSpeech* to construct and train the basic bilingual ASR model, it is necessary to generate the corresponding decoder files. These files include: (1) N-grams for the text transcript; (2) a binary file that encodes the trained parameters of the LM; and (3) a trie file encoding aligned sequence frequencies for the unified alphabets. This is a typical export format for LMs such as Heafield's *KenLM*, the one used in this research [77] [78] [79]. *KenLM* is a fast and compact language model toolkit and query engine for speech recognition tasks. Other language models include: SRILM, which is based on tries used in several decoders, and IRSTLM, a sorted trie implementation designed for lower memory consumption [80].

[30]KenLM's trie uses a popular reverse trie, in which the last word of an N-gram is looked up first. For example, as shown below in Figure 4.1 (the sentence is read leaf-to-root), for the phrase "*is one of*" in a reverse trie, the order of each node is sorted by vocabulary identifier but not alphabetically; thus, "*is*" always appears before "*are*". Nodes are also sorted in column-major

[29] In general, a discriminative model is used to approximate the decision boundary between classes, while a generative model explicitly estimates the actual conditional distribution of each class [112].
[30] A *trie* is defined as a search tree which stores a dynamic set or associative array by an ordered data structure that usually contains the keys as strings. "A trie data structure is a trie with bit-level packing, sorted records, interpolation search, and optimal quantization aimed at lower memory consumption" [80].

order; for instance, nodes corresponding to these N-grams appear in this order: “are one”, “<s> Australia”, “is one of”, “are one of”, “<s> Australia is”, and “Australia is one” [80].

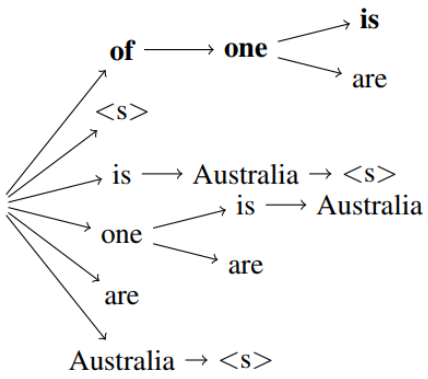

Figure 4.1: lookup of "is one of" in a reverse trie. [80]

In an N-gram language model, the next word is conditioned on a sequence of at most $N-1$ preceding words; thus, the model has states ranging over combinations of $N-1$ words, the maximum length of a prefix (node-to-leaf path). Because each word corresponds to a distinct state, syntactic ambiguity in natural language (when te decoder prunes many low-scoring partial hypotheses) may result in search errors (an encoder of a state has too many information to influence the computation of language model scores, so the decoder will be slower and find wrong result word). One technical contribution of *KenLM* is to provide a way to minimize these distinct states, by precomputing a state function over word sequences $w_1^n = (w_1, w_2, \dots, w_n)$. Heafield (2013) defines the suffix function $s(w_1^n) = w_m^n$ where w_m^n, is a suffix of w_1^n [80], for example, as Figure 2.3 shown, the phoneme <s> can occur in two different possible words: *sing* or *song*. The state function is applied to the query process then the application issues query which returns $s(w_1^n)$ [80]. Because all language model queries issued by machine translation decoders follow a left-to-right pattern, starting with either the beginning-of-sentence token or null context for mid-sentence fragments[31] [80]. Thus, KenLM also extends word states to store backoff information[32] for the problem as shown below:

[31] A *null context for mid-sentence fragments* is the set of phrases that are omitted from the middle of sentences.
[32] *Backoff information* refers to the probabilistic terms memoized to make queries efficient given an already-estimated model and minimal context.

$$s(w_1^{n-1}) = (w_m^{n-1}, \{b(w_i^{n-1})\}_{t=m}^{n-1}) \quad (4.1)$$

The computation of finding the longest matching entry w_f^n for query $p(w_n|s(w_1^{n-1}))$ is:

$$p(w_n|w_1^{n-1}) = p(w_n|w_f^{n-1}) \prod_{i=1}^{f-1} b(w_i^{n-1}) \quad (4.4.2)$$

The probability p of the next word w_n conditioned on all previous words w_1^{n-1} can be factored into a conditional factor of the next word w_n given the longest unique suffix w_f^{n-1} times the product of all conditionally independent factors for the prefix, where backoff penalties $b(w_i^{n-1})$ are given by an already-estimated model. Each visited w_i^n stores backoff $b(w_i^n)$ [80]. These are stored into the state $s(w_1^n)$ and returned so that they can be used for the following query [80].

Thus, this thesis uses KenLM to prepare aligned corpora for training and then export the corresponding parametric LM to be by operated on by *DeepSpeech*. In its learning phase, *DeepSpeech* updates the parameters generated by KenLM, and in its inference phase, it and decodes utterances into recognized text sequences (obtains transcripts from new utterances).

4.2. Acoustic Model

DeepSpeech is character-level speech recognition model, thus, a set of characters A (the alphabet) that consists of lowercase English and Chinese characters (Hanzi), which has length of $|A|$. Alphabetic characters are space-delimited, so that the character set is augmented: $A' = A \cup \{blank\ character\}$ with the length of $n = |A'|$ [81]. A temporal classification method is needs in order to properly insert blank characters for sequence alignment. Connectionist Temporal Classification (CTC) is one such method [15]. The model is trained using the CTC loss function to make sure all the audio aligns with the synchronous transcript, such that the correct transcript has a higher likelihood of being generated within this alignment. [82]. The input can be considered a sequence of T feature vectors $\{x_1, x_2, \dots, x_T\}$. Each feature vector in the sequence is the result of extracting mel-frequency cepstral coefficients (MFCCs) from an audio sequence that is typically about 20 milliseconds in length [81]. Figure 4.2 depicts a spectral file corresponding to the mel-frequency cepstrum (MFC), which is extracted from the signal and

provides a quantitative description of the audio that can also be used for symbol prediction [81]. The audio feature extraction stage that produces the MFC is the preliminary stage of the training process shown in Figure 4.3.

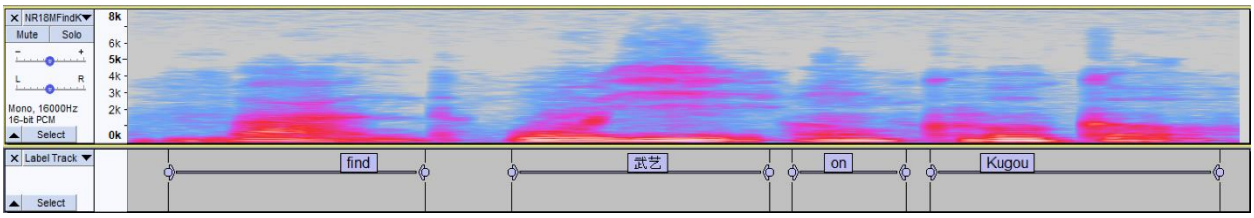

Figure 4.2:an example of spectrum from the data

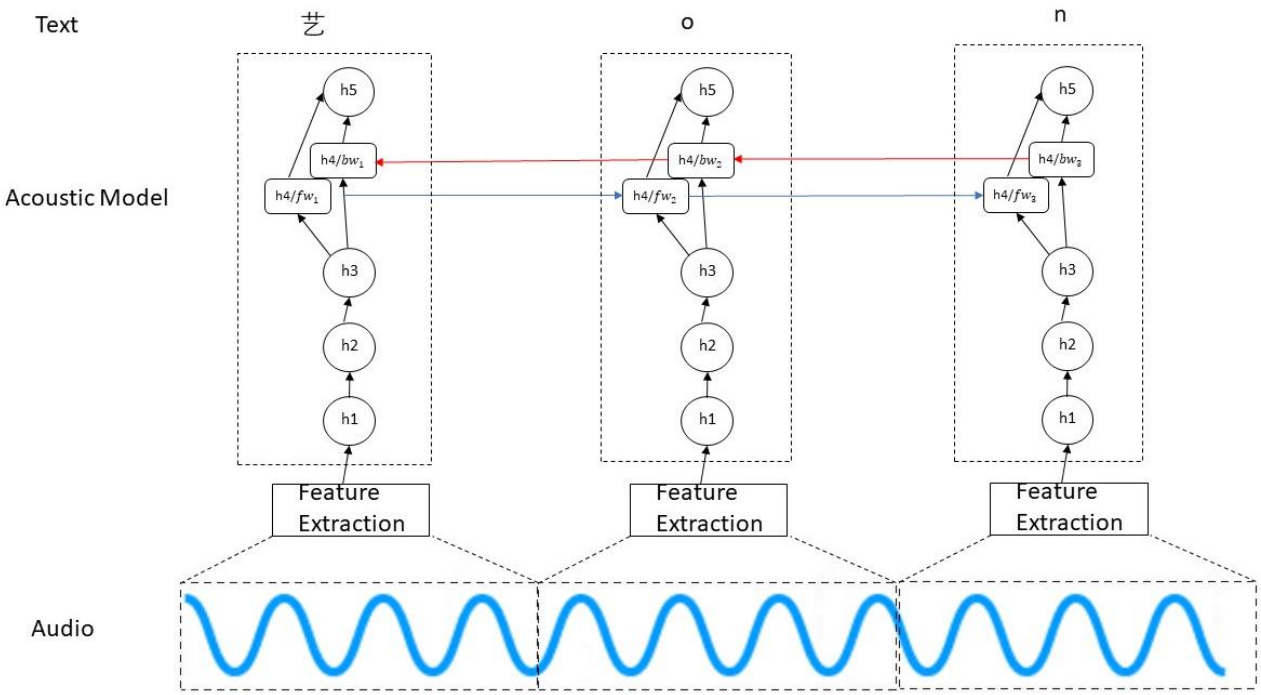

Figure 4.3:DeepSpeech Architecture [82]

4.3. Model Architecture

As Morais (2017) writes, "*DeepSpeech* is an end-to-end trainable, character-level, deep recurrent neural network (RNN)" [82]. In other words, *DeepSpeech* uses an artificial neural network with multiple recurrent layers which receives audio features as inputs and produces the characters of a text transcript of the audio as outputs. As Section 2.1.2 discusses, some ASR use

a phoneme-based pronunciation dictionary to build their language models; however, *DeepSpeech* is trained from scratch using supervised learning, achieving the same effect as a grapheme-to phoneme-converter to force alignment of the input sequences, without any explicit phoneme representation. This *ab initio* approach replaces sources of background knowledge such as phoneme dictionaries.

The RNN model contains five layers of hidden units: the input is fed into three fully-connected layers, followed by a bidirectional RNN layer, and a fully-connected layer (Figure 4.3). The training input is a collection $X = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots\}$ of utterances $x^{(i)}$ and labels $y^{(i)}$. Each utterance, $x^{(i)}$, is a time series of length $T^{(i)}$ where every time slice is a vector of audio features, $x_t^{(i)}$, $t = 1, \dots, T^{(i)}$. For an input $x_t^{(i)}$ to the RNN, the hidden units at layer l can be denoted $h^{(l)}$ with $h^{(0)}$ as the input [13]. The first three layers are not recurrent; in the first layer, at each time t, the output depends on the spectrogram frame x_t, along with a context of C frames on each side [13] (Figure 4.4). The remaining non-recurrent layers operate on independent data for each time step. For each time t, the first 3 layers can be computed as follows:

$$h_t^{(l)} = g(W^{(l)} h_t^{(l-1)} + b^{(l)}) \tag{4.3}$$

where $g(z)$ is the clipped rectified linear (ReLu)[33] activation function and $W^{(l)}$, $b^{(l)}$ are the weight matrix and bias parameters for layer l. The fourth layer is a bidirectional recurrent layer [83], which includes two sets of hidden units: a set $h^{(f)}$ with forward recurrence, and a set $h^{(b)}$ with backward recurrence:

$$h_t^{(f)} = g(W^{(4)} h_t^{(3)} + W_r^{(f)} h_{t-1}^{(f)} + b^{(4)}) \tag{4.4}$$

$$h_t^{(b)} = g(W^{(4)} h_t^{(3)} + W_r^{(f)} h_{t+1}^{(b)} + b^{(4)}) \tag{4.5}$$

In Equation (4.4), $h^{(f)}$ is computed from $t = 1$ to $t = T^{(i)}$ for the ith utterance, and in Equation (4.5), $h^{(b)}$ is computed in reverse from $t = T^{(i)}$ to $t = 1$. The fifth (non-recurrent) layer is the

[33] ReLu is a unit employing the rectifier which is an activation function defined as the positive part of the argument.

combination of the forward value $h_t^{(f)}$ and backward value $h_t^{(b)}$ as inputs, and the output layer is a function that yields the predicted character probabilities for each time slice t and character k in the alphabet. After the model outputs candidate predictions in the decoding phase, it needs to compute the CTC, which takes the probabilities of all candidate sequences found by the model and returns the most likely text sequence, to measure the achieved prediction error. CTC trained model's decoding can be transformed to frame synchronous Viterbi beam search, so network travers can be done for each frame in an overall optimized search space *[84]*. For example, as Figure 4.3 shows, at time step 0, the letter "艺" is most likely, then "o" is most likely at time step 1, and "n" is most likely happen at time step "2", so the output transcription will be given by the simplest possible for code-mixed "艺 on". The sole difference between Baidu *DeepSpeech* and Mozilla *DeepSpeech* is that Mozilla *DeepSpeech* uses LSTM at the cost of greater running time, but without achieving a comparably low word error rate. The Mozilla team attempted many fine-tuning improvements before moving on to LSTM; these improvements included binary search for the best-performing learning rate, and changing the priors for weight initialization and the size of the hidden layers. With all improvements combined, Mozilla *DeepSpeech* attains a Word Error Rate of under 10%, because Baidu *DeepSpeech* reached to Word Error Rate 5% *[76]*.

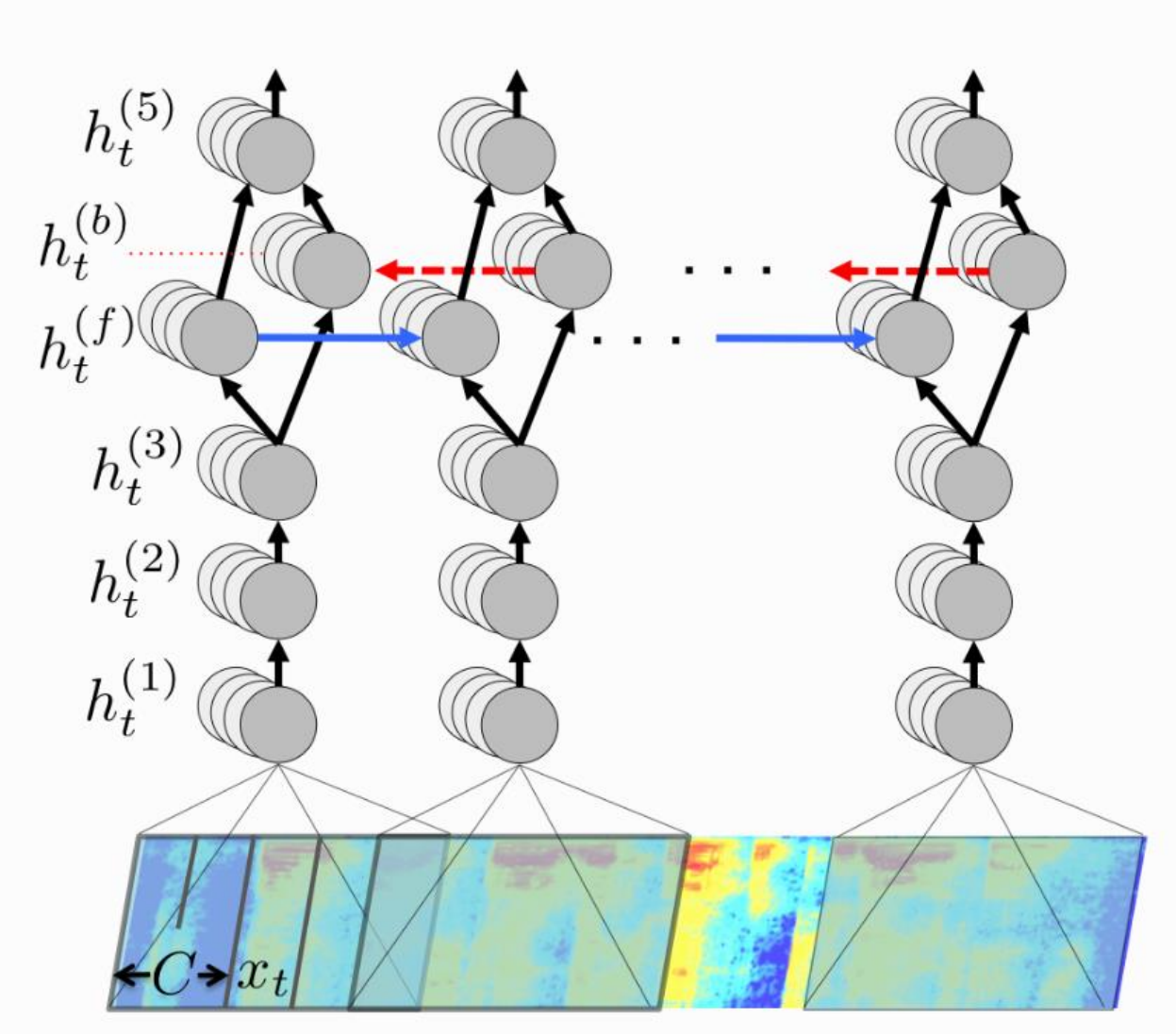

Figure 4.4: Baidu DeepSpeech structure of RNN model and notation [13]

Chapter 5 Experiment Design and Results

5.1. Experiment Design

This section surveys the experiment design and setup for training the *DeepSpeech* model, a process that includes corpus design, data collection, system setup, training, and validation.

5.1.1. Corpus Design and Data Collection

Some studies use data from the Linguistic Data Consortium (LDC) or collect corpora from conversations between volunteer speakers. Due to the specialized needs and limited resources of this project – specifically, the scarcity of English sentences mixed with Chinese named entities - I decided to recruit volunteers by distributing a survey (Appendix A). A central scientific purpose of this thesis is to assess the performance of multilingual ASR systems on English sentences mixed with Chinese names. Furthermore, having a concrete test bed based on a realistic cognitive service use case, with its own corpus and appropriate performance metrics, allows technical innovations to be systematically developed to improve the quality and accuracy of recognizing code-mixed speech in such services.

The first step of corpus design for this thesis was to create a survey which includes six questions. Five of these are fill-in-the-blank, four of them demographic: the respondent's age, gender, education level, whether they speak more than one language. The fifth and most important part of the survey is to ask participants to give their email address if they are interested in participating by contributing voice recordings to the project. This allowed me, as experimenter, to contact them by email to ask for them to participate. A sixth and final question was open-ended and asked users whether they had any comments about the research project. This question seeks to investigate the reflections of the audience about the project and curate a record of participants' opinions and remarks.

The corpus design also had to meet crucial participant constraints on this project: because every participant was a volunteer, they might not have enough time to record or hold conversations for hours. This indicated that transcripts needed to be generated from phrases instead of complete sentences. Thus, to reduce participant fatigue and save each participant some time, the audio component of the corpus was generated by recombining recorded phrases automatically instead of having training speakers read every combination [85]. The primitive

phrases (including single-word named entities) were manually created and consisted of: 100 Chinese names (chosen from Chinese popular singers on *Kugou*), 5 English commands (*play*, *search for*, *find*, *look up*, *who is*) and 5 English-named services (*Kugou*, *QQ*, *Pandora*, *Spotify*, *LastFM*).[34]

My preliminary attempt at corpus collection involved combinatorial enumeration of scripts. Each training speaker would have to read 2100 sentences ($(4\ (commands) \times 5(services) + (1(command{:}\ who\ is) \times 100\ (Chinese\ names) = 2100)$. As reading even 100 variations on "Play (Chinese name) on Spotify" would be tedious, this was soon deemed infeasible. As the literature on modern ASR systems such as Mozilla *DeepSpeech* Baidu *DeepSpeech* indicates [76], however, better results depend on having a corpus containing sufficient length of recorded voice data. A typically reported elbow point is 100 hours of recordings, which for tens of speakers means on the order of ten hours per speaker – an infeasible speaking time for a volunteer participant pool. Therefore, I started to split sentences (commands and services in the same sentence) into phrases or words, allowing each participant to record their voices by using a phone app[35], in no more than 3 minutes. The full audio recordings were then recombined using a script that enumerated the constituent phrases and concatenated them using utilities from the *PyAudio* library [86]. The resultant audio files for these synthesized sentences were then aligned with the text of the full sentences by *DeepSpeech*.

Distribution for the survey commenced in December 2018 after this thesis research was approved by an Institutional Review Board (IRB), and was advertised via the Chinese social media service WeChat, plus Facebook, Twitter, Kansas State University's International Student mailing list and personal social contacts, such as friends, colleagues, and classmates. The survey was closed to end recruitment of participants in mid-April 2019 and the last voice recordings were collected by the end of the month. This was to allow enough time to ask people to record their voices and clean the data. The survey got around 80 responses. Not everyone finished the survey, though: there were only 64 people finished and 45 people gave their email to indicate a willingness to record.

[34] To elicit voice recordings, the specifications of phrases to be read aloud were shared with participants as Google documents.

[35] All participants were asked to record their speech samples using their own devices. In most cases these were voice memo applications of the type available on most smartphones, which produce .m4a (MPEG-4 audio) files.

5.1.1.1. Speaker Information

There are 64 people for participating the survey with valid response, but 3 people recruited from these 64 people, other 42 people recruited from friends or family. Among the 64 participants in the survey, 9 have high school degrees; 1 , a trade/technical degree; 2, associate degrees; 23. , bachelor's degrees; 25, with master's degrees; and 4, doctoral degrees. As Figure 5.1 shows, most participants' highest education levels are bachelor's and master's degrees.

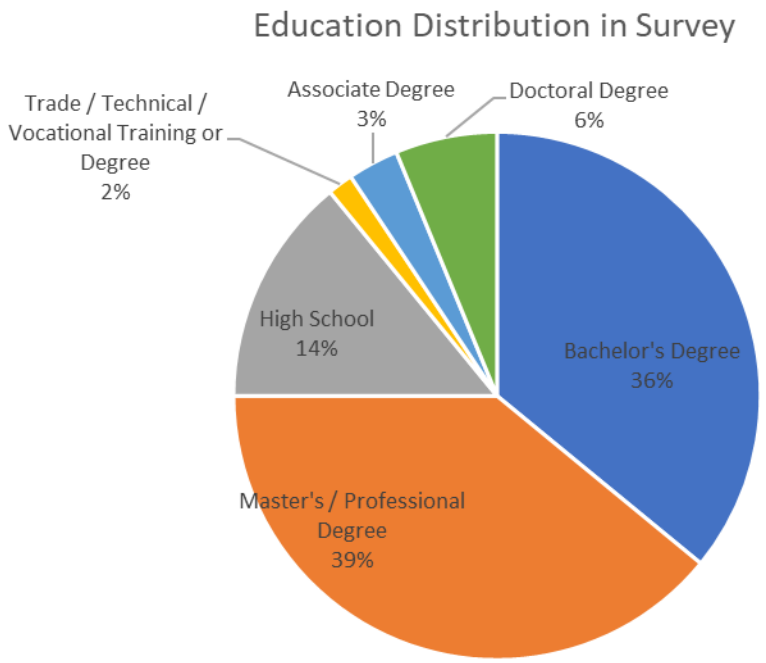

Figure 5.1: distribution of highest education level among 64 survey participants

Meanwhile, the survey also asked participants about their native languages and other languages they can speak. All participants were able to read English as a native or foreign language. Figure 5.2 shows that 21 respondents speak Chinese and English; 22 consider themselves bilingual and speak one language in addition to English, **other** than Mandarin; and 13 people speak more than two languages, including English. The statistics showed that the distributed survey included the right target group of people, and it also showed that there were numerous bilingual speakers. This also provides evidential support for premise of need for this thesis, stated in Section 1.4.2.

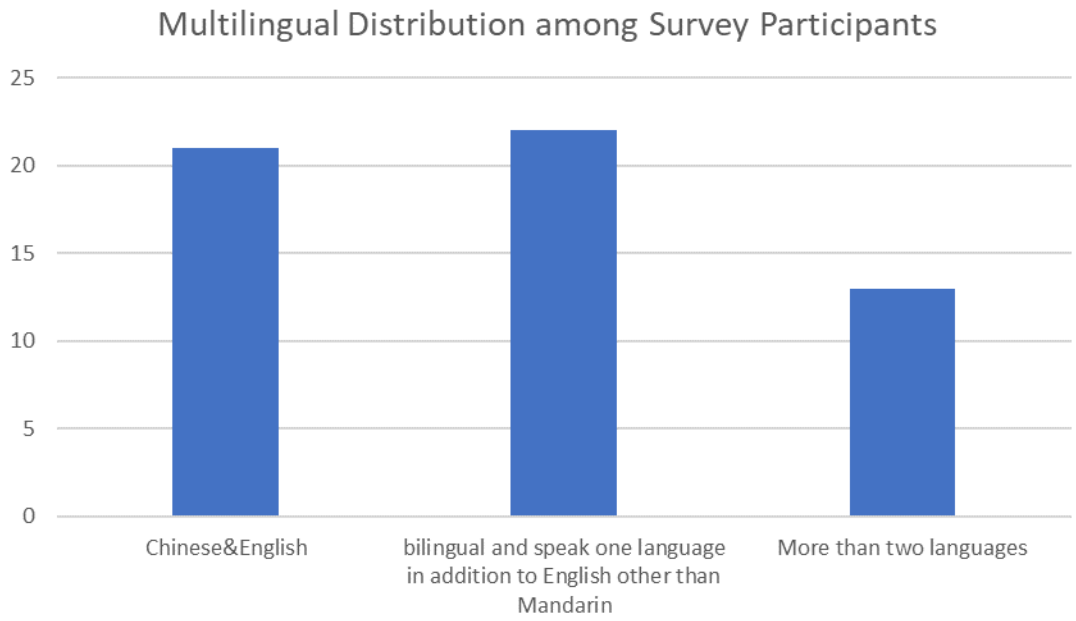

Figure 5.2:Multiligual Speakers Distributions

5.1.1.2. Age Information

The age distribution of the respondents to the survey varies, as shown in Figure 5.3, the largest age group of respondents is from 18 to 35, but some respondents are older than 40. As the literature indicates, multilingualism is becoming more common in younger generations (echo from Figure 5.2) **[87]**.

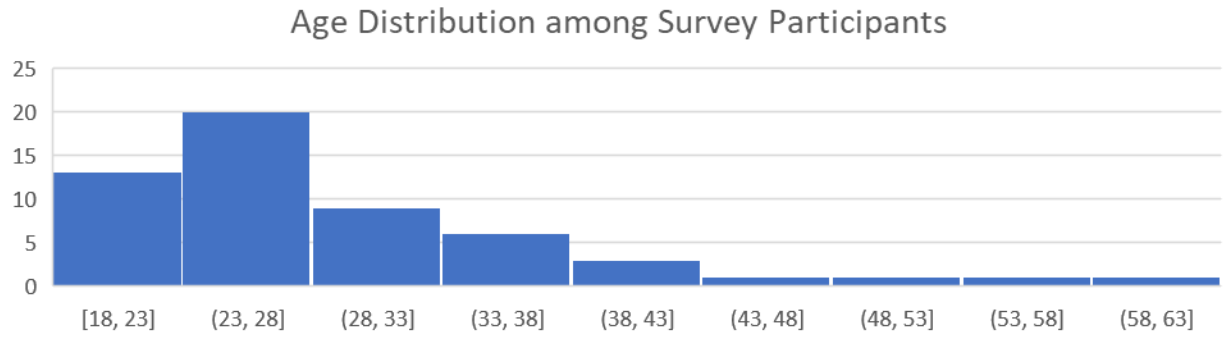

Figure 5.3: age distribution

5.1.1.3. Gender Information

According to the survey, because not every respondent was willing to record their voices, the distribution of gender is different between initial survey respondents and the actual volunteer pool from which the speech corpus was collected. As Figure 5.4 shown, 30 are female and 32 are male, there were slightly more male participants than female participants who took the survey; the set of participants who produced recordings, however, was gender-balanced (both are 21), as shown in Figure 5.5. There are only 3 people collected from survey, other 42 people are from friends with consent form though email or social media.

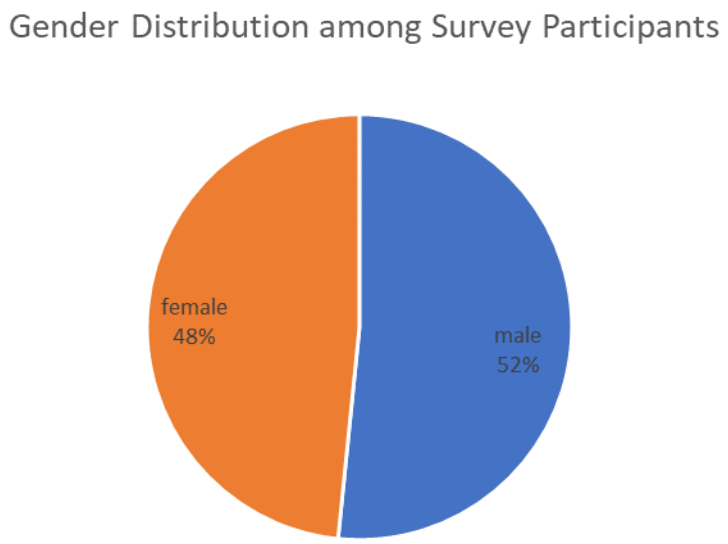

Figure 5.4: gender distribution in taking survey

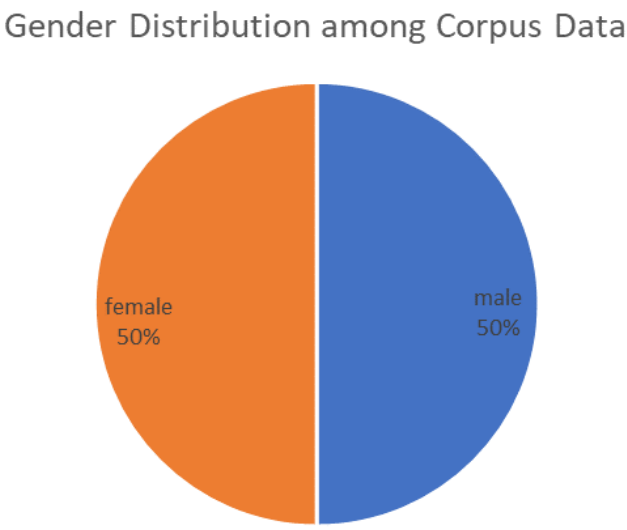

Figure 5.5:gender distribution in corpus

5.1.2. Data preparation

After collecting recordings through e-mail or online (via social media), I converted the files from M4A to WAV. I received recordings from 45 distinct individuals; however, I discarded three recordings due to: ambient noise or background voices; poor voice clarity, volume, and phrasing; and incorrect pronunciation of multiple original words. For example, some people recording their voices had a significant amount of background noises, were giggling when they recorded, or they did not pronounce words correctly (even allowing for alternate pronunciations) from the lexicon. Some recordings were recorded in low-pitched voices, possibly because the speakers recorded in a large room or were trying lowering their voices from normal volume (whispering or minimizing the sound of their voices in a workplace); an example of a waveform for this scenario is shown in Figure 5.6. By contrast, Figure 5.7 depicts an example of a good quality WAV file that has clear stops for pauses and clear frequencies for each word in the sample.

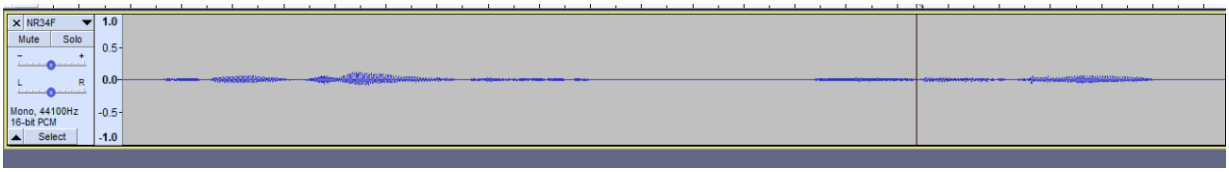

Figure 5.6: linear wave file

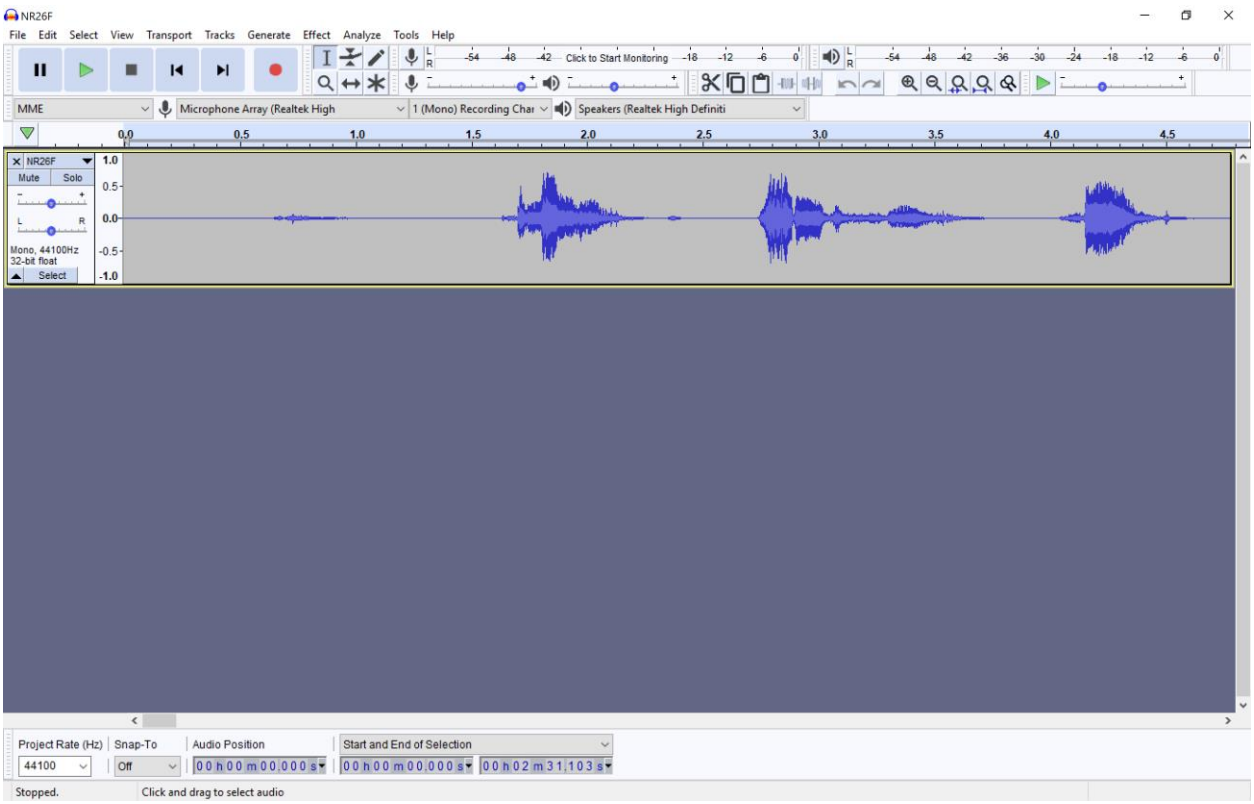

Figure 5.7:female high pitch wave file

The program that I used to trim each utterance is *Audacity*, a free software package for audio editing. For a WAV file such as the one shown in Figure 5.7, it provides an "*Analyze*" tool to segment sounds punctuated by silence (Figure.5.8). As shown in the dialog, this tool is controlled by adjusting the parameters associated with "treat audio below this level as silence" and "minimum duration of silence between sounds". This method can automatically label contiguous segments of sound from 1 to 110, so they can be exported as multiple WAV files, named by their labels. As I experienced while reviewing these automatically segmented WAV files to verify them by ear, some files need to be recut by hand if the speaker did not pause enough between each word when they recorded. Figure 5.9 shows an example of this, which had clear fundamental frequencies [36]for each word but pauses that were nonexistent or too short, resulting in segmentation errors.

[36] Fundamental frequencies mean the comparison of higher and lower point is clear to human eyes.

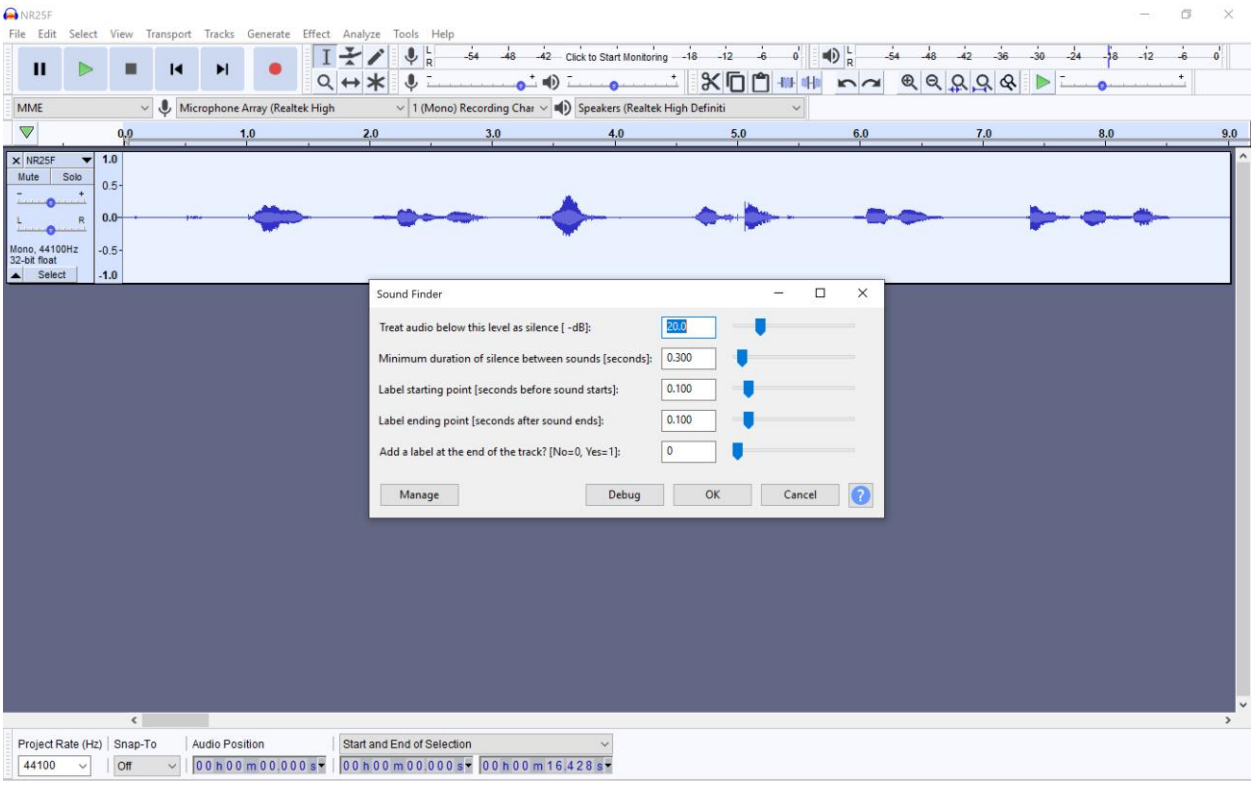

Figure.5.8:sound finder in audacity

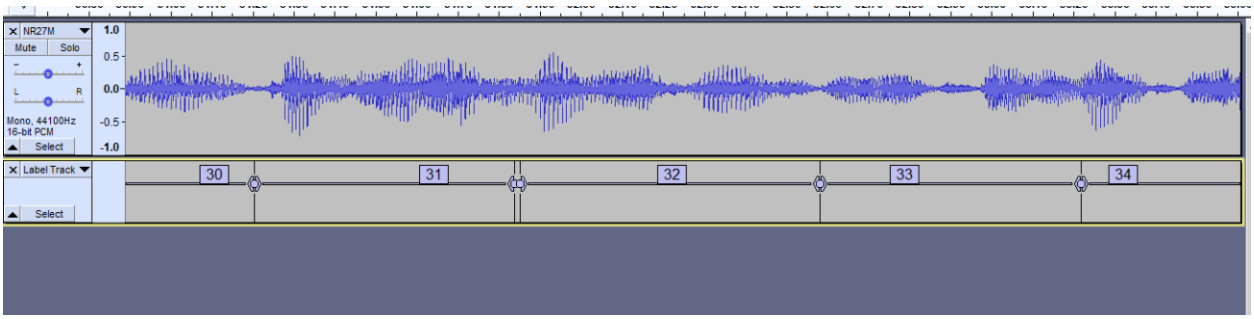

Figure 5.9: no pause corpus

When I labeled each corpus during the segmentation phase, I gave an ID for each named entity (musical performers in this test bed domain), numbered from 1 to 100, and wrote out the command phrases and service names as shown in Figure 5.10. I then used Python scripts to recombine them into 2100 sentences (an example of which is shown in Figure 5.11). The final corpus consists of recordings from 42 individuals. The names are formatted as in the following example: ***NR01FFindQ1***[37], where N stands for Normal (can be used for training), R stands for

[37] The combinations is: NR01FSearchS1, NR01FSearchK1, NR01FSearchL1, NR01FSearchP1, NR01FSearchQ1, NR01FFindS1, NR01FFindK1, NR01FFindL1, NR01FFindP1, NR01FFindQ1, NR01FLookupS1,

Recording, 01 is a speaker identifier, F is for Female, *Find* is the command (*Search*, *Lookup*, *Play*, *Whois*) Q is the QQ service (S for Spotify, K for Kugou, L for LastFM, P for Pandora), and 1 is the index of the Chinese name (Appendix B). Because each speaker has their own identity and gender in the filename, it is easy to track how many females and males are in the corpus in case I need to group any training data by gender to train models separately in future experiments.

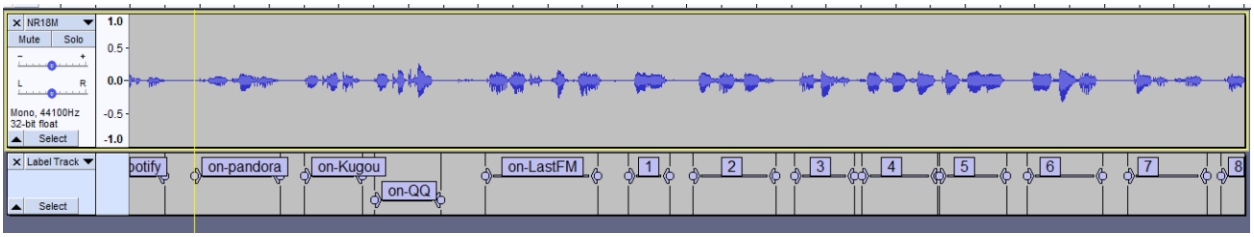

Figure 5.10: label data

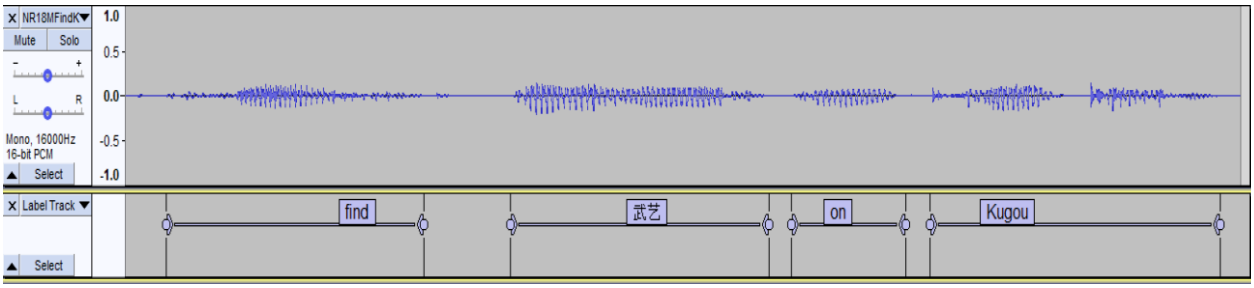

Figure 5.11:final code-mixed utterance

5.1.3. System Environment Setup and Training Process

Before I used *DeepSpeech* to train an RNN-based model, I explored DNN-HMM models as part of my literature review. As mentioned in Section 2.1.1, a precursor to using a DNN-HMM model is preparing pronunciation dictionary for GMM-HMM. For specific named entity task I developed as a test bed in this thesis, the existing CMU English pronunciation dictionary does not contain any of the named entities representing services, such as *Spotify*, *LastFM*, *QQ*, *Kugou* and *Pandora*, nor does the Chinese pronunciation dictionary contain any of the 100 personal names. Although it is possible to learn these from phonetic spellings or other feedback, as systems such as NetTalk do [88], this task is orthogonal to and beyond the scope of this thesis. The rationale is for monolingual tasks, upon which this work builds, RNN-based models such as

NR01FLookupK1, NR01FLookupL1, NR01FLookupP1, NR01FLookupQ1, NR01FPlayS1, NR01FPlayK1, NR01FPlayL1, NR01FPlayP1, NR01FPlayQ1, NR01FWhoisS1

DeepSpeech have already shown to outperform DNN-HMM model by a consistent margin of 3-4% [89]on both the Switchboard [90] and CallHome [91]benchmark data sets.

A premise of this work is that the benefits of using an RNN-based model accrue at the sub-phrasal level due to the learning representation that multi-layer temporal ANN implements. This is demonstrated by the overall performance of the RNN. Testing the further hypothesis that the margin of advantage over DNN-HMMs is maintained from monolingual tasks in previous work to the multilingual task in this work is beyond the scope of this thesis. If one uses only the extant framework of GMM-HMM or DNN-HMM by such these two pronunciation dictionaries, without additional generalization from the sub-word (phone) level to the word level, the WER would be 100%. This is not fair comparison; but the required implementations to make it a fair comparison – either training a subword level pronunciation model or manually encoding pronunciations of new named entities - are also not part of the main contribution of this work.

Mozilla *DeepSpeech* is based on the *TensorFlow* library, which is an end-to-end open source platform for machine learning developed by Google. The basic environment setup is on a clean instance of the Linux operating system with an edition of TensorFlow configured for DeepSpeech. This environment is more compatible with Pip than with Anaconda. This thesis used TensorFlow 1.13, CUDA 10.0, and cuDNN 7.4 as the base environment setup. Furthermore, the computer hardware requirements include at least an Intel Core i7 processor and a NVidia 1080 or better GPU; because the total corpus amounts to less than one hour of speech, I only use a single GPU instead of multiple GPUs.

My original results were obtained using holdout validation and test sets. I divided the data, consisting of 87,679 utterances, into three sets: train, dev[38], and test; the ratio was 70%, 20% and 10%. I split these data using a randomized `sort` command to get these three sets. These three sets are organized into three folders within the *DeepSpeech* data folder, and each folder contains a comma delimited file listing out the wav file path name, wav file size and transcript for each utterance. I developed a script for finding the unique words within a transcript, generating an alphabet file that is the one of configuration files in the training shell script. As Section 4.1 explains, I need to use the KenLM language model construction tool to

[38] **Dev** is the validation set, used as a criterion for stopping the training process.

generate training hyperparameters, which are saved as binary and trie files. We first create a file that would contain the transcripts corresponding to the utterances read by each speaker; I can then use KenLM to generate the corresponding binary file and trie file.

The following hyperparameters allow us to fine-tune the training process:

1. n_hidden: The number of nodes in the hidden layers of the neural network. "It is recommended that this parameter should be defined accordance with the amount and variety of training data" **[92]**. "In Mozilla *DeepSpeech*, the default value was set at 2048 for the English language training model." **[81]**
2. learning_rate: The standard learning rate in Mozilla *DeepSpeech* is 0.0001.
3. epoch: "The number of epochs, each of which is one complete forward pass and one complete backward pass of all the training samples in a data set." **[81]**
4. train_batch_size, dev_batch_size, test_batch_size: "The number of samples in each of the batches used in training, validation, and testing. The maximum value allowed depends on the available video memory (usually 8 to 64)" **[81]**.
5. dropout_rate: the probability of node dropout in one training step. Dropout[39] is an operation used to prevent the network from overtraining **[93]**. The default value is 0.3 **[81]**.

5.2. Results

This section will mainly present the results from experimental evaluation of the system described in Chapter 4, focusing on the collection of evaluation metrics that I previously adapted and specified for this new speech recognition task. These results center around both loss functions and the predictive accuracy of the model on the framing task, a multilingual cognitive service for music retrieval. I present both experimental baselines and comparative analyses of these baselines with variations on my experimental treatment.

5.2.1. Evaluation Metrics

Word error rate (WER) is the main metric applied in ASR evaluation [94]. The difficulty of measuring the performance for the speech recognition lies in the fact that recognized word

[39] *Dropout* is a simple and powerful regularization technique for neural networks and deep learning models, and it is a technique where randomly selected neurons are ignored during training [111].

sequence can have a different length from the reference word sequence. The WER works at the word level instead of the phoneme level, and is computed as an edit distance between two sequences: the reference sequence (ground truth sequence of words from the transcript) and the output sequence from the recognizer [95]. The specific edit distance I use is the Levenshtein distance metric over aligned word sequences – that is, words are the atomic units of the strings being compared. This WER metric calculates the number of word-level transformations needed to transform one sequence into another:

$$WER = d(s_{transcript}, s_{actual}) = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C} \tag{5.1}$$

where $s_{transcript}$ is the utterance of ground truth text, and s_{actual} is the actual text the model detects. As Table 5.1 shows, the WER counts the minimum number of substitutions, deletions, and insertions of words needed to get from $s_{transcript}$ to s_{actual} and divides this numerator the total number of sequences to get the rate of the word inaccuracies (Figure 5.12). The WER is calculated using dynamic programming *[96] [97]*.

Parameters	*Denotations*
S	word substitution count
D	word deletion count
I	word insertion count
C	correct word (alignment score) count
$N = S + D + C$	Number of reference words

Table 5.1: Denotations of parameters for WER

Correct transcript Who is 武 艺

Prediction transcript Who is 武 艺 亦 凡

$$\text{WER} = \frac{2\ words\ (insertion)}{4\ words(orignal\ legth\ of\ the\ sentence)} = 0.50$$

Figure 5.12:example of WER

A complementary evaluation measure to the WER, the *word accuracy (WAcc)* rate, is sometimes reported. This is simply 1 minus the WER, the accuracy measure corresponding to this error measure:

$$WAcc = 1 - WER = \frac{(N - S - D) - I}{N} = \frac{C - I}{N} \quad (5.2)$$

Another measurement metric used in ASR is the character error rate (CER), which is calculated in the same way as WER above but with character strings and edit distance units. The CER is used as a more sensitive evaluation metric for the final performance instead of word error rate because ASR outputs and ground truth can also be aligned and matched at the character level, even if the language model was trained based on words only. This is useful because word-based language models tend to achieve better-seeming performance in applications but do not account for low-level errors that may lead to task failure (e.g., in a cognitive service). [98]. The limitation of CER is that the length of words is correlated with the CER, so that more characters in the alphabet will cause more recognition errors. This in turn is related to perplexity, which we use as an estimate of task difficulty, and which Klakow and Peters (2002) showed is related to WER (or CER for character-level alignment) by a power law [94].

5.2.2. Results

To get better results from training data, several factors can be adjusted empirically by controlling the data preparation process. These include the format of audio portions of speech corpora, the quality of each utterance, and the number and length of speech samples in each corpus, each of which can have a significant effect on recognition accuracy of sentences. Before I trained the model, I used 16 bits and arbitrary dynamic ranges across speech samples (from about 22KHz to 44+KHz); this disparity resulted in a *TensorFlow* compatibility error. I therefore needed to convert all = WAV files to a consistent bitrate and dynamic range (e.g., 16bits, 16kHz) and monoaural sound instead of stereo, as used by *DeepSpeech*.

The quality of each utterance was sometimes poor because some people recorded the phrases with insufficient interphase pauses; therefore, I need to edit some audio of overlapping names by splicing in silence to better delimit them. Another problem is that some speakers vary in pronunciation clarity and accent for each language spoken; for example, some do not

enunciate the, English part clearly and some read Chinese names with nonstandard or regionalized pronunciations. Moreover, the total length of recorded speech in the corpora was only 48 minutes; as mentioned above, achieving high accuracy has often been shown to require on the order of a hundred hours of speech or more [76] [74]. These requirements can be even more stringent for code-mixed task: for example, for a Hindi-English mixed dataset that contained 2000 hours of speech, and for which the experimenter achieved a test WER of about 0.24 and a CER of 13.90 [74], a Mozilla *DeepSpeech* developer advised her to collect more hours of speech for training and hypothesized that that the reason her system could not yet recognize certain words was because of less frequent usage in her existing training speech.

The number of hidden units and batch size can significantly influence the accuracy of the model. I started the number of hidden units from 375, then increased them 512,1024, and finally 2048. I started the batch size from 4, then increased it to 48, and finally to 80. Raising the number of hidden units translates directly into a higher number of trainable parameters; the rationale for doing so is to attain the requisite capacity for the learning representation used. Meanwhile, a higher batch size and exposes the RNN to a larger, more diverse set of observations (combined across utterances and speakers) per training epoch. A higher batch size can provide a stable enough estimate of what the gradient of the full data set would be, while a lower batch size will jump around if the data is noisy [99]. The testing loss dropped from 2.3 to 0.19 when the number of hidden units increased from 1021 to 2048. The cumulative testing WER decreased as the number of hidden units and batch size increased (Table 5.2).

Number of hidden units	**Batch size (train, dev, test)**	**WER**	**CER**	**Loss**
375	80, 80, 40	0.019050	0.012940	2.181961
1024	48,40,40	0.039543	0.017363	2.364254
2048	48,40,40	0.009682	0.008435	1.375311
2048	80, 80, 40	0.002081	0.001976	0.198724

Table 5.2: The cumulative test scores for 8768 utterances

The WER varied for each test utterance: the observed range during testing was from 0.5 to 0.00, as shown in Table 5.3. Later, I recorded another 100 test sentences for speaker-

independent validation (Table 5.4). This validation run was purely qualitative, without WER or sentence-level accuracy measures. These two tables show that the model detects Chinese personal names better than English command and service names; in most cases, the personal names and service name are correct, but not the command, or the command and personal names are correct but not the service name. Among the newer 100 utterances, there are about 21% with 0.00 WER, and more than half had the command wrong but not the personal name and service names. As mentioned above, at least hundreds of hours of training speech would be needed to achieve a WER suitable for industrial use. Thus, given the speaking time of my transcript, I would need at least 5000 speakers, each recording the same lines, to attain a usable accuracy for full sentence recognition on this code-mixed task.

Utterance transcript	**Test Utterance**	**WER**
who is 武艺	who is 武艺亦凡	0.5
Find 武艺 on lastfm	find 吴亦 on lastfm	0.4
look up 张杰 on qq	Look up 张学友 on qq	0.33
Find 陶喆 on kugou	Find 陶喆喆 on kugou	0.20
Who is 周柏豪	Who is 周柏豪 on	0.167
Find 孙楠 on spotify	Find 孙楠 on spotify	0.00
Look up 罗大佑	Look up 罗大佑	0.00

Table 5.3: Test result example

Utterance transcript	**Test Result**
Play 武艺 on Spotify	play 武艺 on spotify
Play 易烊千玺 on spotify	play 易烊千玺 on spotify
Play 王俊凯 on spotify	Look 王俊凯 on lastfm
Play 邓紫棋 on spotify	Play 邓紫棋 on spotify
Play tfboys on lastfm	play tfboys on lastfm
Play 张杰 on lastfm	Look 杰 on lastfm
Play 五月天 on spotify	play 武月天 on spotify

Play 迪丽热巴 on spotify	Search 李宇宇热巴 on spotify

Table 5.4: simultaneous representative utterance test

The above results were collected using the original set of all utterances that were shuffled and then split into three sets. I also applied 3-fold cross-validation across utterances, where each fold had about 29226 utterances across all 42 speakers (who are represented in training, dev/validation, and testing segments, still in the ratio: 70% train, 20% dev and 10% test). I also tried 3-fold cross-validation by splitting the data set **by** speaker instead **across** speaker, so that all the data from each speaker belonged to the training, validation, or testing segment (30 training, 8 validation, 4 testing), but this produced very poor results. The reason is that the error rate for a fully speaker-independent system is much usually higher than that for a speaker-dependent or speaker-adaptive system, and using speaker-independent training speakers usually produces less initial training data [100].

The results of 3-fold cross-validation (splitting **across speakers** and **by utterance**) are shown in Table 5.5 below. Compared with the holdout method for data splitting documented in Section 5.1.3, the validation WER results is generally comparable[40] in trend to those for holdout-based validation, but are not stable across folds (standard deviation: 1.098 (set 1), 0.17 (set 2), 0.38 (set 3)). This issue can be solved by increasing the number of speaking hours of the corpus.

To estimate the potential for scalability and continued improvement given larger corpora, I also did an ablation study using 23 minutes of speech samples selected uniformly at random from among the full set of utterances measuring 48 minutes in duration (Figure 5.13). Table 5.6 shows that the loss and WER achieved by training on 23 minutes of speech are higher than those achieved using 48 minutes.

[40] Set 1 dose not have full sentence correct ones, but it has about 37 out of 200 testing corpora that miss preparations "on", and other parts are correct. Set 2 has **97 out of 200 testing corpora fully correct**, and this is higher than the original set (21 out of 100). Set 3 has **139 out of 200 testing corpora fully correct**, which is even higher than original and set 2.

Fold set	Training validation loss	Training validation standard deviation	Training validation mean	Test set WER	Test set CER	Test set loss	Test set Highest WER (200 count)	Test set Lowest WER (200 count)
1	4.13	1.098	1.69	0.00662	0.00395	0.68	0.75	0.14
2	1.84	0.17	0.54	0.00305	0.00262	0.42	0.67	0.00
3	0.48	0.38	0.81	0.00207	0.00231	0.39	0.75	0.00

Table 5.5: three-fold training and testing

Minutes	Loss	Standard deviation	Mean	Test set WER	Test set CER	Test set loss	Test set Highest WER (200 count)	Test set Lowest WER (200 count)	Test set Lowest loss (200 count)
23	13.8	0.24	14.17	0.42	0.17	14	0.8	0.43	91.35
48	0.22	0.18	0.54	0.003	0.002	0.199	0.5	0.00	19.76

Table 5.6: ablation study comparison

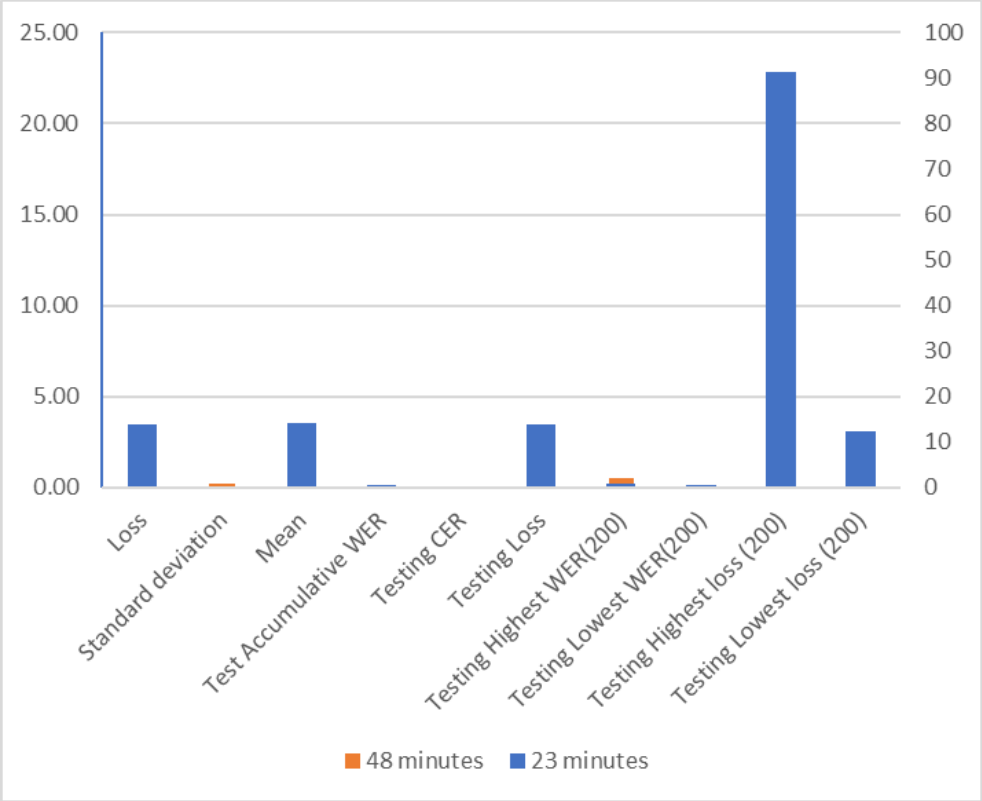

Figure 5.13: ablation study for different sets

5.2.3. Trade-offs

The main purpose of *DeepSpeech* is to align the transcript of a sentence with the corresponding WAV of the recorded utterance, and then, through a process of training, to learn the specific patterns of speech in the acoustical waveform. This learning approach entails multiple trade-offs: one is that if the input utterance sample rate increases to 44.1kHz, the model detection will fail due to sample rate is different as 16kHz. For example, as Table 5.4 shown above, if the sample rate is 44.1kHz, the first example of the table would show "search 武艺 on spotify" instead of "play 武艺 on spotify". After down sampled the speech recordings submitted by participants in my study to 16kHz, the system successfully recognizes test cases with a WER of between 0 and 0.5 (the mean WER over 8768 cases is 0.537411 and valid loss is 0.215866). Another trade-off is that the pronunciations for some Chinese names are correct but not the characters, such as in the sixth row of Table 5.4, where they utterance is fully and correctly recognized, but not the characters. When a Chinese name contains four characters rather than the more common two or three, sometimes the system fails to detect the full name due the reading

errors by the training speakers. For example, the last example of Table 5.4 is incorrectly recognized; this is because more than three people read the personal name (迪丽热巴) incorrectly as 迪丽巴热, reversing the proper order of the final two characters are flipped. In the process of cleaning the user-submitted speech data, I chose to reorder the recorded words to correct these faulty training examples, but splicing these words into the right character order resulted in some discontinuity in the speech waveform for independent characters. Furthermore, some people did not pause and spoke too quickly, resulting in the blurring together of contiguous spoken names; because recordings were unsupervised, and it was infeasible to contact participants for do-over recordings, I used manual segmentation. This process thus involved some tradeoffs to obtain distinct, recognizable names. Due to these tradeoffs, the model does not attain higher accuracy than present competitors, such as Baidu *DeepSpeech* [76] and Lyu et al.'s [9]; however, based on recommended practice, more training data (hundreds of hours or more of recorded speech) should result in improved performance [74].

Chapter 6 Conclusions and Future Work

This chapter will interpret key findings discussed in the experiment chapter and present conclusions of the thesis that are drawn from these findings along with lessons learned that indicate future directions for research. The future work section will describe tasks for scaling up, extending, and improving the efficiency of problem solutions that were demonstrated using the initial test bed I developed in this thesis. I also discuss applications based on my proof of concept system that connect the problem I have addressed with current interactive cognitive services.

6.1. Conclusions

I investigated different approaches for code-mixed speech recognition and found that deep learning ASR simplifies and streamlines the process of model development compared to conventional ASR. For example, phoneme dictionaries, which are needed in traditional HMM-based ASR, are elided in the deep learning-based system. I conducted some preliminary explorations using conventional ASR with HMMs and phoneme dictionaries. I found that in order to expand the existing pronunciation dictionary to incorporate Chinese names that occurred in my speech task but which I did not already have in a phoneme dictionary, I would need to enumerate and collect speech samples for all alternate pronunciations for every name.

For my task, this problem was restricted to personal names, but in general, it proliferates with the set of named entities to be recognized. This cause higher WER and failure to correctly detect sentences, making traditional ASR methods impractical to scale up for this task. However, the *DeepSpeech* model uses pattern transcript alignment to force the machine learning system to acquire the correct pattern for the utterance to achieve higher recognition accuracy. This methodology does not need a dictionary or language boundary detection, but will treat the code-mixed utterances as a new language and learn from observed patterns by training on large corpora.

The system developed in this thesis does not do excellent job as a recognizer of code-mixed speech due to lack of training data: the corpus I developed contained a total only 48 minutes of recorded speech. It does, however, achieve higher accuracy than reported for code-mixed Hindi-English ASR, [74] [6] where a current model achieves a WER of about 60%. As a

proof of concept for using deep learning to implement ASR on code-mixed speech for multilingual cognitive service tasks, this demonstrates that an RNN approach has potential.

Another trade-off is that, as mentioned above, in order to acquire enough speech data, I had to collect bilingual speech from speakers who knew just a few words of English, so that their accents and pronunciation quality adversely affected data quality. In addition, some participants in the study read names incorrectly from the transcripts I prepared, necessitating the culling of three participants (out of 45) from among the training speakers. In summary, the deep learning RNN method worked on a code-mixed speech recognition task, though it would need much more training data to decrease the WER to less than 10% as *DeepSpeech* does for its original monolingual task [82].

6.2. Current and Future Work

One major area of future work to make the kind of ASR system I have studied in this thesis practicable for code-mixed cognitive service applications is improving the corpus size and grammar complexity for the task. Currently, I only use very short, simple sentences for code-mixed speech recognition; in future work, one of my goals is to find more participants to have conversations recorded that consist of Chinese words mixed with English sentences. This will facilitate a natural scaling up of complexity in grammar and word use (including word sense[41]). We can build a web site to ask people to contribute code-mixed speech for free narration, reading of transcripts, or similar tasks. Because *DeepSpeech* needs large corpora (with a large number of hours of speech, training speakers, and different utterances) to decrease error rate, a general objective is to obtain least 100 hours of recorded speech for future experimentation.

A second major objective of future work is to build a talk app that can connect with current popular home assistant products, such as Amazon Alexa and Google Home. I was inspired by the *Pikachu* talk app to build a similar app that lets users can use bilingual code-mixed language to perform some simple commands, such as play songs or search for Chinese names on the web. Furthermore, I would also like to implement a reusable and extensible

[41] In linguistics, *word sense* is the indented meaning of a word, where the correct meaning among multiple options can be inferred from contextual information, [118]

(especially speaker-adaptive) model that can train on different combinations of languages for code-mixed speech recognition.

Last but not least, the training aspect of current work is only a preliminary experimental use of deep learning. In future work on deep learning, my objective is to improve performance on the transfer learning task by extending the current *DeepSpeech* library to incorporate transfer learning algorithms for our scenario. A relevant further goal, for engagement of more researchers in the deep learning community, is to develop a deep learning model inspired by *DeepSpeech* that uses *PyTorch* rather than *TensorFlow* to make the training procedure more developer-friendly for research domains such as code-mixed speech recognition.

Bibliography

[1] "Multilingual People," 2018. [Online]. Available: http://ilanguages.org/bilingual.php. [Accessed 8 6 2019].

[2] K. Waddell, "Bezos' big blunder: Amazon's Alexa is getting clobbered," 1 7 2018. [Online]. Available: https://www.axios.com/smart-speakers-voice-assistant-china-amazon-google-alibaba-baidu-beb83c4f-d3d2-41d8-98df-10ca73f2e129.html. [Accessed 14 6 2019].

[3] H. Adel, N. T. Vu, K. Kirchhoff, D. Telaar and T. Schultz, "Syntactic and semantic features for code-switching factored language models," in *IEEE Transac- tions on Audio, Speech, and Language Processing*, 2015.

[4] J. J. Ohala, "Coarticulation and Phonology," *Language and Speech,* Vols. 36(2, 3), pp. 155-170, 1993.

[5] N. T. Vu, D.-C. Lyu, J. Weiner, D. Telaar, T. Schlippe, F. Blaicher, E.-S. Chng, T. Schultz and H. Li, "A first speech recognition sys- tem for mandarin-english code-switch conversational speech," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. IEEE*, 2012.

[6] K. K. Bhuvanagiri and S. K. Kopparapu, "An Approach to Mixed Language Automatic Speech Recognition," in *ResearchGate*, 2015.

[7] W. Hsu, "William Hsu's Facebook Page AI Fails," 06 06 2019. [Online]. Available: https://www.facebook.com/hsuwh/media_set?set=a.10105789822800661&type=3. [Accessed 18 6 2019].

[8] J. Y. C. Chan, P. C. Ching, T. Lee and H. Cao, "Automatic speech recognition of Cantonese-English code-mixing utterances," in *INTERSPEECH 2006 - ICSLP*, Pittsburgh, Pennsylvania, 2006.

[9] D.-c. Lyu, R.-y. Lyu, Y.-C. Chiang and C.-N. Hsu, "Speech Recognition on Code-Switching Among the Chinese Dialects," in *Conference: Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference*, 2006.

[10] D. Wang, Z. Tang, D. Tang and Q. Chen, "OC16-CE80: A Chinese-English Mixlingual," 27 Spetember 2016. [Online]. Available: https://arxiv.org/pdf/1609.08412v1.pdf. [Accessed 18 06 2019].

[11] H. M. Meng, B. Chen, S. Khudanpur, G.-A. Levow, W.-K. Lo, D. Oard, P. Schone, K. Tang, H.-m. Wang and J. Wang, "Mandarin–English Information (MEI): investigating translingual speech retrieval," *Computer Speech and Language*, vol. 18, pp. 163-179, 2004.

[12] J. Konik, "Best Speech-to-Text Software: Dictating Results," 27 August 2018. [Online]. Available: https://www.cloudwards.net/best-speech-to-text-software/. [Accessed 18 06 2019].

[13] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates and A. Y. Ng, "Deep Speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv: 1412.5567v2*, 19 December 2014.

[14] "QnA Maker knowledge base limits and boundaries," Microsoft Azure, 21 05 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/limits. [Accessed 10 07 2019].

[15] D. Amodei, R. Anubhai, C. C. Eric Battenberg, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun and P. LeGres, "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin," 2015. [Online]. Available: https://arxiv.org/abs/1512.02595#aHR0cHM6Ly9hcnhpdi5vcmcvcGRmLzE1MTIuMDI1OTVAQEAw. [Accessed 25 06 2019].

[16] S. Karlin and H. M. Taylor, A first course in stochastic processes, New York: Academic Press,Inc., 1975.

[17] X. Huang, J. Baker and R. Reddy, "A Historical Perspective of Speech Recognition," *Communication of the ACM,* vol. 57, no. 1, pp. 94-103, 2014.

[18] K.-F. Lee, H.-W. Hon and R. Reddy, "An Overview of the SPHINX Speech Recognition System," in *IEEE Transaction on Acoustic Speech, and Signal Processing*, 1990.

[19] N. Morgan, H. Bourland, S. Greenberg, H. Hermansky and S.-L. Wu, "Stochastic perceptual models of speech," in *ICASSP 1995*, 1995.

[20] N. A. Gershenfeld and A. S. Weigend, The future of Time Series, Addision-Wesley, 1993.

[21] M. Gales and S. Young, "The Application of Hidden Markov Models in Speech Recognition," *Foundations and Trends in Singal Processing,* vol. 1, pp. 195-304, 2008.

[22] J. A. Bilmes, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," *Berkeley, CA: International Computer Science Institute,* pp. 7-13, 1998.

[23] S. H. Deshmukh and D. S. G. Bhirud, "Analysis and application of audio features extraction and classification method to be used for North Indian Classical Music's singer identification problem," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 3, no. 2, pp. 5401-5406, 2014.

[24] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics Speech and Signal Processing,* vol. 28, pp. 357-366, 1980.

[25] "Phoneme Detection, A Key Step in Speech Recognition," 22 11 2017. [Online]. Available: http://www.authot.com/en/2017/11/22/phoneme-detection-speech-recognition/. [Accessed 19 6 2019].

[26] A. contributor, "Why isn't voice recogniton software more accurate?," Quora, 20 12 2016. [Online]. Available: https://www.forbes.com/sites/quora/2016/12/20/why-isnt-voice-recognition-software-more-accurate/#47ac72b91c6a. [Accessed 10 07 2019].

[27] I. M. M. El-emary, M. Fezari and H. Attoui, "Hidden Markov model/Gaussian mixture models (HMM/GMM) based voice command system: A way to improve the control of remotely operated robot arm TR45," *Scientific Research and Essays,* vol. 6, no. 2, pp. 341-350, 18 January 2011.

[28] l. M. M. El-emary and M. F. a. H. Attoui, "Hidden Markov model/Gaussian mixture models (HMM/GMM) based voice command system: A way to improve the control of remotely operated robot arm TR45," *Scientific Research and Essays,* vol. 6, no. 2, pp. 341-350, 18 January 2011.

[29] M. Vyas, "A Gaussian Mixture Model Based Speech Recognition System Using Matlab," *Signal & Image Processing : An International Journal (SIPIJ),* vol. 4, 08 2013.

[30] "Merriam-Webster phoneme," [Online]. Available: https://www.merriam-webster.com/dictionary/phoneme. [Accessed 19 06 2019].

[31] "The 44 Phonemes in English," [Online]. Available: https://www.dyslexia-reading-well.com/44-phonemes-in-english.html. [Accessed 19 06 2019].

[32] "Madarin (Chinese)," [Online]. Available: https://www.mustgo.com/worldlanguages/mandarin/. [Accessed 19 06 2019].

[33] S. Sivasankaran, B. M. L. Srivastava, S. Sitaram, K. Bali and M. Choudhury, "Phone Merging for Code-switched Speech Recognition," [Online]. Available: https://www.microsoft.com/en-us/research/uploads/prod/2018/05/phone-merging-acl_cameraready.pdf. [Accessed 19 06 2019].

[34] "The CMU Pronouncing Dictionary," [Online]. Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict. [Accessed 19 06 2019].

[35] D. Wang, X. Zhang and Z. Zhang, "THCHS-30 : A Free Chinese Speech Corpus," 2015.

[36] D. Jurafsky and J. H. Martin, "N-gram Language Models," in *Speech and Language Processing.*, 2018.

[37] J. Valicek and P. Mizera, "Language models for spontaneous speech recognition," *POSTER 2015,* 14 05 2015.

[38] J. Duchateau, K. Demuynck and P. Wambacq, "Confidence Scoring Based on Backward Langauge Models," in *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 2002.

[39] D. Ramage, "Hidden Markov Models Fundamentals," 1 December 2007. [Online]. Available: http://cs229.stanford.edu/section/cs229-hmm.pdf. [Accessed 10 07 2019].

[40] Y. Bengio, A. Courville and V. P., "Representation Learning: A Review and New Perspectives," *Representation Learning: A Review and New Perspectives,* vol. 35, no. 8, pp. 1798-1828, 2013.

[41] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks,* vol. 61, p. 85–117, 2015.

[42] Y. Bengio, Y. LeCun and G. Hinton, "Deep Learning," *Nature,* vol. 521, p. 436–444, 2015.

[43] C. Szegedy, A. Toshev and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems*, 2013.

[44] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," 27 04 2012. [Online]. Available: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/38131.pdf. [Accessed 21 06 2019].

[45] D. Yu and L. Deng, "Deep Neural Networks," in *Automatic Speech Recognition A Deep Learning Approach*, London, Springer-Verlag London, 2015, pp. 57-76.

[46] Y. Upadhyay, "Introduction to FeedForward Neural Networks," Towards Data Science, 7 March 2019. [Online]. Available: https://towardsdatascience.com/feed-forward-neural-networks-c503faa46620. [Accessed 10 07 2019].

[47] S. Amari, "Theory of adaptive pattern classifiers," *IEEE Trans.* , Vols. EC-16, no. 3, pp. 299-307, 1967.

[48] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016, p. 196.

[49] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, pp. 533-536, 1986.

[50] C.-Y. Liou, J.-C. Huang and W.-C. Yang, "Modeling word perception using the Elman network," *Neurocomputing*, vol. 71, no. 16-18, p. 3150, 2008.

[51] H. Geoffrey, D. Li, Y. Dong, E. D. George, M. Abdel-rahman, J. Navdeep, S. Andrew, V. Vincent, N. Patrick, N. S. Tara and et al., "“Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, pp. 82-97, 2012.

[52] P. Saikia, "HMM-DNN SPEECH RECOGNITION TECHNIQUES: A REVIEW," *International Journal of Development Research*, vol. 07, no. 07, pp. 14068-14072, 2017.

[53] Ye, W. Wang and K. Chen, "Speech-to-text".

[54] S. Ruder, "Transfer Learning - Machine Learning's Next Frontier," Sebastian Ruder, 21 March 2017. [Online]. Available: http://ruder.io/transfer-learning/. [Accessed 10 07 2019].

[55] "Procedural Knowledge," Training Industry, [Online]. Available: https://trainingindustry.com/glossary/procedural-knowledge/. [Accessed 10 07 2019].

[56] T. T. Um, M. S. Park and J.-M. Park, "Independent Joint Learning: A Novel Task-to-Task Transfer Learning Scheme for Robot Models," in *IEEE International Conference on Robotics & Automation (ICRA)*, Hong Kong, China , 2014.

[57] S. Geisser, "An Introduction," in *Predictive Inference: An Introduction*, 1993.

[58] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications*, J. M. R. M. M. M. a. A. E. Soria, Ed., IGI Global, 2009.

[59] D. Wang and T. F. Zheng, "Transfer Learning for Speech and Language Processing," 16 10 2015. [Online]. Available: http://www.cslt.org/mediawiki/images/b/b2/Tlearn.pdf. [Accessed 23 06 2019].

[60] D. Li and Y. Dong, "“Deep learning: Methods and applications," *” Foundations and Trends in Signal Processing*, vol. 7, pp. 197-387, 2013.

[61] X. He, J. Gao and L. Deng, "“Deep learning for natural language processing and related applications (tutorial at icassp)," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[62] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, pp. 261-266, 2015.

[63] D.-c. Lyu, R.-Y. Lyu and Y.-c. Chiang, "Speech Recognition on Code-Switching Among the Chinese Dialects," in *Acoustics, Speech, and Signal Processing, 1988*, 1988.

[64] J. Y. C. Chan, P. C. Ching and T. Lee, "Development of a Cantonese-English Code-mixing Speech Corpus," in *InterSpeech*, Lisbon, Portugal, 2005.

[65] D.-C. Lyu and R.-Y. Lyu, "Language Identification on Code-Switching Utterances Using Multiple Cues," in *2008 ISCA*, Brisbane Australia, 2008.

[66] H. Zhang, "Code-switching Speech Detection Method by Combination of Language and Acoustic Information," in *Proceedings of the 2012 2nd International Conference on Computer and Information Application (ICCIA 2012)*, 2012.

[67] K. K. Bhuvanagiri and S. K. Kopparapu, "An Approach to Mixed Language Automatic Speech Recognition," 13 01 2015. [Online]. Available: https://www.researchgate.net/publication/267408450_An_Approach_to_Mixed_Language_Automatic_Speech_Recognition?enrichId=rgreq-1528b04a65d5ee8a16c0e3da2a178208-XXX&enrichSource=Y292ZXJQYWdlOzI2NzQwODQ1MDtBUzoxODUxMjA0NzY2MzkyMzVAMTQyMTE0NzU2OTY2Mw%3D%3D&el=. [Accessed 23 06 2019].

[68] S. Sivasankaran, B. M. L. Srivastava, S. Sitaram, K. Bali and M. Choudhury, "Phone Merging for Code-switched Speech Recognition," in *Proceedings of The Third Workshop on Computational Approaches to Code-Switching*, Melbourne, Australia, 2018.

[69] E. Yılmaz, H. v. d. Heuvel and D. v. Leeuwen, "Investigating Bilingual Deep Neural Networks for Automatic Recognition of Code-switching Frisian Speech," *5th Workshop on Spoken Language Technology for Under-resourced Languages, SLTU 2016*, pp. 159-166, 9-12 05 2016.

[70] Z. Zeng, H. Xu, T. Y. Chong, E.-S. Chng and H. Li, "Improving N-gram Language Modeling for Code-switching Speech Recognition," in *Proceedings of APSIPA Annual Summit and Conference 2017*, Malaysia, 2017.

[71] A. Biswas, F. d. Wet, E. v. d. Westhuizen, E. Yılmaz and T. Niesler, "Multilingual Neural Network Acoustic Modelling for ASR of Under-Resourced English-isiZulu Code-Switched Speech," in *InterSpeech*, 2018.

[72] H. Adel, N. T. Vu, F. Kraus, T. Schlippe, H. Li and T. Schultz, "RECURRENT NEURAL NETWORK LANGUAGE MODELING FOR CODE SWITCHING CONVERSATIONAL SPEECH," in *IEEE ICASSP 2013*, 2013.

[73] N. T. Vu and T. Schultz, "Exploration of the Impact of Maximum Entropy in Recurrent Neural Network Language Models for Code-Switching Speech," *Proceedings of The First Workshop on Computational Approaches to Code Switching*, p. 34–41, 25 10 2014.

[74] N. Alam, "Unable to get good decoding transcript," Mozilla, 14 05 2019. [Online]. Available: https://discourse.mozilla.org/t/unable-to-get-good-decoding-transcript/40316. [Accessed 25 06 2019].

[75] A. Hamza, "Bilingual Speech Recognition using Deepspeech (English - Urdu)," Mozilla, 27 02 2019. [Online]. Available: https://discourse.mozilla.org/t/bilingual-speech-recognition-using-deepspeech-english-urdu/36296. [Accessed 25 06 2019].

[76] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates and A. Y. Ng, "Hannun, Awni Y., Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates and Andrew Y. Ng. “Deep Speech: Scaling up end-to-end speech recognition.” CoRR abs/1412.5567 (2014): n. pag.," *CoRR,* vol. abs/1412.5567, 2014.

[77] A. Stolcke, "SRILM - an extensible language," *In Proceedings of the Seventh International Conference on Spoken Language Processing,* pp. 901-904, 2002.

[78] M. Federico and N. Bertoldi, "How many bits are needed to store probabilities for phrasebased translation?," *In Proceedings of the Workshop on Statistical Machine Translation,* pp. 94-101, 2006.

[79] M. Federico, N. Bertoldi and M. Cettolo, "IRSTLM: an open source toolkit for handling large scale language models," in *In Proceedings of Interspeech*, Brisbane, Australia, 2008.

[80] K. Heafield, "KenLM: Faster and Smaller Language Model Queries," 2013. [Online]. Available: https://kheafield.com/papers/avenue/kenlm.pdf. [Accessed 25 06 2019].

[81] O. Iakushkin and A. S. A. D. O. S. G.A. Fedoseev, "RUSSIAN-LANGUAGE SPEECH RECOGNITION SYSTEM BASED ON DEEPSPEECH," in *Proceedings of the VIII International Conference "Distributed Computing and Grid-technologies in Science and Education" (GRID 2018)*, Dubna, Moscow region, Russia,, 2018.

[82] R. Morais, "A Journey to <10% Word Error Rate," 29 11 2017. [Online]. Available: https://hacks.mozilla.org/2017/11/a-journey-to-10-word-error-rate/. [Accessed 29 6 2019].

[83] M. Schuster and K. K. Paliwal., "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, 1997.

[84] Z. Chen, W. Deng, T. Xu and K. Yu, "Phone Synchronous Decoding with CTC Lattice," in *INTERSPEECH*, San Francisco, USA, 2016.

[85] D. Wilkins, C. Fagerlin, W. H. Hsu, E. T. Lin and D. Kruse, "Design of a Damage Control Simulator," Knowledge Based Systems Laboratory Technical Report UIUC-BI-KBS-96005, 1996.

[86] "Pyaudio," MIT, [Online]. Available: https://people.csail.mit.edu/hubert/pyaudio/. [Accessed 10 07 2019].

[87] Mosaic, "The amazing benefits of being billingual," BBC, 12 08 2016. [Online]. Available: http://www.bbc.com/future/story/20160811-the-amazing-benefits-of-being-bilingual. [Accessed 10 07 2019].

[88] T. J. Sejnowski and C. R. Rosenberg, "NETtalk: a parallel network that learns to read aloud," Baltimore, 1986.

[89] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coate and A. Y. Ng, "Deep Speech: Scaling up end-to-end speech recognition," 19 12 2014. [Online]. Available: https://arxiv.org/abs/1412.5567#aHR0cHM6Ly9hcnhpdi5vcmcvcGRmLzE0MTIuNTU2N0BAQDA=. [Accessed 25 06 2019].

[90] J. Picone, "Switchboard," August 1997. [Online]. Available: https://www.isip.piconepress.com/projects/switchboard/. [Accessed 10 07 2019].

[91] Canavan, Alexandra, D. Graff and G. Zipperlen, "CALLHOME American English Speech LDC97S42," Philadelphia: Linguistic Data Consortium, , 1997. [Online]. Available: https://catalog.ldc.upenn.edu/LDC97S42. [Accessed 10 07 2019].

[92] A. Weigend, "On overfitting and the effective number of hidden units," in *Proceedings of the 1993 connectionist models summer school*, 1993.

[93] e. a. Srivastava N., "Dropout: A simple way to prevent neural networks from overfitting," in *The Journal of Machine Learning Research 2014*, 2014.

[94] K. Dietrich and J. Peters, "Testing the correlation of word wrror rate and perplexity," *Speech Communication*, vol. 38, no. 1-2, pp. 19-28, 2002.

[95] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707-710, 1966.

[96] L. Gad M and V. Uzi, "Introduction Efficient Paralleism Into Approximate String Matching and a New Serial Algorithm," *Proceeding of the 18th Annual ACM Symposium on Theory of Computing*, pp. 220-230, 1986.

[97] T. H. Cormen, C. E. Leiseron, R. L. Rivest and C. Stein, "Dynamic Programming," in *Introduction to Algorithms*, Cambridge, Massachusetts, McGraw-Hill Book Company, 2002, pp. 323-356.

[98] P. Wang, R. Sun, H. Zhao and K. Yu, "A New Word Language Model Evaluation Metric for Character Based Languages," *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, vol. 8202, pp. 315-317, 2013.

[99] "Does Batch_size in Keras have any efects in results' quality," Stack Exchange, 01 07 2016. [Online]. Available: https://datascience.stackexchange.com/questions/12532/does-batch-size-in-keras-have-any-effects-in-results-quality. [Accessed 10 07 2019].

[100] X. Huang, "A Study on Speaker-Adaptive Speech Recognition," in *Speech and Natural Language: Proceedings of a Workshop*, Pacific Grove, California, 1991.

[101] M. Hassonah, "What is end to end learning in machine learning?," Quora, 22 04 2016. [Online]. Available: https://www.quora.com/What-is-end-to-end-learning-in-machine-learning. [Accessed 08 07 2019].

[102] Y. Bengio, A. Courville and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013.

[103] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview". Neural Networks," vol. 61, pp. 85-117, 2015.

[104] Y. Bengio, Y. LeCun and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[105] J. West, D. Ventura and S. Warnick, "Spring Research Presentation: A Theoretical Foundation for Inductive Transfer," Brigham Young University, College of Physical and Mathematical Sciences, 01 08 2007. [Online]. Available: https://web.archive.org/web/20070801120743/http://cpms.byu.edu/springresearch/abstract-entry?id=861. [Accessed 08 07 2019].

[106] K. Murphy, "A brief Introduction to Graphical Models and Bayesian Networks," 1998. [Online]. Available: https://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html#infer. [Accessed 09 07 2019].

[107] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in neural information processing systems*, 2002.

[108] M. J. F. Gales, "Discriminative models for speech recognition," in *ITA Workshop*, University San Diego, USA, 2007.

[109] "Phone," Merriam-Webster, [Online]. Available: https://www.merriam-webster.com/dictionary/phone. [Accessed 10 07 2019].

[110] "Vector Quantization and clustering," 2003. [Online]. Available: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-345-automatic-speech-recognition-spring-2003/lecture-notes/lecture6new.pdf. [Accessed 10 07 2019].

[111] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salabhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research,* vol. 15, pp. 1929-1958, 2014.

[112] P. M. Joshi, "Generative VS Discriminative Models," Medium, 30 08 2018. [Online]. Available: https://medium.com/@mlengineer/generative-and-discriminative-models-af5637a66a3. [Accessed 10 07 2019].

[113] L. Besacier, V.-B. Le, C. Boitet and V. Berment, "ASR and Translation for Under-Resourced Languages," in *Icassp2006*, 2006.

[114] A. Graves, S. Fernandez, F. Gomez and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proceedings of the 23 rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

[115] "Vocapia Research," Speech Processing Glossary, [Online]. Available: https://www.vocapia.com/glossary.html. [Accessed 10 07 2019].

[116] M. K. Moore, "Research methods in Human Skeletal Biology," in *Sex Estimation and Assessment*, 2013 Elsevier Inc, 2013, pp. 91-116.

[117] D. A. Schum, "Evidence. Probability. and Statistics," in *The Evidential Foundations of Probabilistic Reasoning. Northwestern University Press*, Fairfax, Virginia, John Wiley & Sons Inc, 1994, p. 49.

[118] N. Ide and J. Véronis, "Word Sense Disambiguation: The State of the Art," *Computational Linguistics,* vol. 24, pp. 1-40, 1988.

Appendix A – Survey

Learning to Detect Named Entities in Bilingual Open Speech Corpora

---

Start of Block: Informed Consent

Q1

Consent

Title:

Learning to Detect Named Entities in Bilingual Open Speech Corpora

Principal Investigator:

Professor William Hsu (Primary Investigator and Contact) Professor, Computer Science Kansas State University, United States of America

With collaborators:
Yihong Theis, Graduate Student, Computer Science, Kansas State University

Purpose Statement:
The purpose of this research study is to improve the ability of AI home assistants to recognize two languages used in the same sentence or successive sentences, specifically for names of people, places, and things. This survey is intended for research with the intent of

better understanding how many people speak more than one language to their smart devices.

Study Procedure:

You will be asked to provide responses to several questions about demographics and background and if you willing to join the recordings of your voices. This survey is expected to take 10 minutes to complete. Please note that you can withdraw from this survey at any time.

Confidentiality:

The information that you provide in this experiment will be anonymous. No personally - identifiable information will be collected. Your responses are collected through this secure website and will be stored with the collaborators for up to 5 years.

Funding:

This is an unfunded student-initiated project.

Contact Information:

If you have any questions or concerns about this research project, you may contact Professor William Hsu. If you have any concerns or complaints about your rights as a research participant and/or your experiences while participating in this study, you may contact the Kansas State University Research Compliance Office:

203 Fairchild Hall

Manhattan KS, 66502

7855323224

comply@kstate.edu

Consent:

You consent to take this survey on a voluntary basis and agree with the terms above, by

clicking to proceed.

- ○ **I consent, begin the study (1)**
- ○ **I do not consent, I do not wish to participate (2)**

End of Block: Informed Consent

Start of Block: Demographics and Background

Q1 Please enter your age:

_____

Q2 Gender:

- ○ **Male (1)**
- ○ **Female (2)**
- ○ **Other (3)**

Q3 Do you speak more than one language?

- ○ **Yes (1)**
- ○ **Maybe (2)**
- ○ **No (3)**

Q4 If you speak more than one language, what is your native language:

_____

Q5 If you speak more than one language, what are other languages you speak except your native language:

_____

Q6 Select the highest level of education obtained:

- ○ **Primary (1)**
- ○ **Some High School (2)**
- ○ **High School (3)**
- ○ **Trade / Technical / Vocational Training or Degree (4)**
- ○ **Associate Degree (5)**
- ○ **Bachelor's Degree (6)**
- ○ **Master's / Professional Degree (7)**
- ○ **Doctoral Degree (8)**

End of Block: Demographics and Background

Start of Block: Join the research

Q1 If you speak Mandarin and English, please provide your e-mail below, so we can contact you about collecting recordings(less than 5 minutes for each person) or read the file (which can found here:https://docs.google.com/document/d/1GmLHhOhA-a4JWOZyVUWXlKmTgXdlq9s9rxEEFSen3FQ/edit?usp=sharing) and recording by your phone to email yihong@ksu.edu:

_____

End of Block: Join the research

Start of Block: Comments

Q1 Do you have any other comments regarding your home assistant or the project?

_____

Q2 Thank you for completing the survey, we appreciate your efforts to contribute to the project!

This project is intended for research only. We are hoping to discover new insights about bilingual speech understanding and smart home assistants. We anticipate synthesizing this information for publication in journals and conferences. As a reminder, your data will be stored on the collaborators' computers for up to 5 years, and your data is anonymous.

Contact Information and Questions

If you have any questions or concerns about the survey, please direct them to Professor William Hsu (bhsu@ksu.edu).

If you have any concerns or complaints about your rights as a research participant and/or your experiences while participating in this study, you may contact the Kansas State University Research Compliance Office:

203 Fairchild Hall
Manhattan KS, 66502
7855323224
comply@kstate.edu

Appendix B – Corpora Phrases

1. play
2. search for,
3. find,
4. look up,
5. who is
6. On Spotify,
7. On Pandora,
8. On Kugou,
9. On QQ,
10. on LastFM
11. 武艺
12. 易烊千玺
13. 王俊凯
14. 薛之谦
15. 周杰伦
16. 林俊杰
17. 邓紫棋
18. 刘德华
19. 张学友
20. TFboys
21. 张杰
22. 毛不易
23. 半阳
24. 刀郎
25. 凤凰传奇
26. 吴青峰

27. 陈奕迅
28. 花姐
29. 李荣浩
30. 庄心妍
31. 华晨宇
32. 蔡徐坤
33. 王杰
34. 邓丽君
35. 黄家驹
36. 鹿晗
37. 张宇
38. 郑源
39. 张靓颖
40. 谭咏麟
41. 张国荣
42. 王力宏
43. 李宇春
44. 汪苏泷
45. 张韶涵
46. 鞠婧祎
47. 许巍
48. 张艺兴
49. 张信哲
50. 任贤齐
51. 李宗盛
52. 王菲

53. 汪峰
54. 周华健
55. 伍佰
56. 陈瑞
57. 刘若英
58. 赵雷
59. 蔡依林
60. 陈慧娴
61. 陈小春
62. 杨坤
63. 李健
64. 五月天
65. 韩红
66. 吴亦凡
67. 周传雄
68. 杨宗纬
69. 梁静茹
70. 莫文蔚
71. 大张伟
72. 周柏豪
73. 田馥甄
74. 林忆莲
75. 那英
76. 欢子
77. 林宥嘉
78. 李克勤

79. 萧敬腾
80. 胡彦斌
81. 谢霆锋
82. 赵丽颖
83. 苏打绿
84. 迪丽热巴
85. 朴树
86. 张惠妹
87. 阿杜
88. 杨千嬅
89. 孙燕姿
90. 郁可唯
91. 霍尊
92. 罗大佑
93. 黎明
94. 筷子兄弟
95. 谭维维
96. 宋祖英
97. 杨幂
98. 黄子韬
99. 庞龙
100. 曲婉婷
101. 成龙
102. 郭富城
103. 胡夏
104. 陈慧琳

105. 齐秦
106. 罗志祥
107. 孙楠
108. 容祖儿
109. 陶喆
110. 陈柯宇