Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Statistics and Machine Learning
2020 | LIU-IDA/STAT-A-20/020--SE

Automatic Speech Recognition in Somali

Naveen Joseph Gabriel

Supervisor : Jose M. Peña
Examiner : Oleg Sysoev

External supervisor : Naveen Sasidharan

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

The field of speech recognition during the last decade has left the research stage and found its way into the public market, and today, speech recognition software is ubiquitous around us. An automatic speech recognizer understands human speech and represents it as text. Most of the current speech recognition software employs variants of deep neural networks. Before the deep learning era, the hybrid of hidden Markov model and Gaussian mixture model (HMM-GMM) was a popular statistical model to solve speech recognition.

In this thesis, automatic speech recognition using HMM-GMM was trained on Somali data which consisted of voice recording and its transcription. HMM-GMM is a hybrid system in which the framework is composed of an acoustic model and a language model. The acoustic model represents the time-variant aspect of the speech signal, and the language model determines how probable is the observed sequence of words. This thesis begins with background about speech recognition. Literature survey covers some of the work that has been done in this field. This thesis evaluates how different language models and discounting methods affect the performance of speech recognition systems. Also, log scores were calculated for the top 5 predicted sentences and confidence measures of predicted sentences. The model was trained on 4.5 hrs of voiced data and its corresponding transcription. The method was evaluated on 3 mins of testing data. The performance of the trained model on the test set was good, given that the data was devoid of any background noise and lack of variability. The performance of the model is measured using word error rate (WER) and sentence error rate (SER). The performance of the implemented model is also compared with the results of other research work. This thesis also discusses why log and confidence score of the sentence might not be a good way to measure the performance of the resulting model. It also discusses the shortcoming of the HMM-GMM model, how the existing model can be improved, and different alternatives to solve the problem.

Keywords: automatic speech recognition, speaker adaptation, generative training, gaussian mixture model, kaldi, finite-state transducers.

Acknowledgments

I would like to mention my special gratitude to my supervisor, Jose M. Peña for his constant help and support. I also want to thank my examiner Oleg Sysoev, for assistance along the way. Also, I would like to thank the Computer and Information Science department, Linköping University for imbibing me with knowledge during Statistics and Machine Learning master course program.

I am very grateful to the people behind KALDI toolkit for being very responsive and helpful. Especially Daniel Povey, the author of Kaldi toolkit, who quickly answered any of my questions and helped me overcome problems along the way. I also like to thank Nikolay Shmyrev for clearing out my doubts and help in understanding how speech recognition works. I would also like to thank Astik Biswas from Stellenbosch University, who was quick to reply to my emails whenever I had a doubt in ASR.

I thank Worldish AB for providing me with this opportunity to collaborate with them to do this thesis. My note of thanks to Naveen Sasidharan (Co-founder & COO, Worldish) who is my industrial supervisor for his inputs and patient hearing of all my doubts. Special thanks to my friends and classmates Jasleen, Mathew, Brian, Maria and Sridhar for being a constant source of support. Thanks to my opponent Alexander Karlsson, for having provided me with valuable and constructive remarks.

And lastly, to my family who supported me throughout this project.

Contents

List of Figures

List of Tables

1 Introduction

Natural language processing (NLP) is a branch of AI which helps machines to understand or manipulate human languages. Within the area of NLP, automatic speech recognition (ASR) is an exciting field where the task for a machine is to recognize the words uttered by the speaker automatically. Speech recognition started with the goal of recognizing isolated words. As time progressed, the interest in solving large vocabulary continuous speech recognition (LVSCR) grew further. Currently, ASR features ubiquitously in broad areas such as in vehicles, entertainment industry, military, educating people with disability, healthcare, video games, hands-free computing and many more. Within speech recognition, voice to text is one such application, also called speech to text (STT). In this STT, the task is to automatically recognize the voice and transcribe it into appropriate text in the same language.

1.1 Motivation

In Sweden, many immigrants who visit hospitals cannot speak Swedish, which creates a communication barrier between doctors and patients. Worldish AB is an e-health company based in Sweden that has a product called Helen, which is a digital healthcare communication and translation tool that aims to reduce the language gap. The Helen software helps doctors and patients to communicate with each other when both speak a different language.

With the help of approved medical translators, Worldish has a curated set of medical questions and answers for both doctors and patients. During a patient-doctor interaction, both the parties are given a tablet device with these pre-loaded set of questions and answers. In a typical interaction between doctor and patient speaking Swedish and Somali, respectively, the doctor clicks a particular sentence in Swedish, which is reflected in the patient device in the Somali language. Then the patient chooses an appropriate answer from the list of answers in Somali, which is reflected in the doctor's device in Swedish. Having many categories for different medical scenarios on the device makes the navigation difficult.

To mitigate the above scenario, navigating with voice command is one of the solutions. So when a healthcare staff speaks to Helen, it recognizes what the staff has uttered and searches for the most relevant medically approved text in its database, which allows the staff to pick

an appropriate text. Since the doctors in Sweden speak mostly in English or Swedish, voice recognition is made possible with third-party software. Clinics and hospitals require a feature in which patients could also similarly talk to the software and select the most relevant suggested phrases from the database. Having this feature would allow the patients to freely interact with a doctor or nurse and have a better user experience and satisfaction. Right now on the patient's side, for voice to text conversion, there are less or few relevant sources and support available in the market, for languages like Somali or Tigrinya. These are two of many languages that are spoken by a significant part of the patient community in Sweden.

1.2 Aim

The search problem in LVCSR is to find the most probable sequence of words given a sequence of acoustic observations, an acoustic model and a language model. Acoustic observations are voice uttered. The acoustic model captures the audio features of spoken words. The language model tells how good is the grammar of a sentence by estimating the probability of a sentence.

Software services like Google API, Microsoft services, Dialogue flow and many other offers help in the transcription of audio data in multiple languages. However, almost none of the services support the Somali language, which is often considered small. An ASR for low resource language on medical data should not confuse words like cough with fracture or flu with cold. It would result in misunderstanding, false diagnosis and time loss.

Worldish has medical data in Somali, which is growing. Moreover, there are other low resource languages such as Tigrinya, Dari, whose data is growing within the company. So from the company perspective, it is interesting to see how well an ASR system can perform on the data they have. In this regard, this thesis intends to implement STT for one of the most in demanded patient languages, Somali.

1.3 Research questions

The research question for this thesis would be:

- How well can the speech be recognized from a given data set in the Somali language?
- How well does the implementation in this thesis compare with results from similar research?
- How does language model affect the performance in predicting words for voiced input?
- How well the trained model performs on an unseen dataset?

1.4 Outline

Organization of this thesis is as follows. Section 2 discusses the literature survey, which gives an abstract idea of previous work done and current methodologies used in solving speech recognition. It also discusses the previous work done on the Somali language. The speech data, its properties and preprocessing are given in Section 3. This section also discusses lexicon models. Section 4 describes the fundamental aspect of a speech recognition system which includes signal feature extraction, modelling of acoustic and linguistic knowledge. This section also discusses how training and decoding of speech signals are done. Also, it discusses

weighted finite transducers and word lattice, which is helpful in the decoding process. Section 5 gives a result of the methodologies used, and section 6 discusses those results. Finally, this thesis concludes by answering the research question in section 7.

1.5 Ethical Consideration

The data that is being used in the thesis is Somali audio and its corresponding transcription. This data has been prepared by the medical translators at Worldish AB. The data does not have any personal information or private user data.

2 Literature Review

One of the most popular paradigms to perform ASR is hidden Markov model-Gaussian mixture model (HMM-GMM) architecture [8] which has been used widely for decades. The speech feature distribution was represented using GMM, which is observations for the state of HMM. These feature vectors can be assumed to come from a 39-dimensional Gaussian distribution, but that would be too restrictive. For example, speaker, accent and gender differences tend to create multiple modes in the data. To address this mixture of Gaussian distributions are used to represent state output distributions. In reality, the features of voice samples are not a mixture of Gaussian distribution but it was an assumption for the acoustic model so that the model becomes solvable by restricting the covariance of Gaussian distribution to contain only diagonal elements[13]. In general, the use of full-covariance Gaussian distribution in large-vocabulary systems would be impractical due to the sheer size of the model set, although given enough data it can be done. Even with small training data, a full covariance matrix is avoided [8]. If d is the dimensionality of feature vector then $\mathcal{O}(d^2)$ is the computational cost of using full covariance matrices compared to $\mathcal{O}(d)$ for the diagonal case [8, 13]. During training, the parameters of the state like mean and variance for GMM is calculated [13].

With more availability of data, representation of acoustic features by GMM was replaced by a deep neural network (DNN) [20]. DNN is a neural network where many layers are stacked together between input and output. Figure 2.1 represents an architecture of HMM-DNN acoustic model. Here, the acoustic vectors of audio were given as an input to DNN. Each state of HMM was mapped to its observation which was the output of DNN. The output of DNN was not a single output. However, like GMM, it represented the features of acoustic data corresponding to each state. In this, GMM assumptions for the features were dropped, and DNN replaced GMM because DNN could generalize much better with a small number of parameters over complex distribution. This model performed better than HMM-GMM model over TIMIT dataset [20].

Above approaches required modelling and finding parameters of different components such as language model, acoustic model and pronunciation dictionary. Even though HMM-DNN model was better than HMM-GMM but the improvement in features extracted by DNN instead by GMM led to an only slight improvement in speech recognition with respect

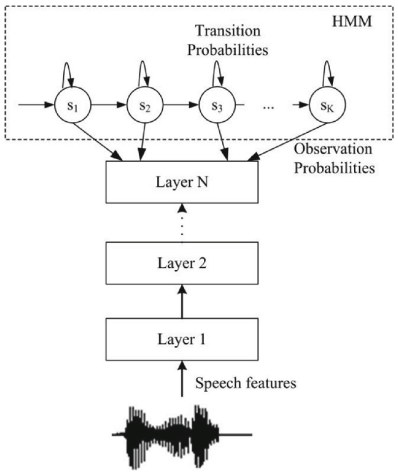


Figure 2.1: Acoustic Model including HMM-DNN [31]. Here s_i represents states of HMM and the observations , which is an acoustic input, is modelled as DNN instead of GMM. Last layer of DNN represents the features of acoustic data which are mapped to HMM states.

to the amount of training data. In research [11], an end to end recurrent neural network (RNN) system was implemented where a single RNN architecture replaced much of the speech pipeline. The network was trained directly on the text transcripts, so no phonetic representation was used. The model was known as the sequence to sequence (Seq2Seq) model. Another variant of the Seq2Seq model was the attention-based model [3] in which for each predicted character, the attention mechanism scans the input sequence and generates appropriate characters. So to predict a particular character, the nearby characters were considered. This model yielded better recognition accuracies.

DNN models require lots of data and good computing power so that the network can learn parameters by itself. With less number of data, the model cannot learn the parameters well enough. Another paradigm called transfer learning is on the rise. Transfer learning is a method where a model implemented for a task is reused as a starting point for another task. It is specifically used when the computing power is low, and data is not in abundance. In paper [16], the researcher adopted a wav2letter convolution neural network (CNN) model trained on English ASR to do German ASR. The parameters of the pre-trained model were frozen for all layers except the last few layers. When this model was applied on German corpus, there was a considerably huge improvement in terms of computing time required for achieving the same loss than without a pre-trained model.

Voiced data which is in analog form cannot be used directly for ASR. A sound is transformed into a set of feature vectors which is then used in the ASR system. This process is called a feature extraction process. Although there are many feature extraction methods, two methods, namely MFCC and perceptual linear prediction (PLP), are widely used in the speech recognition community. According to paper [22], PLP tends to perform slightly better than MFCC in a speaker-independent continuous speech recognition task. However, with a slight adjustment of a parameter, both give nearly the same performance.

There are several toolkits available to do speech recognition, and one of the popular toolkits is Kaldi. Kaldi toolkit [25] provides an ASR pipeline to have better control over the model than the modern state of the art tools like TensorFlow, HTK toolkit etc. Word error rate (WER) is a common metric to measure the performance of an ASR system. WER is similar to Levenshtein distance which is computed at the word level to give a score. A Levenshtein

distance between two strings is the minimum number word edits like insertion, substitution or deletion that is required to change one sentence to another.

Using Kaldi, ASR for Swedish was developed [15] with GMM-HMM model. Four hundred hours of training data were used, and 100 hrs was used for testing. The lowest WER was 19% using the tri4a GMM-HMM model[15]. tri4a model is a feature transform model for MFCC, which includes a combination of linear discriminant analysis (LDA), maximum likelihood linear transform (MLLT), speaker adaptive training(SAT). With this transform, the original feature space of MFCC is projected to a space with the lower dimension using LDA [17]. It has been shown that LDA transform works best when a diagonalizing MLLT has applied [10] afterwards.

In the recent past, research has focused on languages whose data can be easily attained. Somali is one such low resource language. There are not many publicly available corpus for the Somali language. Corpus were either too costly or were made by researchers who did not make the corpus publicly available. Moreover, research in the Somali language on ASR was not abundantly available, and two accounts of relevant research were found. In paper [1], the researchers used a mapping system to train Somali acoustic with French acoustic data. Though they got the result using a different framework, the paper does not mention which model was used so a relevant comparison with this thesis can not be made. In [19], researchers used 1.57 hrs of annotated speech for acoustic model training and 10 mins for testing. The paper [19] compared the performance of several different models like RNN, subspace GMM (SGMM), bidirectional long short-term Memory (BLSMT). The lowest WER they got was 55% using a combination of convolution neural network, time-delay neural network and BLSTM as an acoustic model. Using GMM-HMM, the WER was 63%.

3 Data

3.1 Speech Data

If the data is abundant, then ASR can have a good performance which is the main problem with Afro countries. For this thesis, Worldish allowed a sample of around 5 hours of audio data and 5000 phrases for the corresponding audio transcription. The annotated text is written in Latin form, which is also the official writing script of the Federal Republic of Somalia. Each sample was recorded in a noise-free environment and transcribed accurately by the Worldish medical translator. A single female medical translator recorded all the voiced samples, so there was no variability regarding voice in the data. Also, it means that a word is pronounced in only one accent. Audio data consisted of a mono channel, which means that the sound intended to be heard as if it was coming from one position. All the sentences were spoken in a medical context like "How are you feeling today", "I have bad throat" etc. in Somali.

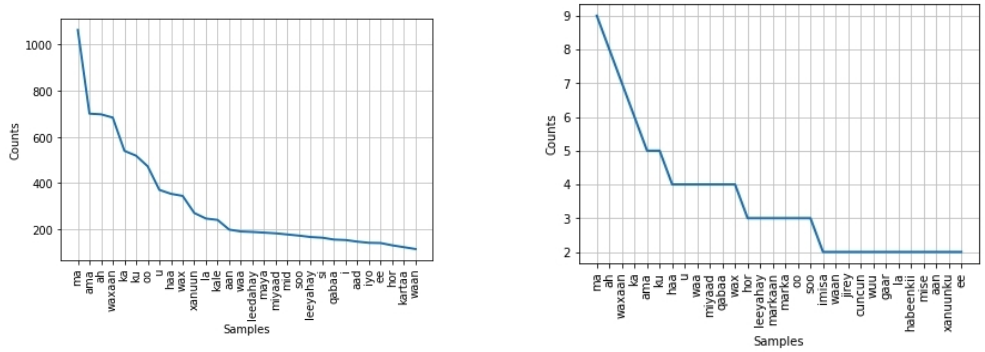
3.2 Preprocessing

The text transcription was provided in a json file which included information about other languages as well. The necessary transcription of text for Somali was pulled from the json file. Also, many of the Somali audio files were encoded in video format. So necessary processing was done to convert the files into the audio format (.wav files) so that Kaldi could process the files.

Some of the text transcriptions did not match correctly with the audio file, so preprocessing was required to align text correctly to the audio. For example, from the transcript, punctuation was removed because the audio cannot capture the comma. Also, there were words which were separated by '/' in the text. However, in the corresponding audio file, 'ama' (which means OR in Somali) pronunciation was used. So those instances of '/' were replaced with the word 'ama' in the text file. Also, digits for numbers were used in the text instead of the corresponding worded transcription. So those files had to be removed. Otherwise, there would be alignment problems.

After preprocessing, the data was divided into train and test data. For speech recognition, the number of training samples usually contain more than 100 hours of audio data [14, 15]. Since the total data provided by the company was around 5 hrs, 95% of the data was used for training and 5% was used for testing.

One of the ways to check similarity or dissimilarity between train and test data is to check the frequency of words occurring in both train and test data set. So, the top 30 frequently occurring words in training and test set were found out, which is represented in figure 3.1.



(a) Training Set

(b) Test set

Figure 3.1: Frequency of top 30 words in training set and test set.

Similarly, 3.2 shows the frequency of the top 16 pairs of words in both train and test set.

Words	Frequency
haa waxaan	138
waxaan qabaa	135
ka hor	124
waxaan leeyahay	108
mid ah	97
kale oo	94
ka mid	85
ma leedahay	81
ah ma	74
maya ma	71
la xidhiidha	59
ku dhaca	59
ma ku	57
halkan lagu	57
ah oo	54

(a) Training Set

Words	Frequency
waxaan qabaa	4
ka hor	3
hor haa	2
haa waan	2
waxaan leeyahay	2
gaar ah	2
marka aan	2
ku dhaca	2
imisa saacadood	1
saacadood ka	1
waan xidhnahay	1
xidhnahay hadda	1
hadda daacuun	1
daacuun caloole	1
caloole waxaan	1

(b) Test set

Figure 3.2: Frequency of top 16 bi-gram words in training set and test set.

In the real world, the spoken voice varies from person to person. So, to test the generalization ability of the model, 5 audio files and its transcription was scraped from the internet[1]. More training data was chosen so that the trained model can learn better from more data so that its generalization capability can be tested on the scraped audio data. These audio files were chosen to test the efficacy of the trained model. For a valid comparison, the scraped audio file was spoken by a female, and it contained medical terms. The scraped audio data was not part of the data provided by the company. Table 3.1 shows the information of the resulting training and test data set.

[1] https://www.cram.com/flashcards/medical-somali-vocabulary-dhakhtarka-erayada-1927696

Table 3.1: Train and test data

Purpose	Recordings	Number of files	Size
Training data	4.5 hrs	4853	1.5GB
Test data 1	3 mins	53	17MB
Test data 2	30 sec	5	5MB

3.3 Lexicon Model

Lexicon model or pronunciation dictionary is a file which maps words to its corresponding phones. Combination of phones makes up a word. For example, the word pin consists of three phones $[p^h in]$. Figure 3.3 is a snippet of a file which represents how each word in Somali can be represented by its corresponding phones. For this thesis, the author did not know the phonetics of Somali language so the lexicon file was created by a rule which would map a word to a phone. If the pronunciations are different, then the same words should be mapped with different phonetics in different lines. Also, since the data was devoid of any background noise or special noise, the lexicon file did not have special symbols for those.

```
waxaan w æ ħ æː n
cunaa ʕ u n æː
cuntooyinka ʕ u n t ɔː j i n k æ
aan æː n
laga l æ ɡ æ
soo s ɔː
qaadin q æː d i n
xoolaha ħ ɔː l æ h æ
sida s i d æ
subaga s u b æ ɡ æ
caanaha ʕ æː n æ h æ
ama æ m æ
hilibka h i l i b k æ
biyuhu b i j u h u
midabkee m i d æ b k eː
ayey æ j e j
lahaayeen l æ h æː j eː n
ilmaha i l m æ h æ
```

Figure 3.3: Lexicon Dictionary file

3.4 Text Corpus

A corpus or text corpus is a language resource consisting of a large and structured set of texts. For this thesis, the text corpus was made by combining all the Somali transcriptions into a single file such that every utterance was placed in a new line. Only training data was used to make this corpus.

4 Method

This section briefs about the basic principle behind Automatic Speech Recognition(ASR). Further subsections will delve more into ASR, which includes feature extraction process, acoustic model, language model, training and decoding process.

4.1 Automatic Speech Recognition

The purpose of ASR is to find the most likely sequence of words given the voice or acoustic data. This process is called decoding. The input audio is converted into fixed-size acoustic vectors $Y_{1:T} = Y_1, Y_2...Y_T$ in a process called feature extraction. The decoder then attempts to find the sequence of words $w_{1:L} = w_1, w_2...w_L$ which would have likely generated the Y, which can be expressed as

$$w^* = \underset{w}{argmax}\ p(w_{1:L}|Y_{1:T} = Y_1, Y_2...Y_T) \tag{4.1}$$

$$w^* = \underset{w}{argmax}\ p(Y_{1:T}|w_{1:L})p(w_{1:L}) \tag{4.2}$$

where w^* is the predicted sequence of words, and the above equation tries to maximize over all possible sequences of w. Although the equation 4.1 would yield the correct result, direct computation is cumbersome. So, Bayes rule is used to transform equation 4.1 to 4.2. $p(Y)$ is not required in the denominator of equation 4.2 as it will be a common value for each numerator. $P(Y|w)$ represent the likelihood of observing the vector sequence Y given some specified word sequence w, and an acoustic model represents this likelihood. P(w) is a priori probability of observing w independent words of the observed signal, and a language model determines this probability.

A hidden Markov model (HMM) represents probability distributions over sequences of observations. HMM assumes that the observation at time t was generated by some process whose state is hidden from the observer. There are two critical assumptions in a hidden Markov model:

- States are conditionally independent of all other states given the previous state.

- Observations are conditionally independent of all other observations given the state that generated it.

In practice, the observation sequence or voiced data is the observable and underlying states or word sequence are hidden. Above assumptions were the crux of the HMM-GMM model where the states of HMM were represented by phones and corresponding acoustic vectors or observation was represented as GMM. Via the feature extraction process, as described in section 4.1, a frame of audio is identified and represented in such a way that it represents a phone. The frame of audio is often represented as a 39-dimensional vector called as Mel frequency cepstral coefficient (MFCC).

Solving the equation 4.2 involves tackling many subproblems. Firstly, a process is required, which can extract all necessary acoustic information in a compact form from the speech waveform with the HMM based acoustic models. The acoustic model maps each phone to its corresponding frame of audio, as shown in figure 4.1. The parameters of these phones are estimated from training data consisting of features and their orthographic transcription. The words are formed by concatenating the phones defined by the pronunciation dictionary or lexicon model. The acoustic model maps each phone to its corresponding frame of audio, as shown in figure 4.1. The parameters of these phones are estimated from training data consisting of features and their orthographic transcription.

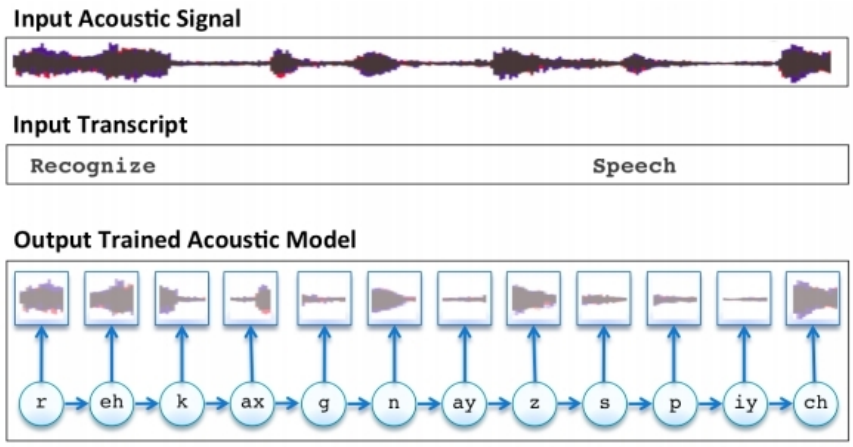


Figure 4.1: This figure shows a trained acoustic model for the audio input "Recognize Speech". Here a frame of audio is identified which represents a phone for the audio input [5].

Secondly, the HMM models should represent the distribution of each sound. Thirdly, the language model should predict the probability of the word given preceding words. It is often computed using n-gram models which is a type of probabilistic language model. The goal of the n-gram model is to compute the probability of a sentence or sequence of words. The n-gram parameters are often computed from a training text corpus. However, the language model must be able to deal with word sequences for which no examples occur in the training data. Finally, the process outlined above for finding w by considering all possible word sequences is difficult. Instead, potential word sequences are found out in parallel; discarding other words sequences in the process which seems unlikely. The process of finding w is called decoding.

One of the assumptions of HMM based systems is that the speech signal is considered stationary (i.e. the spectral characteristics are relatively constant) for a few milliseconds. So,

as a preliminary step, the audio is sampled and then it is divided into 25ms-30ms frames. From these frames, a set of features are extracted which act as an observation to each state. The acoustic model is trained on the data to find out which phone represents an acoustic symbol (parameter learning). After training, a new acoustic feature is given to the model to find the word sequence. Figure 4.2 illustrates the components of the resulting ASR.

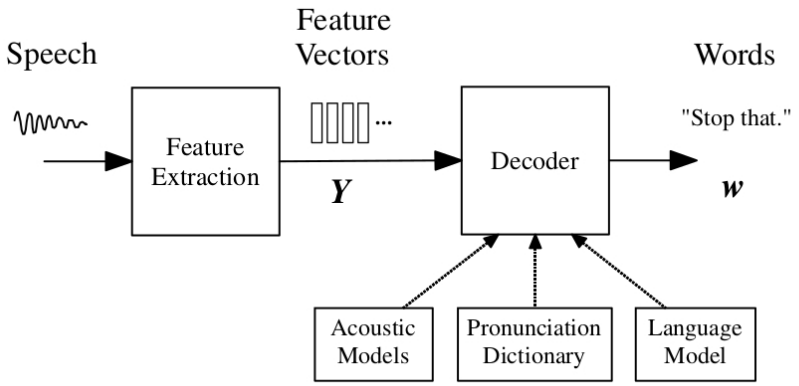


Figure 4.2: HMM based ASR architecture. As shown, from the speech, feature vectors are found. The decoder which consists of an acoustic model,pronunciation dictionary and language model then predicts the voiced input [8].

Feature Extraction

The goal of this step is to find a segment of audio which can represent a phone. Analog signals are continuous wave signals that change with the time period. Audio, which is an analog signal, needs to be transformed in such a way that a set of features can represent it. These set of features are called as MFCC [7] which can be considered as a set of vectors, and it mainly represents a compact form of the audio.

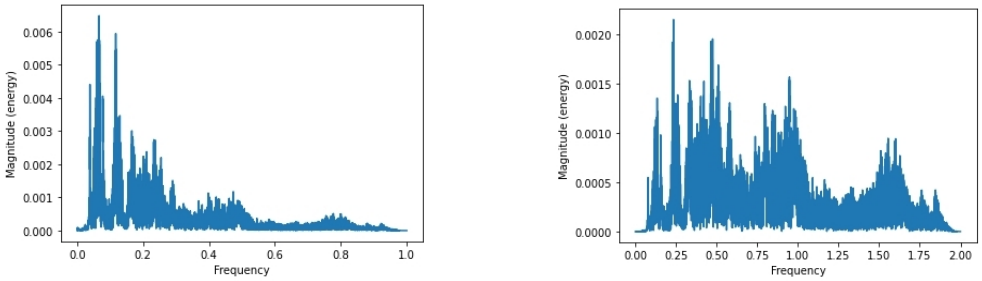
In speech recognition, sampling audio at 8Khz is considered optimal. However, in most of the ASR, the speech is sampled at 16Khz to obtain enough sample points without loss of information [12]. A sampling at 8Khz means that for a one second of analog signal 8000 amplitudes are selected, and this contains enough information related to speech. A sampling rate of more than 16Khz is considered unnecessary and does not improve performance. Moreover, it consumes more memory. Each amplitude of the sampled waveform is represented by a number which can be coded in 16 bit or 8 bit. This process is called quantization.

Pre-emphasis marks the first stage of MFCC feature extraction, which emphasizes the higher frequency. Typically, the amplitude or energy of a signal at high frequencies is lower than the low frequency. So in this step, the amplitude of a high-frequency signal is increased so that the information contained at high frequency is not lost. Each value in the signal is re-evaluated by :

$$s[n] = u[n] - \alpha.u[n-1] \tag{4.3}$$

where u[n] is the sampled signal at time n, u[n-1] is the signal at time n-1, s[n] denotes the resulting signal at time n, and α is the pre-emphasis coefficient which should be in the range between 0.9 and 1 [12]. Signals of low frequency sampled at a high enough rate tend to yield adjacent samples of similar numerical value. The reason is that low frequency essentially means slow variation in time and so the numerical values of a low-frequency signal tend

to change slowly or smoothly from sample to sample. By the subtraction, the part of the samples that did not change in relation to its adjacent samples is removed and the part of the signal that changes rapidly, i.e. its high-frequency components remains the same. Figure 4.3 shows a pre-emphasis step of one of the Somali audio data.



(a) Before pre-emphasis

(b) After pre-emphasis

Figure 4.3: Pre-emphasis of a signal in frequency domain.

Since our goal is to find MFCC for each phone, the entire signal is divided in a set of overlapping windows which is typically 25ms-30ms length that characterizes a phone. The offset between adjacent windows is kept around 10ms [12]. The speech extracted from each window is called a frame. Also, the shape of the window plays a crucial role in the feature extraction process. A rectangular window frame cuts off the signal abruptly at its boundaries. These discontinuities in windows create problems during Fourier analysis. For this reason, a more typical window used in MFCC extraction is the Hamming window, which shrinks the value of the signal towards zero at the window boundaries, avoiding discontinuities [12]. A Hamming window has the following form:

$$w[n] = \begin{cases} 0.54 - 0.46cos\left(\frac{2\pi n}{L-1}\right) & 0 \leqslant n \leqslant L-1 \\ 0 & otherwise \end{cases} \tag{4.4}$$

where, L is the window length. To extract the signal, we multiply the value of the signal at time n, s[n] by the value of the window at time n,w[n]:

$$x[n] = w[n]s[n] \tag{4.5}$$

Any physical signal can be decomposed into a number of discrete frequencies, or a spectrum of frequencies over a continuous range. The next step is to extract spectral information from the windowed signal, so it is essential to know how much energy or amplitude the signal contains at different frequency levels. Extracting the spectral information from each window is done by a discrete Fourier transform (DFT). The discretized speech signal is denoted $x_i(n)$, where i ranges over the number of frames and n ranges over the number of samples (n = 1,2,...,N). The input to the DFT is a windowed signal, and the output for each N discrete frequency bands is a number X[k] representing the energy or amplitude of that frequency component in the original signal. For each particular frequency component, k, of a window, its energy, X[k], is computed by DFT as:

$$X_i[k] = \sum_{n=0}^{N-1} x_i[n]e^{-j*\frac{2\pi}{N}kn} \qquad 0 \leqslant k < N \tag{4.6}$$

where j is the imaginary unit of Euler identity which is given by:

$$e^{jx} = cos(x) + jsin(x) \tag{4.7}$$

The equation 4.6 gives the amount of energy at each frequency band. The periodogram for the speech signal estimate the mean square spectrum of the input and is calculated by taking the modulus square of the DFT.

$$P_i[k] = \frac{1}{N} |X_i[k]|^2$$

At lower frequency, humans can discern the sound and this ability to differentiate decreases at a higher frequency. The Mel-scale aims to mimic the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. This mapping of Hertz to Mel scale is accomplished by the use of a triangular Mel spaced filterbank. Figure 4.4 shows how the energy extracted using DFT is mapped on a mel scale using a triangular filter bank.

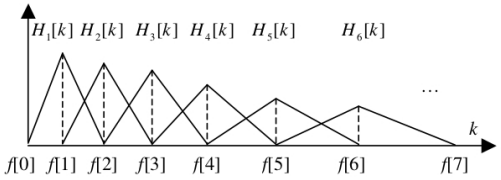


Figure 4.4: Triangular filter bank. Here each mel frequency (e.g.:$H_1[k]$) holds a weighted sum representing the spectral magnitude in that filterbank channel [12].

The first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. The human ear captures differences and changes on lower frequencies better than on higher. Therefore in Mel-scale equally placed points in Hertz-scale will be denser on the lower frequencies than the higher.

An upper frequency f_h and a lower frequency f_l is chosen in Hz. Between the upper and lower frequency, boundaries are M filters (m = 1,2,..., M) uniformly placed in the Mel-scale:

$$f[m] = \left(\frac{N}{F_s}\right) B^{-1}\left(B(f_l) + m\frac{B(f_h) - B(f_l)}{M+1}\right)$$

where F_s is the sampling frequency in Hz, B(f) is the transform from Hertz scale to Mel-scale and B^{-1}(b) is the inverse given by:

$$B(f) = 1127 log_{10}\left(1 + \frac{f}{700}\right)$$
$$B^{-1}(b) = 700\left(exp(\frac{b}{1125}) - 1\right)$$

where filter m is a triangular filter which is calculated as:

$$H_m[k] = \begin{cases} 0 & k < f[m-1] \\ \frac{k - f[m-1]}{f[m] - f[m-1]} & f[m-1] \leqslant k < f[m]) \\ 1 & k = f[m] \\ \frac{f[m+1] - k}{f[m+1] - f[m]} & \\ 0 & k > f[m+1] \end{cases} \tag{4.8}$$

where f[] is the list of M+2 Mel-spaced frequencies. Such filters compute the average spectrum around each center frequency with increasing bandwidths as displayed in figure

4.4.

Filter bank coefficients computed in the previous step are highly correlated. Therefore, Discrete Cosine Transform (DCT) is applied to decorrelate the filter bank coefficients and yield a compressed representation of the filter banks. For the purpose of MFCC extraction, the first 13 values are considered. The higher coefficients are discarded as it has been established that it degrades or does not improve ASR performance [12, p. 423]. The two final steps of computing the MFCCs include computing the log and calculating the Discrete Cosine Transform (DCT) for each of the M filters.

$$S[m] = \log\left(\sum_{k=0}^{N-1} NP_i(k)H_m[k]\right) \qquad 0 \leqslant m < M \tag{4.9}$$

$$z_t[n] = \sum_{m=0}^{M-1} S[m]cos(\pi n(m+\frac{1}{2})/M) \qquad 0 \leqslant n < M \tag{4.10}$$

where $z_t[n]$ contains 13 coefficient for each frame t. For speech recognition, typically only the first 13 (M=13) cepstrum coefficients are used [12].

The cepstral coefficients are usually referred to as static features since they only contain information from a given frame. The extra information about the temporal dynamics of the signal is obtained by computing the first and second derivatives of cepstral coefficients. The first-order derivative is called delta coefficients, and the second-order derivative is called delta-delta coefficients. Delta coefficients tell about the speech rate, and delta-delta coefficients provide information similar to the acceleration of speech.

For an acoustic feature vector y the first order deltas are defined as:

$$\Delta y_t = \frac{\sum_{n=1}^{N} n(z_{t+n} - z_{t-n})}{2\sum_{n=1}^{N} n^2} \tag{4.11}$$

where Δy_t is a delta coefficient of frame t computed in terms of the static coefficients z_{t+n} to z_{t-n}. The value of n is typically set to 2. The delta–delta coefficients ($\Delta^2 y_t$) are computed by taking the first order derivative of the delta coefficients.The combined feature vector for a frame t becomes:

$$y_t = [z_t \ \Delta y_t \ \Delta^2 y_t] \tag{4.12}$$

Figure 4.5 gives an overview of feature extraction process.

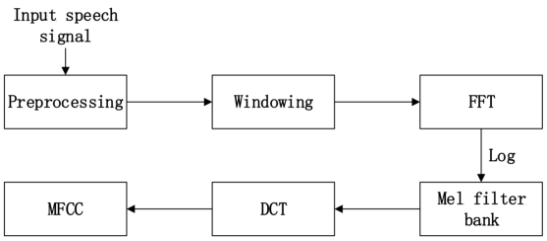


Figure 4.5: Feature Extraction process.

Language Model

The goal of the language model (LM) is to find the probability of a sentence or sequence of words. The prior probability P(w) for a sequence of words w_1, w_2...w_n is given by the chain

rule of probability :

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

The conditional distribution always requires a history of preceding words. Sometimes, the probability of a word conditioned on a long sequence of preceding words will not exist in the corpus. As a result, the entire probability of a sentence might become zero. To tackle the problem, one of the famous LM used is the n-gram model. The n-gram determines the probability of the nth word given n-1 words. So limiting the condition to n-1 values gives a better approximation.

So a bigram model approximates the probability of a word conditioned on one preceding word. Applying a bi-gram to the above equation, it transforms to :

$$P(w) = P(w_1).P(w_2|w_1).P(w_3|w_2).P(w_4|w_3)...P(w_n|w_{n-1})$$
$$P(w) = \prod_{k=1}^{n} P(w_k|w_{k-1})$$

In general, the probability of a sentence for an n-gram model is given by :

$$P(w) = \prod_{k=1}^{n} P(w_k|w_{k-1}...w_{k-n+1})$$

where k is each word in the sentence.

The probability for each of the n-gram is computed using MLE. For a bigram, the estimate can be given by:

$$P(w_k|w_{k-1}) = \frac{C(w_{k-1}, w_k)}{C(w_{k-1})}$$

where $C(w_k, w_{k-1})$ is the count of a particular 2 words in a corpus and $C(w_{k-1})$ is the count of the penultimate word in the corpus. Extending this to n-gram, the estimate for each n-gram is given by :

$$P(w_k|w_{k-N+1}^{k-1}) = \frac{C(w_{k-n+1}^{k-1}, w_k)}{C(w_{k-n+1}^{k-1})}$$

The words $C(w_{k-n+1}^{k-1})$ preceding the current word w_k are sometimes called history. If n is 3, then this model is referred to as a trigram model. The n-gram model captures the grammar of language but not the meaning. If the corpus is terrible, then the language model will be equivalently wrong. Moreover, the model might assign a zero probability if a particular word is never seen in the corpus. Data sparsity is one of the problems of language models [6]. Not seeing a word sequence in training data does not necessarily mean that the word sequence cannot occur in test set.

Smoothing is a technique in the language model to not to assign zero probability to unseen words. It is done by dropping some probability mass from the seen words and giving it to the unseen words [6]. Discounting is a similar method where the intuition is to use the count of things seen once to help estimate the count of things never seen. Two of the many discounting algorithms are Witten-Bell discounting and Kneser-Ney smoothing. Witten-Bell smoothing is defined as :

$$P_{WB}(w_k|w_{k-n+1}^{k-1}) = \lambda_{w_{k-n+1}^{k-1}} p_{ML}(w_k|w_{k-n+1}^{k-1}) + (1 - \lambda_{w_{k-n+1}^{k-1}}) P_{WB}(w_k|w_{k-n+2}^{k-1}) \quad (4.13)$$

where p_{WB} denotes the Witten-Bell probability and p_{ML} denotes the maximum likelihood. From the equation, it follows that if w_k is seen, then higher model n-gram should be used otherwise backoff to lower order model. One can interpret $(1 - \lambda_{w_{k-n+1}^{k-1}})$ as the probability of going back to lower-order model and $(\lambda_{w_{k-n+1}^{k-1}})$ as probability of using higher-order model. In particular, the nth-order smoothed model is defined recursively as a linear interpolation between the nth-order maximum likelihood model and the (n-1)th-order smoothed model [6].

Kneser-Ney is based on absolute-discounting interpolation, which makes use of both higher-order and lower-order language models, i.e. it reallocates some probability mass from 4-grams or 3-grams to simpler unigram models. Kneser-Ney smoothing is defined as:

$$P_{KN}(w_k|w_{k-n+1}^{k-1}) = \frac{\max\{c_{KN}(w_{k-n+1}^{k}) - D, 0\}}{\sum_{w_k} c_{KN}(w_{k-n+1}^{k-1})} + \lambda(w_{k-n+1}^{k-1}) P_{KN}(w_k|w_{k-n+2}^{k-1}) \tag{4.14}$$

where λ is a normalizing constant, which is the probability mass discounted for higher-order and D is the discounted weight which tells how much weight should be subtracted from seen n-grams. c_{KN} is defined as:

$$c_{KN} = \begin{cases} \text{count}(\bullet) \text{ for the highest order} \\ \text{continuationcount}(\bullet) \text{ for the lower order} \end{cases} \tag{4.15}$$

where continuationcount(•) is the number of unique sequences of words for the lower order n-gram [6]. The first term on the right-hand side of the equation 4.14 calculates the number of times the word appears after any other word divided by the number of distinct pairs of consecutive words in the corpus for the highest order n-gram. The lower order of n-gram becomes a significant factor in the combined model only when few or no counts are present in the higher order distribution.

Both Witten-Bell discounting [6] and Kneser-Ney smoothing [6] follows interpolation technique. The interpolated technique combines the probability of a sentence with that of its lower order, e.g. combined probability of trigram, bigram, and unigram for trigram language model.

Acoustic Model

The task of the acoustic model is to find the likelihood $P(Y|w)$, which is to find the probability of an observation given a phone. As briefly mentioned above, words are broken down into K_w basic sounds called base phones using lexicon dictionaries. This sequence is called as pronunciation $q_{1:K_w}^{(w)} = q_1, ..., q_{K_w}$. To allow for the possibility of multiple pronunciations, the likelihood p(Y | w) can be computed over multiple pronunciations :

$$p(Y|w) = \sum_{Q} p(Y|Q)P(Q|w) \tag{4.16}$$

where the summation is over all valid pronunciation sequences for w, Q is a particular sequence of pronunciations,

$$P(Q|w) = \prod_{l=1}^{L} P(q^{(w_l)}|w_l) \tag{4.17}$$

and where each $q^{(w_l)}$ is a valid pronunciation for the word w_l. These phones represent the chained states of HMM and the observations of HMM states are a vector of 39 features which are assumed to be GMM as illustrated in figure 4.6. In HMM, a state s_i makes a transition

to another state s_j with transition probability a_{ij} and generates a feature vector using the distribution associated with that state with probability b_j.

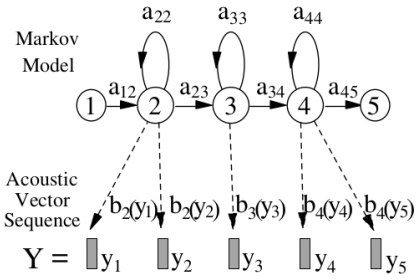


Figure 4.6: This figure represents a HMM based phone model. For example ① represents the state of HMM which can make transition to another state with probability a_{ij} to generate observation y_j with probability $b_j(y_t)$. Each of the states and its corresponding observation represents the phone and its acoustic feature respectively. Here ① and ⑤ are non emitting states [8].

Given the composite HMM Q formed by concatenating all of the constituent base phones $q^{(w_1)}, ..., q^{(w_L)}$ then the acoustic likelihood is given by:

$$p(Y|Q) = \sum_{\theta} p(\theta, Y \mid Q) \tag{4.18}$$

where $\theta = \theta_0, ..., \theta_{T+1}$ is a state sequence through the composite model and

$$p(\theta, Y|Q) = a_{\theta_0\theta_1} \prod_{t=1}^{T} b_{\theta_t}(y_t) a_{\theta_t\theta_{t+1}} \tag{4.19}$$

where θ_0 and θ_{T+1} correspond to the non-emitting entry and exit states shown in Figure 4.6. The parameters of the state $\lambda = [a_{ij}, b_j()]$ are learned during training which will be discussed in the next section 4.1.

Equation 4.20 shows the equation for the GMM function; the resulting function is the sum of M number of Gaussian. Equation 4.21 represents the observation probability ($b_j(y_t)$) of seeing y_t and subscript j denotes the state from which the observation is assumed to come from, μ_{jm} is the mean of the state j for a particular mixture component m and likewise for the covariance matrix.

$$f(x \mid \mu, \Sigma) = \sum_{k=1}^{M} c_m \frac{1}{\sqrt{2\pi|\Sigma_m|}} . \exp\,(x - \mu_m)^T \Sigma^{-1} (x - \mu_m) \tag{4.20}$$

$$b_j(y_t) = \sum_{m=1}^{M} c_{jm} \frac{1}{\sqrt{2\pi|\Sigma_{jm}|}} . \exp\,(y_t - \mu_{jm})^T \Sigma_{jm}^{-1} (y_t - \mu_{jm}) \tag{4.21}$$

Where M is the number of mixtures and the following stochastic constraints for the mixture weights, c_{jm}, holds:

$$\sum_{k=1}^{M} c_{jm} = 1 \qquad j = 1, 2, ..., N$$

$$c_{jm} \geqslant 0 \qquad k = 1, 2, ..., M$$

The most common configuration to represent states is to use three HMM states: a beginning, middle and an end state. Each phone thus consists of three emitting HMM states instead of one. Consider the phone "/a/" is uttered after the phone "/i/". For phone "/a/",the three states capture the initial transition from "/i/" to "/a/", a steady-state where "/a/" is uttered and a transition out state from "/a/". These three states capture different amplitude and frequency of a phone from start to end [13]. The use of self-loops allows a single phone to repeat so as to cover a variable amount of the acoustic input. The figure 4.7 represents a three HMM states per phone.

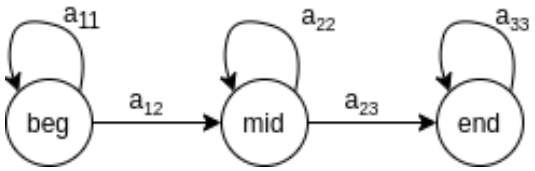


Figure 4.7: Three state HMM model.

The phones which are made out of words do not carry information about the surrounding phones. For example, the base form pronunciations for "mood" and "cool" would use the same vowel for "oo". In practice, the realizations of "oo" in the two contexts are very different due to the influence of the preceding and the following consonant [8]. Since the words are decomposed into context-independent phones, it fails to capture the substantial degree of context-dependent variation that exists in real speech. Context independent phone models are called monophones.

Acoustic Model Training

A word is made by concatenating phones defined by a pronunciation dictionary. The parameters of these phone models such as - transition probability, observation probability and initial probability are estimated from training data consisting of speech and corresponding orthographic transcriptions.

Estimation of HMM parameters is commonly performed using a maximum likelihood estimate (MLE), which maximizes the probability of seeing the training data given the parameters of the model. MLE is done using an iterative procedure called as expectation-maximization (EM) algorithm where an iteration alternates between performing an expectation step (E), which computes expectation of log-likelihood of training data using the current estimate of parameter and a maximization step (M) which computes new parameters by maximizing the expected log-likelihood of parameters found in the expectation step. This leads to the Baum-Welch algorithm [5]. The MLE criterion can be approximated by maximizing the probability of each training sample, given the model, which is also known as Viterbi forced alignment. In Viterbi training, since the phonic states are known for the corresponding utterance, the path is forced to follow certain states. Figure 4.8 highlights the main steps of the Viterbi training process [5], which is :

Step 0: Load training data.

Step 1: Compute observation probabilities.

Step 2: Assign observations to specific states and Gaussian components within the model (E-step).

Step 3: Maximize the model parameters

The alignments are obtained using the Viterbi algorithm, but the update equation is the same as Baum-Welch. To begin with, GMM distributions are initialized with a uniform value. A common choice for the initial parameter set $\lambda^{(0)}$ is to assign the global mean and covariance of the data to the Gaussian output distributions and to set all transition probabilities to be equal. This gives a so-called flat start model [8].

During alignment Viterbi algorithm tries to align audio features to HMM states, followed by the re-estimation of GMM distribution parameters. The process is repeated many times until the model parameters converge.

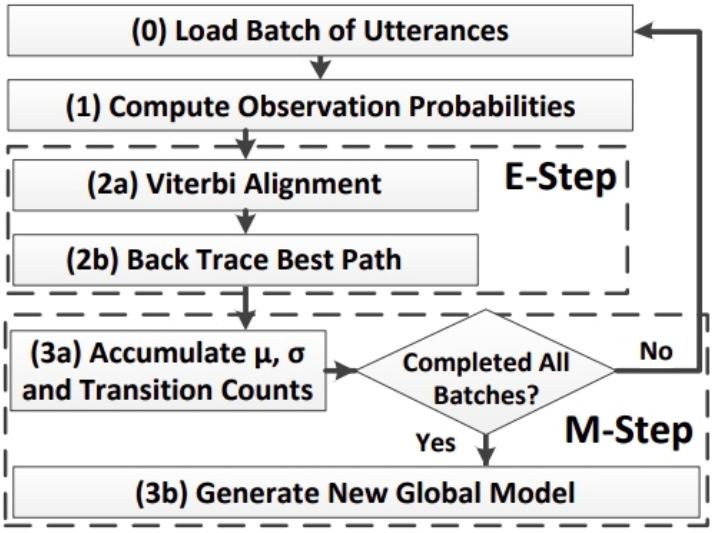


Figure 4.8: This shows one iteration flow of Viterbi training. It starts with initial estimates of the parameters $\lambda = [a_{ij}, b_j()]$. During E-step, expected state occupancy count and expected state transition count are computed. In M-step parameters λ, mean and covariance of states are re-estimated. [5]

For each utterance $Y^{(r)}$, r = 1,..., R, of length $T^{(r)}$ the sequence of base phones, the HMMs that correspond to the word-sequence in the utterance, is found and the corresponding composite HMM is constructed. Given a set of training observations $Y^{(r)}$ and a HMM state sequence $1 < j < N$, the observation sequence is aligned to the state by Viterbi alignment. The alignment is found by maximizing by following equation :

$$\phi_T^{(N)} = \max_i [\phi_T^{(i)} a_{iN}] \tag{4.22}$$

for $1 < i < N$ where

$$\phi_t^{(j)} = \max_i [\phi_{t-1}^{(i)} a_{ij}] b_j(y_t) \tag{4.23}$$

with intial conditions given by

$$\phi_1^{(1)} = 1 \tag{4.24}$$

$$\phi_1^{(j)} = a_{1j} b_j(y_1) \tag{4.25}$$

where $\phi_1^{(j)}$ denotes that the observation probability at time 1 by state j. $\phi_T^{(N)}$ is the best score (highest probability) along a single path, a time T, which accounts for the first T observation and ends in state N. a_{ij} represents the transition from state i to j. So equation4.25 represents the probability of transiting from initial state to state j and emitting the observation corresponding to the state j with probability $b_j(.)$. The output probability $b_j(.)$ is defined as equation 4.21. In Viterbi training, model parameters are updated based on the single best alignment of individual observations to states and Gaussian components within the states.

The sequence of states which maximizes $\phi_T^{(N)}$ implies the alignment of training data observations with states. The means ($\hat{\mu}^{(jm)}$) and variances ($\hat{\Sigma}^{(jm)}$) of observation densities are updated using an indicator function $\gamma_t^{(rjm)}$ which is 1 if $y_t^{(r)}$ is associated with the mixture component m of state j and is zero otherwise.

$$\hat{\mu}^{(jm)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T^{(r)}}\gamma_t^{(rjm)}y_t^{(r)}}{\sum_{r=1}^{R}\sum_{t=1}^{T^{(r)}}\gamma_t^{(rjm)}} \tag{4.26}$$

$$\hat{\Sigma}^{(jm)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T^{(r)}}\gamma_t^{(rjm)}(y_t^{(r)}-\hat{\mu}^{(jm)})(y_t^{(r)}-\hat{\mu}^{(jm)})^T}{\sum_{r=1}^{R}\sum_{t=1}^{T^{(r)}}\gamma_t^{(rjm)}} \tag{4.27}$$

and mixture weights are computed based on number of observations allocated to each component

$$c^{(jm)} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T^{(r)}}\gamma_t^{(rjm)}}{\sum_{r=1}^{R}\sum_{t=1}^{T^{(r)}}\sum_{l=1}^{M}\gamma_t^{(rjl)}} \tag{4.28}$$

From this alignment, transition probabilities are estimated from the relative frequencies as:

$$\hat{a}_{ij} = \frac{A_{ij}}{\sum_{k=2}^{N}A_{ik}}$$

where A_{ij} is the total number of transition from state i to state j.

The above process is iterated until the parameter values converge.

Decoding, Lattice Generation and WFST

With the acoustic, pronunciation lexicon and language model built, the task for decoding is to decode the audio clips into words. Decoding uses the test data. The most likely word sequence w^* given a sequence of feature vectors $Y_{1:T}$ is found by searching all possible state sequences arising from all possible word sequences for the sequence which was most likely to have generated the observed data $Y_{1:T}$.

An efficient way to solve this problem is to use dynamic programming called the Viterbi algorithm [12]. Let $\phi_t^{(j)} = \max_\theta p(Y_{1:t}, \theta_t = s_j; \lambda)$, i.e., the maximum probability of observing the partial sequence $Y_{1:t}$ and then being in state s_j at time t given the model parameters λ. This probability can be efficiently computed using the recursion:

$$\phi_t^{(j)} = \max_i[\phi_{t-1}^i a_{ij}]b_j(y_t) \tag{4.29}$$

It is initialized by setting ϕ_0^j to 1 for the initial, non-emitting, entry state and 0 to all other states. The probability of the most likely word sequence is then given by $max_j\{\phi_T^{(j)}\}$ and if every maximization decision is recorded, a traceback will yield the required best matching state/word sequence.

Likelihood computed by the HMM, and the probability from the language model, P(w), are on different scales [13]. Therefore, some scaling of these two values is necessary before combining them. It is done by adding weight in the language model, also called a language model scaling factor (LMSF). This factor is an exponent on the language model probability P(w). The equation then becomes:

$$w^* = \underset{w}{argmax} p(Y|w)P(w)^{LMSF}$$

The same effect is attained when the acoustic model is scaled down by a factor N. In addition, log-likelihood for the objective function is used to improve numerical stability. So the final objective function becomes:

$$w^* = \underset{w}{argmax}\ (Y|w)^{\frac{1}{N}} P(w) \tag{4.30}$$

$$w^* = \underset{w}{argmax}\ \frac{\log p(Y|w)}{N} + \log P(w) \tag{4.31}$$

Consider a speech recognition system whose lexicon has multiple pronunciations. Because of multiple pronunciations, the Viterbi algorithm may ignore these many pronunciation words in favour of an incorrect word with only one pronunciation path. Another problem with plain Viterbi decoder is when the language model defined using n-gram model goes beyond 2-gram. Since a trigram grammar allows the possibility of a word based on 2 previous words, it is possible that the best trigram probability path for a sentence might go through that word but not the best path. So usually multiple hypotheses of potential utterance are chosen instead of a single best. For Viterbi decoding, the search space can either be constrained by maintaining multiple hypotheses in parallel [30], or it can be expanded dynamically as the search progresses [2, 23, 24].

There are a number of algorithms to extend Viterbi to generate not just the most likely hypothesis but the N-best set of hypotheses where N ranges from 100-1000 [8]. Word lattice [26, 27] is one of the compact and efficient structures for storing multiple hypotheses .

A word lattice consists of a set of nodes representing points in time and a set of spanning arcs representing word hypotheses. To predict a Somali utterance: "ma qaadataa wax dawooyin ah", the corresponding word lattice generated is given as in figure 4.9. The number on the arc is how a word is represented for a word as an integer. As can be seen, there are multiple predicted possible words sequence of the utterance of the Somali text.

The total log score of a sentence is measured according to equation 4.31 where each path holds the acoustic score and likelihood score to a state (or word). The likelihood and acoustic score are measured by summing up the log values across different paths. Also, a relative confidence measure is computed by taking the difference between the total costs of the best and second-best paths in the lattice.

The main task of the decoder is to map HMM states to words using a pronunciation dictionary. A finite-state transducer (FST) consists of a set of nodes whose state transitions

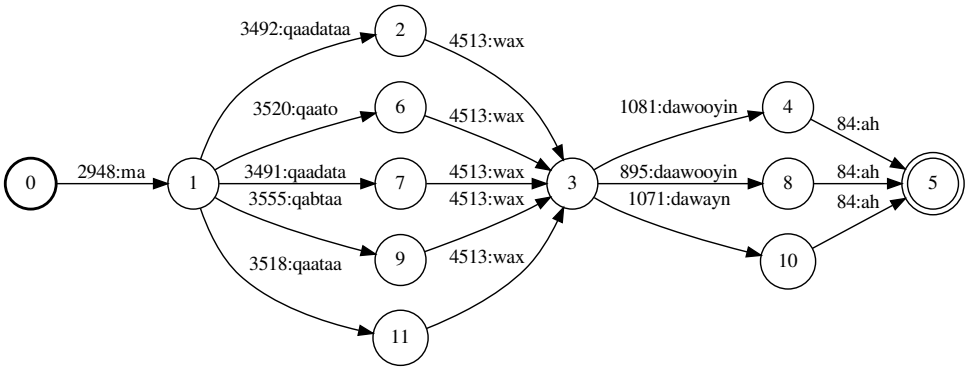


Figure 4.9: This figure shows a word lattice generated for a uttered voice in Somali. Here nodes are points in time and arcs represents hypothesis for words.

are labelled with both input and output symbols. So, a path through the transducer encodes a mapping from input state sequence, symbol or string to an output string. A weighted transducer puts weight on the transition that may represent probabilities, duration, penalties [21].

A weighted transducer can be considered as having states and transition probabilities; hence it is a good way to represent HMM, pronunciation dictionaries, language grammar and phones. As an example, a transducer can be considered, which maps HMM states to phones with weight as HMM transition probability. In the context of ASR, weighted finite state transducer (WFST) can model acoustic, language model and can represent phones as well [21].

4.2 Kaldi

Kaldi is a free open source toolkit designed for only ASR [25]. The tool is majorly written in C++. There are other toolkits available like HTK, CMUSphinx but choosing Kaldi was based on community support, configuring traditional ASR pipeline(GMM), flexibility and less restrictive licensing. Almost all speech recognition uses some graphical method to solve HMM based speech recognition. Kaldi decoding and training graphs use WFST and word lattices [25].

5 Results

Basic python packages were used for preprocessing the audio files and text transcriptions. The analysis of text corpus was done using *nltk* package [4]. The audio files were preprocessed using *librosa* [18] in python. Rest of the computation was done using scripts provided by Kaldi. The script *mfcc.sh* was used to extract the MFCC acoustic features. *ngram-count* was used to compute the language model. The resulting output was in ARPA format, which is a file that contains the various probability of n-gram. An example is mentioned in appendix A.1. *arpa2fst* was used to convert an ARPA format language model into an FST. Training the monophone models was done using *train_mono.sh*. The command *makegraph.sh* combines the HMM structure in the trained model, the Lexicon and the Grammar and creates a decoding graph in the form of an FST. For decoding, features of the test set were extracted using the *mfcc.sh* script. The graph created using *makegraph.sh* was used to decode the test utterances using the script *decode.sh* which generates lattices of word (phone) sequences for the data given the model, which was then used for scoring. The script *lattice-to-nbest* was used to compute and get log values from the lattice. All the scripts take configuration options which can be set by passing them as flags to script as follows: `<option-name> <value>`. Beside above scripts several other recipe files were necessary for computation which are mentioned in appendix section A.

All the audio files were downsampled to 8Khz. The α level for pre-emphasis was kept at 0.95. The acoustic scaling factor was chosen to be to 0.1, which is considered as standard. The length of the frame window was kept to 25ms, and the Hamming window was used. For MFCC, delta and double delta features were also computed. Two discounting methods, namely, Kneser-Ney and Witten-Bell, have been used with different N-gram language models. In particular 1,2,3,4 n-gram models have been tried.

Table 5.1 gives an overview of the decoding result on test data-1 using GMM-HMM algorithm. The data was trained on monophones using different n-gram language models with two different discounting methods. With Kneser-Ney smoothing on different language models, the minimum WER and SER achieved was 24.10% and 46.15% respectively on test data-1.

Table 5.1: Result of GMM-HMM using Kneser-Ney discounting on test data-1.

WER	SER	n-Gram
55.04%	84.62%	1
24.82%	46.15%	2
24.10%	50%	3
24.82%	48.08%	4

With Witten-Bell discounting and different n-gram language model the minimum WER and SER achieved was 17.63% and 40.38% respectively on test data-1 which is shown in table 5.2

Table 5.2: Result of GMM-HMM using Witten-Bell discounting on test data-1.

WER	SER	n-Gram
56.47%	84.62%	1
19.42%	40.38%	2
18.35%	40.38%	3
17.63%	42.31%	4

The result is certainly far better than the result in the paper [19] which had a minimum WER as 51%. Comparison with languages such as Swedish [15] and Russian [14] was also made. The result of which is given in table 5.3. Only the best results were chosen for each model for comparison purposes.

Table 5.3: Results on different language.

Language	Model	Train	WER
Swedish	GMM-HMM (mono)	400 hrs	48.86%
Swedish	DNN-HMM	20 hrs	21.03%
Russian	GMM-GMM(triphone)	25+ hrs	25.32%
Russian	DNN-HMM	25+ hrs	20%

Total log score was also calculated for top 5 predicted sentences for each of the audio from test data-1. For example, the total log score of top 5 predicted sentence was computed for a correct utterance "daacuun caloole" was computed using 3-gram language model with both Witten-Bell discounting and Kneser-Ney discounting which is shown in table 5.4 and 5.5.

Table 5.4: Likelihood, prior and total cost of top 5 predicted sentence for "daacuun caloole" using 3-gram language model and Kneser-Ney discounting

No	Predicted sentence	Acoustic likelihood score	Language model score	Total score
1	waa cun calooleed	1872.38	37.49	1909.87
2	waxaan calooleed	1881.67	28.68	1910.36
3	waxan calooleed	1879.96	30.99	1910.95
4	daacuun calooleed	1874.88	36.28	1911.17
5	waa cod calooleed	1876.35	34.86	1911.21

A sentence is predicted better if it has a lesser total log score. Comparison of table 5.4 and 5.5 shows that the total score using 3-gram Witten-Bell discounting has the lowest score when compared to the 3-gram Kneser-Ney method. Comparing tables 5.4 and 5.5 shows that the predicted word "daacuun caloole" is the best-predicted word which has the lowest

total log score of 1909.02 .

Table 5.5: Likelihood, prior and total cost of top N predicted sentence for "daacuun caloole" using 3-gram language model and Witten-Bell discounting.

No	Predicted sentence	Acoustic likelihood score	Language model score	total score
1	daacuun caloole	1884.98	24.03	1909.02
2	daacuun calooleed	1874.88	34.41	1909.30
3	waxaan calooleed	1881.67	28.41	1910.08
4	waa cun calooleed	1872.38	38.07	1910.45
5	waxan calooleed	1879.96	30.76	1910.72

Since the 3-gram model gave better results than other N-gram models in both discounting methods, confidence measure (CM) was calculated on test data-1. Table 5.6 shows the comparison of CM values between 7 of the audio files from test data 1.

Table 5.6: Confidence measure of predicted sentence for 7 audio files from test data-1.

No	Audio file	CM-Witten-bell	CM-Kneser-Ney
1	Audio file 1	0.31	0.48
2	Audio file 2	4.69	0.98
3	Audio file 3	1.22	1.14
4	Audio file 4	1.46	0.09
5	Audio file 5	1.84	0.38
6	Audio file 6	3.71	2.89
7	Audio file 7	4.19	2.41

Figure 5.1 shows a comparison of CM for both discounting methods on the whole of test data-1. It can be seen the CM calculated using Witten-Bell discounting is better than Kneser-Ney for most audio files.

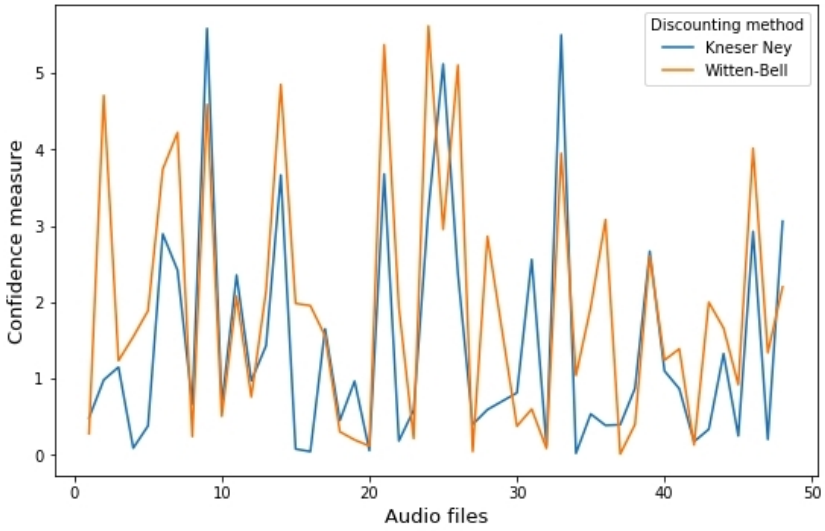


Figure 5.1: This figure shows the confidence measure on the test data-1 using both discounting method.

Audio spoken by a female and its medical transcription was scraped from the web to evaluate the efficacy of the model. After necessary preprocessing, the audio was given to the trained model to predict the sequence of words, but the model was not able to predict the utterance as shown in table 5.7. The results were the same for Witten-bell and Kneser-Ney discounting method.

Table 5.7: Result of GMM-HMM on test data-2 on both discounting method.

WER	SER	n-Gram
100%	100%	1
100%	100%	2
100%	100%	3
100%	100%	4

Below example shows the predicted outcome for one of the audio from test data-2. It can be seen that the model could not recognize the actual sentence.

Original Sentence maxaad dhareemeysaa?

Predicted Output dhiig áma

6 Discussion

6.1 Results

With the fundamental model, the performance of HMM-GMM model with Somali data in table 5.1 and 5.2 performed well with comparison to result from table 5.3 for the amount of data. This performance can be since the audio data was spoken only by one speaker, and it was clean without any background noise. Assuming that the same words were spoken in different contexts might have also attributed to a good performance. Also, it could be that the words which occurred in the training set were also present in the test set. Referring to figure 3.1, it can be seen that most of the words which occurred in the training set also occurred in the test set. Which means the voice data which occurred in training occurred on the test set as well.

A much better comparison of similarity between training and test would be word sequence. A comparison of frequently occurring bigram words for train and test sentence is shown in figure 3.2. Some of the most frequently occurring bi-gram words present in the training set were present in the test set as well. So having good training data is essential for predicting words from test data.

More than 90% of the data was used to train the model. So this is another reason why it generally performed well on test data-1. More data was chosen to train so that the model could learn better parameters to predict on unseen data (test data-2). But, the resulting trained model failed to recognize the sentences which were not from the Worldish dataset, as shown in table 5.7. WER and SER for test data-2 was 100%. There are multi-facet reasons for the failure of this model, but majorly it is the lack of variability in the data.

The pitch of voice, accent varies from person to person. So the training set must have a lot of data so that the model can generalize well when the new data is seen. Since a single female medical translator recorded the voiced data in the training set, the model failed to perform well on different voiced data. WER also plays its role as it does not entirely identify the true meaning of the predicted sentence.

Furthermore, the data used in paper [15, 14] for Swedish and Russian consisted of multiple speakers, and it was tested on different speakers. Hence, the result from table 5.3 had around 20% WER even with huge amounts of training data. Different speakers make the prediction task difficult if there is not enough variability in training data which is evident from table 5.7. This essentially implies that there should be more number of training data consisting of audio spoken by different speakers.

Table 5.4 and 5.5 gives the top 5 predicted versions of the uttered sentence "daacuun caloole". On comparing the total score from table 5.4 and 5.5, it is evident that the score from the language model makes a difference between predicted sentences as the acoustic model score is nearly the same. For a specific discounting method log score gives the best sentence for an utterance. However, a comparison between two discounting methods based on log score implies that there is very little difference in total log score, yet the prediction by Witten-Bell discounting was better than the other method. So having two log scores for which the total score values are similar, it does not account for the stark difference in prediction.

From table 5.6 it is evident that for 3-gram language model, Witten-Bell discounting method is better than Kneser-Ney method. It appears that Witten-Bell discounting performs better than Kneser-Ney in predicting new unknown words for the test data-1. Figure 5.1 shows that for the whole of test data-1, Witten Bell mostly had a higher value of CM in predicting utterance than Kneser-Ney. However, one pitfall is that since the confidence measure computed is the difference between the total costs of the best and second-best paths in the lattice. So for each 3-gram lattice, a relative score is calculated, which tells by how much degree a best-predicted sentence is from the second-best.

Table 5.6 is useful to compare the confidence between two different methods and pick one. However, it provides no information when evaluated standalone. For example, the confidence of audio file 7 being 4.19 provides no information on how good is the predicted sentence, but when compared to 2.41, it certainly informs that the earlier model is better. A better confidence measure would be to have a metric which can rate a predicted sentence on a scale from 0 to 1. WER can also be a better metric in computing the error, but as mentioned before, it too has flaws.

6.2 Method

In HMM-GMM based models, different modules like the acoustic model, language model, and the lexicon model play a crucial role in the performance of the model. The working model of HMM-GMM model poses the following difficulty:

- Different modules use different training methods and data sets. Each module needs to be independently optimized with its own optimization objective functions. The optimality of each module does not mean that the resulting model will be optimal [11]. This makes HMM-GMM based models difficult to use in LVCSR, especially in deploying to various machine platforms.
- As mentioned before HMM assumes conditional independent assumptions so that the training and model construction becomes simple. In real speech scenarios, voice samples are not independent [28, 11].
- The actual distribution of voiced data to be GMM cannot be said with certainty.

Because of the above limitations and abundance in data, many of the current models use DNN. Due to lack of data, HMM-GMM model was build for this thesis.

The errors on test data-1 can be attributed to many factors like not having enough training data. Also, it is possible that poor modelling of the pronunciation dictionary might have contributed to the errors since the pronunciation dictionary was made with the limited knowledge of the author and could not be verified by experts. For some words like *la'aan*, the author had no knowledge of how to map to its phones. Though words like these were included as a part of the training, it must have contributed to the error as no accurate mapping was present. Ideally, the pronunciation dictionary is made by linguistics. There are packages available for popular languages which can map words to phones.

It was surprising to see that Witten-Bell performed better than Kneser-Ney. There have been studies which compared Kneser-Ney, Witten-Bell and other discounting methods [6]. The studies were conducted with different dataset sizes. In all the comparison, Kneser-Ney performed better than Witten-Bell. It is not clear why Witten-Bell performed better than Kneser-Ney in the current implementation.

6.3 Future Work

The current HMM-GMM model can be studied or further improved by taking several steps. In this thesis, MFCC has been used for feature extraction. Many features transform can be employed on top of MFCC. One of the features transform called cepstral mean and variance normalization [13] normalizes the vector features and divides by its mean. PLP, which is similar to MFCC, can also be used to extract features. Instead of log compression, it uses the cube root. Also, different window length can be tried to check if the model improves. There are several windows, such as the Hann window which can also be used to capture frame level information.

Instead of calculating Δ and $\Delta\Delta$ features, the original feature space of MFCC can be projected to a space with the lower dimension using LDA to get new features [17]. One of the potential areas where the current model can be improved is the use of a proper pronunciation dictionary created by a linguistic expert. For the discounting method in the language model, Jelinek-Mercer discounting can be used as well. In study [6] conducted, Jelinek-Mercer discounting appeared to be better than Kneser-Ney and Witten-Bell discounting.

It would also be interesting to see how transfer learning can be used here. In transfer learning, the pre-trained model is fine-tuned to the existing data. However, it would be interesting to see what type of pre-trained model should be used for speech recognition. Should the pre-trained model need to be trained in the Somali language? Is it necessary to initially train on language which is similar to Somali and available abundantly? These are some of the open questions. Furthermore, the for transfer learning a minimum number of training data has been above 40 hours[9, 16, 29] but it is another area of research to check how transfer learning fares with data less than 6 hours. Although the results indicate that the model has performed well on test data, it fails on the unseen data. It is essential to get data spoken by different speakers so that the model can generalize better.

Furthermore, instead of GMM, DNN acoustic models can be used, to check if this model can improve upon the results achieved in this thesis. However, this model will also fail for the same reason as the HMM-GMM model. Nowadays, many modern frameworks are used to do automatic speech recognition which involves DNN, which requires much data. So going ahead, more data needs to be acquired to get a conclusive result.

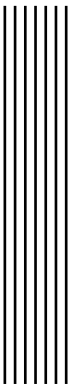
7 Conclusion

In this study, a basic speech recognition model using HMM-GMM framework was implemented. MFCC was used to represent the audio feature. 1-gram, 2-gram, 3-gram and 4-gram language models were analyzed with both Kneser-Ney and Witten-Bell discounting.

First, two research questions can be answered by comparing the WER and SER of the model with other research papers [15, 14]. The minimum WER and SER achieved for the test data-1 was 17.63% and 40.38% respectively. The current implementation fared better given the amount of data on comparison with results from other research papers [15, 14]. But this comparison is entirely not right because the other research [15, 14] had multi speaker audio dataset while this thesis trained on data which was recorded by single speaker.

To answer third research question, the effect of the language model in tandem with the discounting method on the resulting model was studied and analyzed. It was seen that the Witten-Bell discounting method performed better than Kneser-Ney. For Kneser-Ney, WER did not improve much beyond the 2-gram model, whereas with the Witten-Bell discounting method, the WER kept dropping with each increasing n-gram model. However, the study of this relationship was not explored beyond the 4-gram model.

Though the results from the test sets are better in comparison with the amount of training data used and with other papers, the model cannot be said as reliable because the data set provided by Worldish had no variability. It was proved by testing the model on audio clips scraped from the internet. The trained model gave 100% WER and SER for the unseen dataset. The reason being that the data provided by Worldish was clean without any background noise. In real-life scenarios, the voiced data consists of background noise. Implementing the current model would fail in real scenarios. This answers the final research question. For the model to perform better, more diverse data is needed so that the model can generalize better.



Bibliography

[1] Nimaan Abdillahi, Pascal Nocera, and Jean-François Bonastre. "Automatic transcription of Somali language." In: *INTERSPEECH*. 2006, pp. 289–291.

[2] X. Aubert and H. Ney. "Large vocabulary continuous speech recognition using word graphs". In: *1995 International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. 1995, 49–52 vol.1.

[3] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Y. Bengio. "End-to-end attention-based large vocabulary speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, pp. 4945–4949.

[4] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. Vol. 44. O'Reilly Media, Inc., 2009, pp. 421–424.

[5] S. Buthpitiya, I. Lane, and J. Chong. "A parallel implementation of Viterbi training for acoustic models using graphics processing units". In: *2012 Innovative Parallel Computing (InPar)*. 2012, pp. 1–10.

[6] Stanley F. Chen and Joshua Goodman. "An Empirical Study of Smoothing Techniques for Language Modeling". In: vol. cmp-lg/9606011. 1996, pp. 310–317.

[7] Steven B. Davis and Paul Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *Acoustics, Speech and Signal Processing, IEEE Transcation On* 28.4 (1980), pp. 357–366.

[8] M.J.F. Gales and Steve Young. "The Application of Hidden Markov Models in Speech Recognition". In: *Foundations and Trends in Signal Processing* (2007).

[9] Pegah Ghahremani, Vimal Manohar, Hossein Hadian, Daniel Povey, and Sanjeev Khudanpur. "Investigation of transfer learning for ASR using LF-MMI trained neural networks". In: Dec. 2017, pp. 279–286. DOI: 10.1109/ASRU.2017.8268947.

[10] R. A. Gopinath. "Maximum likelihood modeling with Gaussian distributions for classification". In: *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*. Vol. 2. 1998, pp. 661–664.

[11] A. Graves and Navdeep Jaitly. "Towards end-to-end speech recognition with recurrent neural networks". In: *31st International Conference on Machine Learning, ICML* 5 (2014), pp. 1764–1772.

[12] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. "Spoken Language Processing: A Guide to Theory, Algorithm, and System Development". In: (2001).

[13] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. USA: Prentice-Hall, Inc., 2009. ISBN: 0131873210.

[14] Irina Kipyatkova and Alexey Karpov. "DNN-Based Acoustic Modeling for Russian Speech Recognition Using Kaldi". In: 2016, pp. 1–6. ISBN: 978-3-319-43957-0.

[15] Emelie Kullmann. "Speech to Text for Swedish using KALDI". MA thesis. 2016.

[16] Julius Kunze, Louis Kirsch, Ilia Kurenkov, Andreas Krug, Jens Johannsmeier, and Sebastian Stober. "Transfer Learning for Speech Recognition on a Budget". In: *Rep4NLP@ACL*. 2017, pp. 1–8.

[17] Vaclav Matousek and Pavel Mautner. "Text, Speech and Dialogue, 10th International Conference, TSD 2007, Pilsen, Czech Republic, September 3-7, 2007, Proceedings". In: (2007).

[18] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th python in science conference*. Vol. 8. 2015, pp. 18–24.

[19] Raghav Menon, Astik Biswas, Armin Saeb, John Quinn, and Thomas R Niesler. "Automatic Speech Recognition for Humanitarian Applications in Somali". In: *SLTU*. 2018, pp. 1–5.

[20] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. "Acoustic Modeling Using Deep Belief Networks". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 20 (2012), pp. 14–22.

[21] Mehryar Mohri, Fernando Pereira, and Michael Riley. "Speech Recognition with Weighted Finite-State Transducers". In: *Handbook of Speech Processing* (2008).

[22] Luděk Müller and Josef Psutka. "Comparison of MFCC and PLP parameterizations in the speaker independent continuous speech recognition task." In: 2001, pp. 1–28.

[23] J.J. Odell, V. Valtchev, P.C. Woodland, and S.J. Young. "A One Pass Decoder Design For Large Vocabulary Recognition". In: *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 199*. 1994.

[24] Stefan Ortmanns, Hermann Ney, and Xavier Aubert. "A word graph algorithm for large vocabulary continuous speech recognition". In: *Computer Speech and Language* 11.1 (1997), pp. 43–72. ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.1996.0022. URL: http://www.sciencedirect.com/science/article/pii/S0885230896900224.

[25] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. "The Kaldi Speech Recognition Toolkit". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, 2011, pp. 1–4. ISBN: 978-1-4673-0366-8.

[26] Fred Richardson, Richard Schwartz, Senior Scientist, Bolt Beranek, and Newman Inc. "Lattice-Based Search Strategies For Large Vocabulary Speech Recognition". In: (Mar. 2000).

[27] Henry S. Thompson. "Best-first enumeration of paths through a lattice—an active chart parsing solution". In: *Computer Speech and Language* 4 (1990), pp. 263–274.

[28] Dong Wang, Xiaodong Wang, and Shaohe Lv. "An Overview of End-to-End Automatic Speech Recognition". In: *Symmetry* 11 (2019), p. 1018. DOI: 10.3390/sym11081018.

[29] J. Yi, J. Tao, Z. Wen, and Y. Bai. "Language-Adversarial Transfer Learning for Low-Resource Speech Recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.3 (2019), pp. 621–630.

[30] S.J. Young, N.H. Russell, and J.H.S Thornton. "Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems". In: (1989).

[31] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN: 1447169670.

A Appendix

For Kaldi to be able to generate the language model a directory called dict, for dictionary, has to be provided by the user. In this directory the following files must exist :

lexicon.txt *nonsilence.txt* *silence.txt* *optional_silence.txt*

lexicon.txt has the following format: `<word> <phone1> <phone2>`. *lexicon.txt* contains repeated entries for the same word, on separate lines, if we have multiple pronunciations for it. A tab separated the word and transcription, and phones in the transcription were separated with blank space. The final number of entries in the *lexicon.txt* is 28732. The lexicon file with Somali phone has been previously mentioned in 3.3.

The file *nonsilence.txt* contains a list of nonsilence phones respectively. The *nonsilence.txt* is the list of all phones used in the transcriptions. Variations of the same phone, different stress, is put on the same line in the file. Figure A.1 represents nonsilence phones used in this thesis.

b t d͡ʒ ħ x d r s ʃ ɖ ʕ g p f q k l m n
w h v j æ e i ɔ u æː eː iː ɔː uː

Figure A.1: Somali phones

The *silence.txt* file contains lists of the silence phones. These phones should be mutually exclusive from the phones mentioned in *nonsilence.txt* and together, should contain all the phones. The *silence.txt* and *optional_silence.txt* is the marker or markers for silent or unknown segments in the audio. The file *optional_silence.txt* contains a single phone which can optionally appear between words: The markers used are SIL and SPN. The sound SPN, meaning spoken noise, is paired with < UNK >, meaning unknown, in the dictionary and Kaldi maps all words that appear in training data but not in the lexicon to < UNK >.

To be able to train the acoustic model, the speech data and transcriptions have to be prepared, and a training directory has to be created. Files needed in the training directory are:

utt2spk *spk2utt* *text* *wav.scp*

spk2utt and *utt2spk* are each others mirrors. File *utt2spk* mentions for each utterance, which speaker spoke it. The format for *utt2spk* is: `<utterance-id> <speaker-id>`. Following is an example of utt2spk file.

som01_00130c46-c666-4c70-b224-d568b7b88d15_AudioAttachment som01

som01_001900a9-0cb5-43b7-a047-65974dce5985_AudioAttachment som01

som01_002d2c4e-cf47-4326-8124-500f3dd2ad5d_AudioAttachment som01

The format for *spk2utt* file is `<speaker-id> <utterance-id1>`.

text file contains the transcript of each utterance and is on the format `<utterance-id> <Transciption>`. When information about speaker is present the speaker-id should be put as a prefix of the utterance-id so that the utterance-id will be `<speaker-id>-<utterance-id>`. In the below sample from the file `text`, four utterances from the speaker "som01" is listed:

som01_dfe912e9-4f32-4427-bb78-f25e767062dd_AudioAttachment maya maan isku dayin isleegsiinta cadaadiska

som01_e00ccab0-8e4d-486e-b418-72c3d33da445_AudioAttachment haa waan leeyahay sonkor ama macaan

som01_e00d44d9-ec60-4811-aa3c-21ad58301a85_AudioAttachment ma mid joogto ahba mise wuxuu imaada marmar

som01_e016df8b-d161-4f5d-a213-6a0b5663a990_AudioAttachment ma ka jawaabi karo suaashan

The *wav.scp* contains the path to the .wav file for each utterance. The format is `<utterance-id> <path-to-utterance.wav>`. The existence of the above files is also a requirement for the speech data used for testing. The test directory is prepared in the exact same way. For decoding, arbitrary speech data, the files needed are spk2utt, utt2spk and wav.scp.

A.1 ARPA file format

```
kaldi/egs/Project_somali$ cat corpus.arpa
\data\
ngram 1=4
ngram 2=3

\1-grams:
-0.4771213    </s>
-99    <s>    -0.1249387
-0.4771213    chase    -0.1249387
-0.4771213    dogs    -0.1249387

\2-grams:
-0.30103    <s> dogs
-0.30103    chase </s>
-0.30103    dogs chase
```

```
\end\
kaldi/egs/Project_somali$
```