Data Selection for the Training of Deep Neural Network in the Framework of Automatic Speech Recognition

Juan Karsten
Supervisor: Dominique Fohr and Irina Illina
Multispeech team, Inria Loria Nancy
Universite de Lorraine

A thesis submitted for the degree of
Master of Computer Science
2017

This thesis is dedicated to
someone
for some special reason

Acknowledgements

plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle.

Abstract

Contents

List of Figures

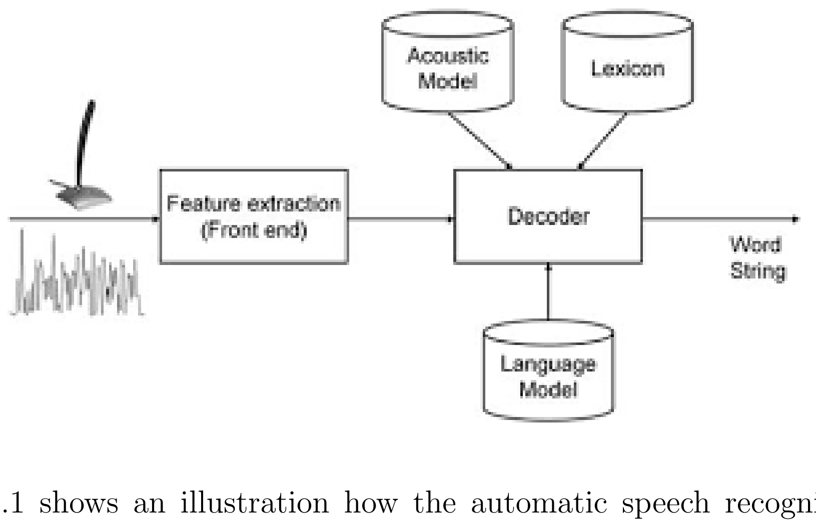
Chapter 1

Introduction

1.1 Motivation

Automatic speech recognition(ASR) is one of the subfield of natural language processing(NLP) with many practical applications. Some examples of practical applications in speech recognition; for example: automatic closed captioning for hearing-disabled persons [24], taking notes of conversations between doctors and patients [12], automatic closed captioning TV broadcasts [38], and many more. This internship will focus on the application of automatic transcription on television broadcasts.

Figure 1.1: automatic speech recognition



Picture 1.1 shows an illustration how the automatic speech recognition works. ASR receives acoustic inputs and outputs a word string. An automatic speech recognition is composed of three components: acoustic model, language model, and lexicon. Acoustic model is a statistical model which represents a relationship between an acoustic input and its corresponding phoneme. Phoneme is a unit (or a group) of sound, for example: /k/, /l/, /sh/, etc. Lexicon is a dictionary which maps from

words to their phonemes. Language model computes the probability of a word string where each word consists of several phonemes. Hence, the lexicon glues the acoustic model and the language model together. These three components work together to search the most likely sentence(from all possible sentences) which represents what people said in the audio.

Despite of the rapid development of speech recognition, there are still many challenges in the field. One of the challenges is training a speech recognizer or adapting a speech recognizer to a new task requires a massive amount of transcribed training data. However, transcribing audio manually is labor intensive and also time consuming. There exist many unlimited supply of training data sources from internet, TV broadcasts, radio, as well as video streaming website, like Youtube. Nevertheless, many of them only possess few corresponding transcriptions or no transcription at all. To build a speech recognition, we can use a training dataset from TV broadcasts which usually have closed captions. Closed captions are close, but not exactly the same as what people said. Furthermore, closed captions are often badly aligned with the audio. These are some examples of good and bad closed captions, as well as noisy utterance.

1. An example of good closed caption. This segment/utterance has the same closed caption as the detailed transcription.

   - Real transcription: That is the very big question that leads to another big question. We're definitely thinking about putting an offer. That's great.
   - Closed caption: That is the very big question that leads to another big question. We're definitely thinking about putting an offer. That's great.

2. An example of bad closed caption. The closed caption is slightly different compared to the real transcription.

   - Real transcription: Russia started badly with the dropping at the hands of Spain. But, they got better and better. Spain looked unstoppable to start with but since then they have looked a little.
   - Closed caption: Russia started badly with at beating at the hands of Spain. Spain looked then they have looked a little

3. Noisy segment. Some segments in training audio may have background noise which makes it hard to be recognized by the ASR. Even, it is sometimes difficult

to be recognized by humans. [Clapping in the background] John Higgins developed very well. He is not five in front.[Still continues clapping in background]

This internship explored how to build a high performance speech recognition system by utilizing less supervision because we did not have correct transcription audio data. The main idea is to use an automatic speech recognizer to automatically transcribe audio data which can be leveraged as a training dataset. The steps of the main idea are as following: firstly, train on all data which are automatically transcribed by other ASR systems, recognize these data with the ASR which we trained, and the compare the decoding transcriptions with the closed captions. Finally, remove all words which do not agree and train only the words which agree.

This idea of using untranscribed data(or unsupervised training) has been proposed by Zavaliagkos and Colthurst in 1998 [39], Kemp and Waibel in 1999 [10]. However, they only utilized small amount of data. Lamel et al were the first who proposed lightly supervised training data with a large amount of training data [15]. In place of utilizing untranscribed data, they trained speech recognition systems by using audio data with closed captions.

Lightly supervised approach is one of the technique to select "good" data. An acoustic model from another task(or another corpus) is utilized to recognize audio data. The decoding results are compared with the closed captions and removed if they disagree. These selected data are trained to generate a new acoustic model which is leveraged to recognize the same or different data. These decoding results are selected again for training a new acoustic model. This internship mainly focuses on the acoustic model component.

The goal of this internship is to try and evaluate the state-of-the-art technique to select the "good" training data. In addition, we proposed multiple other possible solutions to select data. The outcome of this internship will be useful as a training dataset to build a high performance automatic speech recognition.

This internship is expected to contribute in solving data selection problem. Internet has infinitely many audio data, for example: podcast, internet radio, tv broadcast, as well as video streaming. But, mostly they do not have transcriptions at all. In this internship, we conceived speech recognition systems by utilizing TV broadcasts and their corresponding closed captions. In the future, we hope that we use audio data without any transcription as the training dataset.

1.2 Contributions

We make several contributions as following:

1. We give some backgrounds related to speech recognition which helps the readers to understand how the speech recognition works in the nutshell. This background helps to grasp the idea of data selection.

2. We did some literature reviews and summarized the data selection method briefly and completely.

3. We implemented and evaluated the data selection techniques.

4. We proposed new approaches in data selection and assessed the approaches against the current state-of-the-art approach.

1.3 Outlines of the thesis

This report is organized as following. In chapter 2, we elaborate some backgrounds in the automatic speech recognition. The chapter gives necessary backgrounds to the readers how speech recognition works; moreover, the data selection method is further explained. Chapter 3 elaborates the data selection technique which was explored and experimented in this internship. Furthermore, the new data selection methods are also proposed. Chapter 4 tells the setup of experiment and evaluation measurement. Then, chapter 5 shows the result of experiments. Lastly, the report is concluded by chapter 6 which conveys conclusion and some suggestions for future works.

Chapter 2

Background and Literature Review

2.1 Background

2.1.1 Automatic speech recognition

The goal of the automatic speech recognition(ASR) is to search the most likely sentence($\hat{W}$) in a language L given acoustic observation O [9]. ASR has been applied in many applications and domains, for example: automatic closed captioning for hearing-disabled persons [24], taking notes of conversations between doctors and patients [12], automatic closed captioning TV broadcasts [38], and many more.

Observation O can be seen as a sequence of sliced acoustic signal. Each successive index represents the consecutive slices of acoustic input.

$$O = o_1, o_2, o_3, ..., o_t \tag{2.1}$$

A sentence is composed of a sequence of words.

$$W = w_1, w_2, w_3, ..., w_n \tag{2.2}$$

Thus, an automatic speech recognition can be formalized as following:

$$\hat{W} = \text{argmax}_{W \in L} P(W|O) \tag{2.3}$$

Applying the bayes rule to the equation 2.3 produces the equation bellow. Because observations O do not change for every possible sentence, we can assume $P(O)$ as a constant and ignore it.

$$\begin{align} \hat{W} &= \text{argmax}_{W \in L} \frac{P(O|W) \times P(W)}{P(O)} \tag{2.4} \\ &= \text{argmax}_{W \in L} P(O|W) \times P(W) \tag{2.5} \end{align}$$

where $\hat{W}$ is the most likely sentence, $P(W)$ is the prior probability of sentence W computed by the language model, and $P(O|W)$ is the observation likelihood calculated by the acoustic model. This internship mainly focuses on the acoustic model. We will discuss the language and acoustic model more in the next subsection.

2.1.2 N-Gram Language model

A language model is a statistical model which computes the probability of word sequence. One of the most common language model is N-gram language model, which we leverage in this internship. The foundation mathematics of N-Gram language model was proposed by Markov in 1913. Then, Shannon re-introduced N-gram to compute the likelihoods of English word sequences [31]. N-gram language model has been applied in many applications, such as: speech recognition [38], handwriting recognition [27], as well as spelling correction [14].

The goal of language model is to compute the probability ($P(w|h)$) of a word w given previous word history h . The way to compute this probability is by using a large corpus and count the relative frequencies. For example: to count the word w_1 given history: $w_2, w_3, ..., w_n$, the probability is as following:

$$P(w_1|w_2w_3...w_n) = \frac{C(w_1w_2w_3...w_n)}{C(w_2w_3...w_n)} \tag{2.6}$$

Because we can create infinitely possible sentences, it is more efficient to compute probability by applying the chain rule of probability.

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)...P(w_n|w_1^{n-1}) \\ &= \Pi_{k=1}^n P(w_k|w_1^{k-1}) \end{aligned}$$

However, the probability of a word given preceding words is infeasible to be computed. Instead of computing the entire preceding history, N-gram only approximates the probability with last few words. For example: the bigram LM approximates the probability of a word given history by the probability of a word given the last word only $P(w_n|w_{n-1})$. This approximation(or assumption) is called Markov assumption.

A major problem exists in N-Gram language model. Information inside a corpus might be limited. Some word sequences probably do not exist in the corpus, but they are perfectly acceptable word sequences. If we calculate the N-Gram likelihood for these sequences, we will get zero value. The solutions to the problem are backoff and smoothing. Backoff allows the N-gram LM to use the value of lower N-Gram LM if we have zero probability in the higher N-Gram. Smoothing is a technique to avoid zero probability for every word sequence. If a word sequence does not exist in the corpus, the LM still produces a non-zero value even though it is small. We used Kneser-Ney smoothing [13] which has been implemented in SRILM framework.

To evaluate the performance of a language model(LM), perplexity can be applied to the LM. The intuition behind LM is given two N-gram LM, the better model is the one which predicts test set better than the other. Lower perplexity means that the model is better. Given a language model and a test corpus, perplexity is calculated as following [2] [11]:

$$PP = exp(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|h_i)) \tag{2.7}$$

where N is the number of words in test corpus, w_i is the i-th word, and h_i is the history of word w_i.

2.1.3 Acoustic model

The acoustic model(AM) is commonly based on hidden markov model(HMM). Thus, before talking about the acoustic model deeply, it is better to elaborate HMM first.

2.1.3.1 Markov chain

A weighted state finite automaton is a finite state automaton where the arc between nodes has probability how likely the path is taken. Markov chain is one kind of weighted state finite automata which the input sequence determines which state the automaton will go.

Markov chain can be defined formally as following:

- $Q = q_1 q_2 ... q_N$ a set of states
- $A = \begin{bmatrix} a_{01}a_{02}a_{03}...a_{0n} \\ a_{11}a_{12}a_{13}...a_{1n} \\ ... \\ a_{01}a_{02}a_{03}...a_{0n} \end{bmatrix}$ is a transition probability matrix where a_{ij} represents the probability of moving from state i to state j. In addition to that, the sum of outgoing probability of every state must be summed into 1. $\sum_{j=1}^{n} a_{ij} = 1$.
- q_0 and q_F represent initial and final states.

2.1.3.2 Hidden markov model(HMM)

Markov chain is only usable if the event is directly observable in the world. However, we are interested in some events which are not observable or hidden for some problem. For example: acoustic event is observable, but word event is not observable in speech recognition problem. Hidden markov model fixes this problem.

Hidden markov model is formalized as

- $Q = q_1 q_2 ... q_n$ is a set of states
- $A = \begin{bmatrix} a_{01}a_{02}a_{03}...a_{0n} \\ a_{11}a_{12}a_{13}...a_{1n} \\ ... \\ a_{01}a_{02}a_{03}...a_{0n} \end{bmatrix}$ is a transition probability matrix where a_{ij} represents the probability of moving from state i to state j such that $\sum_{j=1}^{n} a_{ij} = 1$ for every i.
- $O = o_1 o_2 ... o_T$ is a sequence of T observations.
- $B = b_i(o_t)$ is observation likelihoods(or emission probability) where state i emits probability given observation o_t.
- q_0, q_F are initial and final states.

  Hidden markov model employs two assumptions:

  1. Markov assumption. The state is only dependent to the previous state.

  $$P(q_i|q_1 q_2 ... q_{i-1}) = P(q_i|q_{i-1}) \tag{2.8}$$

  2. Output independence assumption. Output observation o_i is only dependent with the state q_i which produces o_t and independent with other states and other observations.

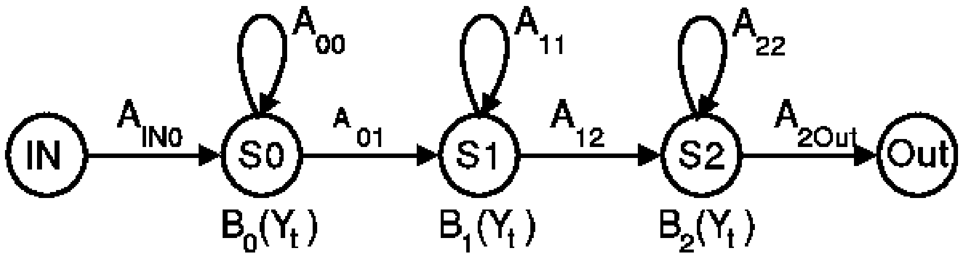
  $$P(o_i|o_1 o_2 ... o_i .. o_T q_1 q_2 ... q_i .. q_N) = P(o_i|q_i) \tag{2.9}$$

2.1.3.3 Applying HMM to acoustic model

In section 2.1.3.2, HMM is explained. Now, it is time to apply HMM on acoustic model.

1. States
   A state in acoustic model can represent a word, a phone, or a sub phone depending on the problem we are trying to solve. For the digit recognition problem, a state represents a sound of one number. A sequence of phone can be represented as a word string. Hence, for more complicated task, like recognizing TV broadcast, it is better to make a state representing a phone or even a sub phone. A sound of a phone is usually influenced by the preceding and successive phone. Therefore, we use three HMM states(for each phone) which capture beginning, middle, and end of the phone. The sub phone state is commonly called as a senone.

Figure 2.1: A triphone HMM model consisting of three emitting states(beginning(S0), middle(S1), and end(S2) of a phone), a transition matrix A, and emission probability(observation likelihood) B [32]

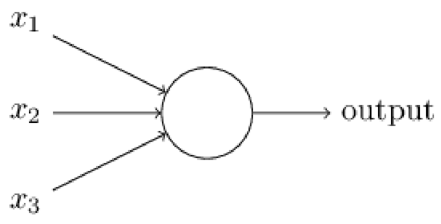


2. Transition matrix
   The transition matrix in speech recognition is the probability of moving from the current sub phone state to the successive sub phone state. Unlike HMM, Speech recognition HMM does not allow arbitrary transition because of the sequential nature of language and speech. A state can go to itself or to the next states, but it can not go back to the preceding state. This kind of left-to-right constrained HMM is called Bakis network.

3. Observation
   The observation is a sequence of acoustic feature vector. The acoustic input is divided into small chunks and converted into a feature vector.

4. Observation likelihood
   Observation likelihood is the probability of a feature vector which is generated by a sub phone state.

Figure 2.2: A perceptron



2.1.4 Deep learning for acoustic model

For our acoustic model, we used deep neural network(DNN) HMM. Beside DNN HMM, there exists gaussian mixture model(GMM) HMM. However, DNN HMM has been proven to outperform GMM DNN in practice [25]. Hence, we only discuss DNN in this report.

2.1.4.1 Feedforward neural network

Feedforward neural network(or sometimes called multilayer perceptron(MLP)) is one kind of neural network. MLP approximates function $f'(\mathbf{x}) = \mathbf{y}$. MLP can be represented as $f(\mathbf{x};\theta) = \mathbf{y}$ and learn parameters θ which maximizes function approximations.

Perceptron is a kind of artificial neuron developed in 1960 by Frank Rosenblatt [29]. A perceptron receives several inputs($x_1, x_2, ..., x_n$), does weighted sum $\sum_i x_i w_i$, and is thresholded by bias b to make a decision: zero or one.

$$output = \begin{cases} 0, \text{if } \sum_i x_i w_i + b \leq 0 \\ 1, \text{if } \sum_i x_i w_i + b > 0 \end{cases} \tag{2.10}$$

A perceptron is too simple to capture non linear pattern. Therefore, the perceptron is modified by applying an activation function into weighted sum. There are several common activation functions, such as:

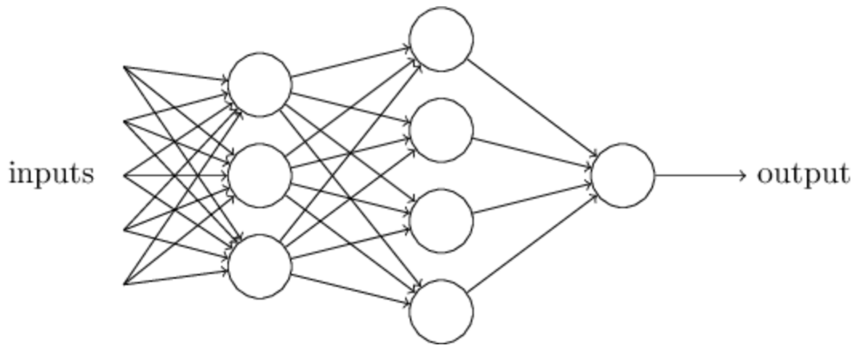
1. Sigmoid
$$a(x) = \frac{1}{1 + e^{-x}} \tag{2.11}$$

2. Rectified linear unit(relu)
$$a(x) = max(0, x) \tag{2.12}$$

3. Softmax
$$a(x) = \frac{e^x}{\sum_i e^{x_i}} \tag{2.13}$$

In addition to activation function, we need to make neuron to capture more complex features by stacking many neurons together. This network is called a feed forward multi layer perceptron. This neural network does not have connections between units which form a cycle. It is also called feedforward because the input values are propagated forward to output. The neural network receives inputs, does weighted sum, and applies activation function. The output of the units in one layer is propagated to the units in the next layer until reaching output layer.

Figure 2.3: Multi layer perceptron



2.1.4.2 Time delay neural network(TDNN)

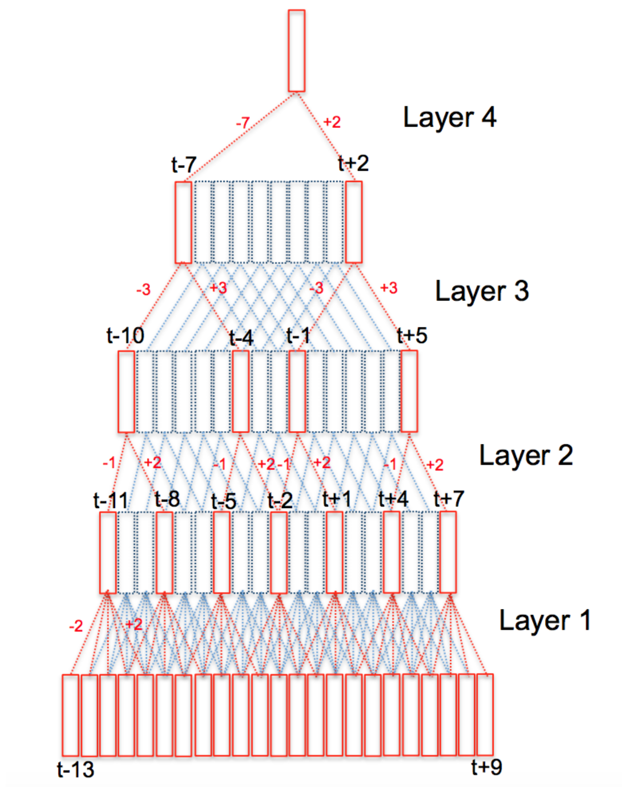
To model a temporal sequence neural network, like the speech recognition problem, the model must have the following properties: it must have multiple layers to learn complex non linear decision surface, represents relationships between events in time, invariant under translation in time, and does not require precise temporal alignment. From all of these properties, time delay neural network(TDNN) qualifies to model a temporal sequence neural network.

There exist two ways to capture long temporal contexts:

1. Feature representation which present the information of long temporal contexts. The examples of the feature representation are TRAPs [7], multiscale spectro-temporal modulations representation [21], deep scattering spectrum representation [1], and modulation frequency feature representations [36]. These feature representations can use the standard feed forward deep neural network to model the temporal sequence neural network.

2. Acoustic model which can learn long temporal contexts and is trained on short-term feature representations. Recurrent neural network(RNN) [30] and time delay neural network [37] are two state-of-the-art acoustic models which are suitable to model this acoustic model.

Figure 2.4: An example of a TDNN architecture. [25]



Peddinti et al showed that TDNN performs better than RNN in term of parallelization during training [25]. Due to recurrent connections in RNN, RNN is not able to fully utilize parallelization in GPU(general processing unit) the same extent as TDNN. In addition, TDNN has been proven to have lower word error rate compared to RNN [25]. Thus, we will focus more on TDNN and trained acoustic models with TDNN in this internship.

Time delay neural network(TDNN) was first proposed by Waibel et al in 1990 [37].TDNN is suitable to model long term temporal relationship within the range of N delays in short term speech features; for example: MFCC feature representations.

TDNN is basically a feedforward neural network. Learning procedure which we used is backpropagation. There are two passes procedure in the backpropagation. The first pass is forward pass where the neural network receives input, does weighted sum, and applies activation function. The output values from the first layer is forward propagated until reaching the output layer. Then, square error is calculated between the expected value and the output of neural network. The second pass is backward pass where the derived square error value is propagated backward to update weights.

Typical TDNN computes all hidden activation in each layer. Figure 2.4 demonstrates an architecture of TDNN. TDNN receives acoustic inputs with some delays to the left and right contexts(before and after time t). Usually, the input context is asymmetric, i.e: more time context to the left than right. The main reasons are to reduce the latency of online decoding and improve word error rate. Each layer splice together contiguous temporal windows within a time range; for example: the third layer splices the frames within range $[t-3, t+3]$ in the figure 2.4. Furthermore, TDNN applies wider splice when going to higher layers in the networks as shown in the figure 2.4. The reason is we force the higher layer to learn wider temporal contexts.

Training a TDNN is still time consuming. The standard TDNN computes all hidden activation in each layer. Consequently, the training time of TDNN is approximately ten times compared to the standard DNN. Peddinti et al found that there exist large overlaps between input contexts at neighboring time steps; thus, they proposed subsampling [25]. The technique generally only splice no more than two temporal contexts with gap instead of contiguous temporal windows. Figure 2.4 illustrates the procedure of subsampling. In the standard TDNN, all hidden units(red and blue) are calculated. In contrast with subsampling technique, only red hidden units are computed. In consequence, the training time of subsampled TDNN is approximately five times faster than the standard TDNN [25]. Moreover, subsampling is able to reduce the model size. Splicing contiguous temporal contexts will require a massive number of parameters. However, subsampling only splices less temporal contexts than the typical TDNN.

2.1.4.3 Hybrid acoustic model

To train a TDNN or other kinds of DNN acoustic model, one needs to label each acoustic input vector. However, our initial data only have utterance level of start and end time. Even, each word does not have start and end time. Each acoustic input must be labeled with its corresponding senone in order to train a DNN-HMM acoustic model.

GMM acoustic model can bootstrap the labeling process [19]. We can train a GMM-HMM acoustic model with flat start, i.e: without previous phoneme-to-audio alignment. Therefore, the step before training a DNN acoustic model is by training a GMM-HMM acoustic model. Then, force align the training set with the acoustic inputs to generate a sequence of senone which corresponds with the transcription in the segments. The aligned data(the acoustic input with its corresponding senone) is utilized to train a DNN acoustic model.

Training a DNN-HMM is a little bit different than the HMM as explained in 2.1.1. An acoustic model usually models the probability $P(o|y)$ of observation(acoustic input) o given its corresponding senone y. Nonetheless, DNN models the probability of a senone given the observation $P(y|o)$. By applying the bayes rule, $P(o|y)$ can be obtained as the following:

$$P(o|y) = \frac{P(y|o) \times P(o)}{P(y)} \tag{2.14}$$

$P(y)$ can be obtained from distribution of a senone in the training set. By using lexicon, each word in the training set can be mapped into the corresponding senone. However, $P(o)$ is infeasible to be obtained. Luckily, the observation o are fixed and can be seen as a constant. Hence, we can provide HMM with the unnormalized probability..

$$\frac{P(y|o)}{P(y)} \tag{2.15}$$

Applying the hybrid approach has been introduced over 20 years ago [4,28]. Nevertheless, it did not outperform GMM until recently. DNN reduces substantial amount of word error rate compare to GMM [6, 8]. DNN excels in improving accuracy due to more representational capacity compared to GMM [19]. There are several factors why DNN outperforms GMM just recently.

- The amount of training dataset has increased. DNN usually works well when there exists a massive volume of training data.
- Training DNN can be done on GPU which accelerates the execution time.
- We have more powerful neural network architecture recently with more number of parameters, more hidden layers(deeper network), and better initialization procedure.

Chapter 3

Methodology

3.1 Lightly supervised data selection approach

Despite of the high development in automatic speech recognition, there exist many challenges. One of the main challenge of speech recognition is to reduce development cost when adapting the speech recognition to other task or creating a new speech recognition for a new language. The biggest development cost is to transcribe the audio data with their exact corresponding transcriptions. To generate a "good" speech recognition, one needs a massive number of training audio and their exact transcriptions. However, transcribing audio is labor intensive and time consuming too. There are unlimited supply of audio data in the internet, television, radio, and other sources. Nonetheless, they usually have a few or even no accurate transcription at all. Some television broadcasts have corresponding closed captions. Closed caption means that the transcription is close, but not exactly the same as what the audio says. Even, the closed captions are often badly time-aligned with the audio.

By utilizing these audio data with the corresponding closed captions, we hope to produce a high performance speech recognizer with less supervision. The main idea is to use the automatic speech recognizer to transcribe audio data which will automatically generate transcriptions, which later be used as training data [15]. The steps are as following. Firstly, we train a bootstrap acoustic model trained on a small manually annotated data. Then, recognize the data with closed captions by using the bootstrap model. We compare the decoding result from the bootstrap model with the closed captions and remove the words which do not agree. Finally, we can train a new acoustic model from the filtering.

The idea of using untranscribed audio data has been explored before(Zavaliagkos and Colthurst in 1998 [39] and Kemp & Waibel [10]), which is called as unsupervised acoustic model training. They utilized small amount of audio data without transcription to train acoustic models. The drawback of their experiments is not using massive volume of training data which is essential to train an acoustic model.

Lightly supervised approach was proposed in 2002 by Lamel et al [15]. In general, the lightly supervised approach operates as following:

1. Train an acoustic model on a small amount of manually annotated data.
2. Recognize(or automatically transcribe) a massive amount of data.
3. Align the automatic transcriptions with the closed captions. Some transcriptions and closed captions might disagree. We can remove or correct these segments.
4. Retrain a new acoustic model using the data which we selected in the previous step.
5. Reiterate from step 2.

These steps can be iterated several times as long as the error rate is decreasing. This method uses the idea of training acoustic model in less supervised manner because the training dataset(closed captions) is not the real detailed transcription.

Using closed caption as training data reduces effort in manual transcription.Detailed transcription process usually takes 20-40 times manual effort compare to live closed captioning. Closed caption is usually produced in real time and less costly compare to the detailed transcription. In addition, the manual transcription is possible to have error [3]. Additionally, some TV broadcasts, such as: CNN headline news, ABC world news tonight, BBC, already have closed captions which can be used as a training dataset. Nevertheless, some problems exist when using the data with closed captions as training dataset. However, training using the closed captions faces several disadvantages compared to the real transcriptions: indication of non speech event, such as: coughing, speaker turn, acoustic conditions: background noise and music. Furthermore, some sentences might be paraphrased, deleted, and changed in word order.

In addition to the closed captions, we can use additional data to build a language model. Other kinds of text are available in internet, such as: news, blog, and closed captions from TV broadcast. However, these data might be less related with the audio data. Some additional sources may be not contemporaneous with the data. Thus, it provides less supervision. We can interpolate these data with the closed captions to build a more general language model.

Despite the fact that closed captions are not accurate, it has been proven that closed captions provides enough supervision in building automatic speech recognition systems [15]. Lamel found that an increase in training data improves accuracy even tough it will increase the model size. Additionally, filter closed captions for the next iteration training set in the lightly supervised technique improves the accuracy slightly. The language model built from closed captions also effectively provides enough supervision in building the ASR. Therefore, lightly supervised technique is our main inspiration to do data selection in this internship.

3.2 State-of-the-art data selection

Some researches have been conducted to investigate the lightly supervised approach to select data in the speech recognition problem [5, 15, 18, 20, 34]. The state-of-the-art data selection was proposed by Lanchantin et al from Cambridge university [17]. Before talking about the state-of-the-art, it is better to discuss some previous related works first.

Lamel et al proposed the lightly supervised approach which harnessed the closed caption as supervision to build a robust speech recognizer [15]. In their experiment, they used a bootstrap acoustic model and biased language model to recognize the audio where the decoding result is compared with the closed caption. Then, the words which do not agree will be removed from training dataset. The new training dataset is utilized to train a new acoustic model. This filtering(removing not-agree words) is called closed caption filtering.

Chan and Woodland explored another way in filtering bad closed captions [5]. They leveraged the confidence measure metric to remove the bad segments. When decoding acoustic inputs, an ASR produces word hypothesis and their corresponding confidence measure. The confidence measure value from each word in each segment is averaged to produce one confidence value per segment. Finally, they determined a threshold confident value to filter out all potentiallly bad segments, where the confident value is lower than the threshold. Moreover, they also introduced to interpolate two different LMs created from two closed caption datasets and merge them into one single LM. This single LM is used as light supervision when decoding acoustic input together with the acoustic model and the lexicon.

Matias et al applied lightly supervised approach on medical conversation data [20]. There are unlimited supplies of medical speech; however, only informal medical reports are available. The medical reports are not the same as what the physicians said in the audio records. This problem matches perfectly with the lightly supervised approach. They explored frame level filtering in place of word level or segment level filtering. Basically, they forced align and compared the decoding results with the closed captions and mark words which are insertion, deletion, and substitution. All corresponding frames which are associated with those marked words are removed or filtered out from training set to produce a new cleaner training set.

The state-of-the-art data selection was proposed by Lanchantin et al from Cambridge university [17]. They used the lightly supervised approach applied on the multi genre broadcast(MGB) challenge. MGB challenge is a challenge to automatically transcribe TV broadcasts. The challenge only provided TV broadcast audio data and their corresponding closed captions. General TV broadcast data are usually recorded in highly diverse environment speech with background music, non speech event, and sound effect. In addition, some closed captions may be different compared to the real transcription due to deletion, insertion, substitution, and paraphrasing. These factors make the challenge interesting as well as complicated.

Lanchantin et al proposed to tackle the problem inspired from the lightly supervised approach. The lightly supervised approach works as following [16].

1. Trained an acoustic model by selecting a subset of 200 hours training set from the total 1600 hours training set. We called this as AM-v1 which is based on deep neural network.

2. AM-v1 recognized the entire training dataset(1600 hours) which resulted in the decoding result. Compare and force aligned the decoding result with the closed captions. By comparing each segment, they calculated phone matched error rate(PMER) and word matched error rate(WMER) as well as average word duration.

3. By utilizing average word duration and phone matched error rate, they selected 700 hour of data(700hr-v1). The way to select the data is

   (a) Reject segments which are not in the range of $0.165 \leq AWD \leq 0.66$

   (b) Sort segments based on PMER and select the top segments until reaching 700 hours set. The threshold for this iteration is 40%. This new selected training data is called 700-v1.

4. Retrain a new acoustic model using the 700-v1. Repeat step 2-4 to create 700-v2, 700-v3 until the data selection algorithm converges.

In every iteration, the language model which they utilized are the same. MGB provides closed captions for each audio and additional text consisting 10 million and 640 million words respectively. From these texts, two language models were built and interpolated into one language model. Furthermore, the merged language model was pruned by 10^{-9} to expedite the decoding process.

A question remains to be asked: when to stop the iteration(the algorithm converges)? The MGB challenge provides a development set which comprises of audio data and their manually transcribed transcriptions. These transcriptions are exactly what people said in the audio with correct time stamp. By recognizing the development set with the ASR system which they built in each iteration, they calculated word error rate(WER). If the WER decreases in the next iteration, continue the iteration. Otherwise, stop the iteration.

3.3 Data

Before talking about the methodology, it is better to talk about the dataset. We have the following dataset:

1. Training set has audio data with their corresponding closed captions.
2. Test set has audio data and their corresponding manual transcriptions. These manual transcriptions are the same as what people said in the audio.
3. Additional text corpus which is utilized for building language model. This additional corpus is different than the closed captions in training set.

3.4 Baseline model

Inspired by the lightly supervised approach(explained in 3.1) and the state-of-the-art data selection approach (explained in 3.2), we developed our baseline model. This baseline model is a benchmark for our proposed model to determine whether the proposed model thrives against the baseline. This section starts with the general explanation of the data selection. Furthermore, we dive deeply into how to build the acoustic model as well as the language model and combine them into a speech recognition system for data selection.

3.4.1 Data selection

The data selection will be divided into two process pipeline which are intertwined together. The first pipeline is to select data and the second one is to evaluate how good the data selection is. The evaluation pipeline is also important to know when to stop the iteration of the data selection pipeline.

Figure 3.2 demonstrates the overall data selection pipeline. Here is the explanation how it works:

1. Build a more constrained language model. In the data selection, we have two different language models: the more constrained and the less constrained language model. The more constrained is built from the closed captions of the full training set. In contrast, the less constrained language model is built from interpolation of the closed caption language model and the additional text corpus language model. The more and less constrained is utilized in the data selection and evaluation pipeline respectively.

2. Randomly select a subset of the training set. The small training set is used to train an initial acoustic model(acoustic model 0).

3. Train an initial acoustic model(acoustic model 0) by utilizing the audio and the closed captions in the small training set.

4. Decode the full training set with the acoustic model 0, the more constrained language model, and the lexicon. This will produce the new decoding results and new values of PMER and AWD. The new values of PMER and AWD are obtained from comparing the closed captions and the decoding results. Minimum edit distance is the algorithm to compare and compute how many substitutions, deletions, and insertions. After

5. Select the closed captions from the training set based on PMER and AWD. This will be the new training set for training the next acoustic model. The selection works as following:

   (a) Select the segments which have AWD within the range $0.166 \leq AWD \leq 0.65$. This range follows the range which was used in [17].

   (b) Sort the segments based on the new value of PMER ascendingly.

   (c) Choose the top segments to make a new small training set.

6. Continue the step 3-5 until the data selection does not improve anymore.

The last step from the data selection pipeline says to do iteration over and over again until the technique does not improve. The technique improves when selecting the closed captions which are closed to the real transcription. However, until some points, it may select the data which can not improve the acoustic model or even the same data in the next iteration. Therefore, the evaluation pipeline is necessary to know when to evaluate how good the data selection is and stop the iteration. Figure 3.1 illustrates the evaluation pipeline. Here is the explanation how it works:

1. In each iteration, a new acoustic model is trained.

2. By utilizing the acoustic model, the less constrained LM, and the lexicon, the development set is decoded. By comparing the decoding result and the real transcription in the development set, one obtains the word error rate. The word error rate is the metric to evaluate the data selection. If the WER decreases in the next iteration compare to the previous iteration, the data selection improves; otherwise, it stops improving and we can stop the iteration.

Figure 3.1: Evaluation pipeline

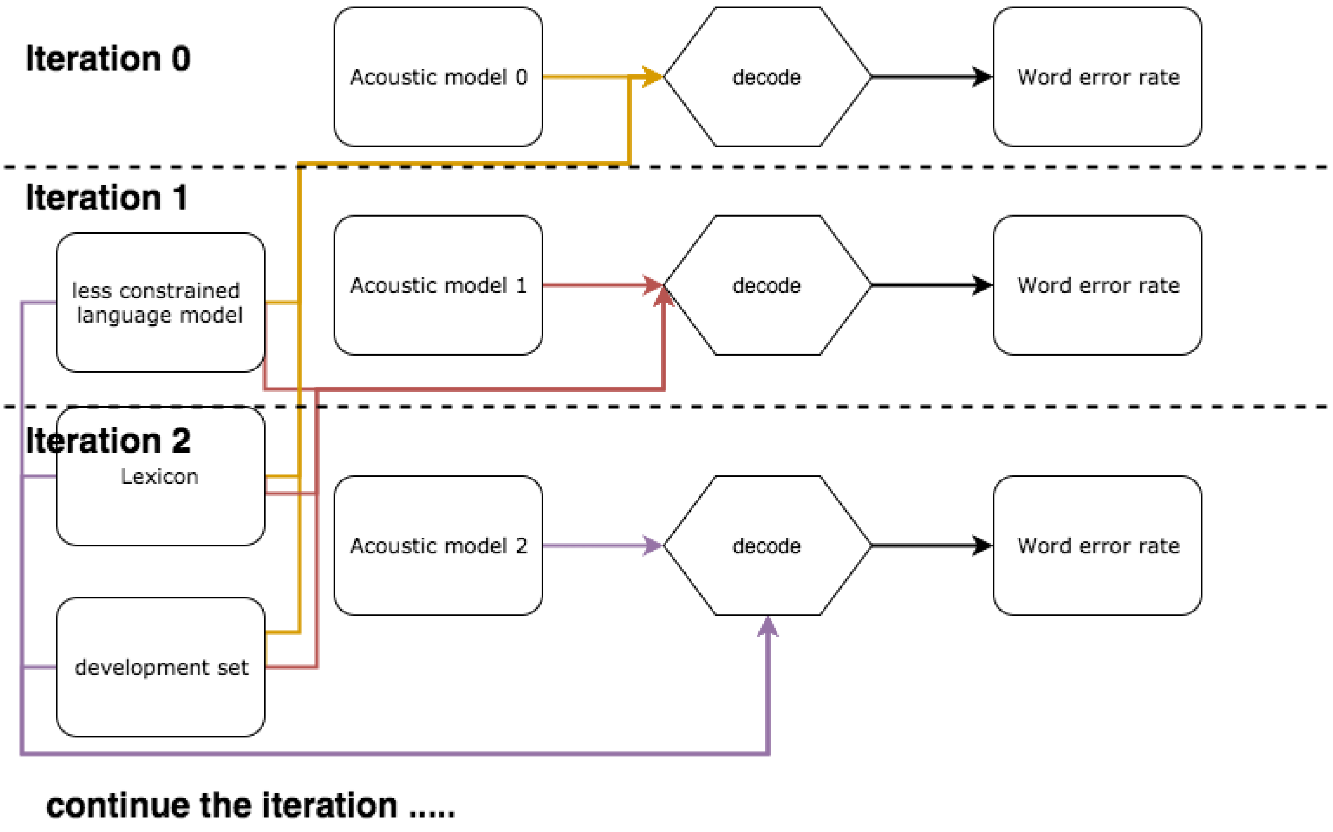
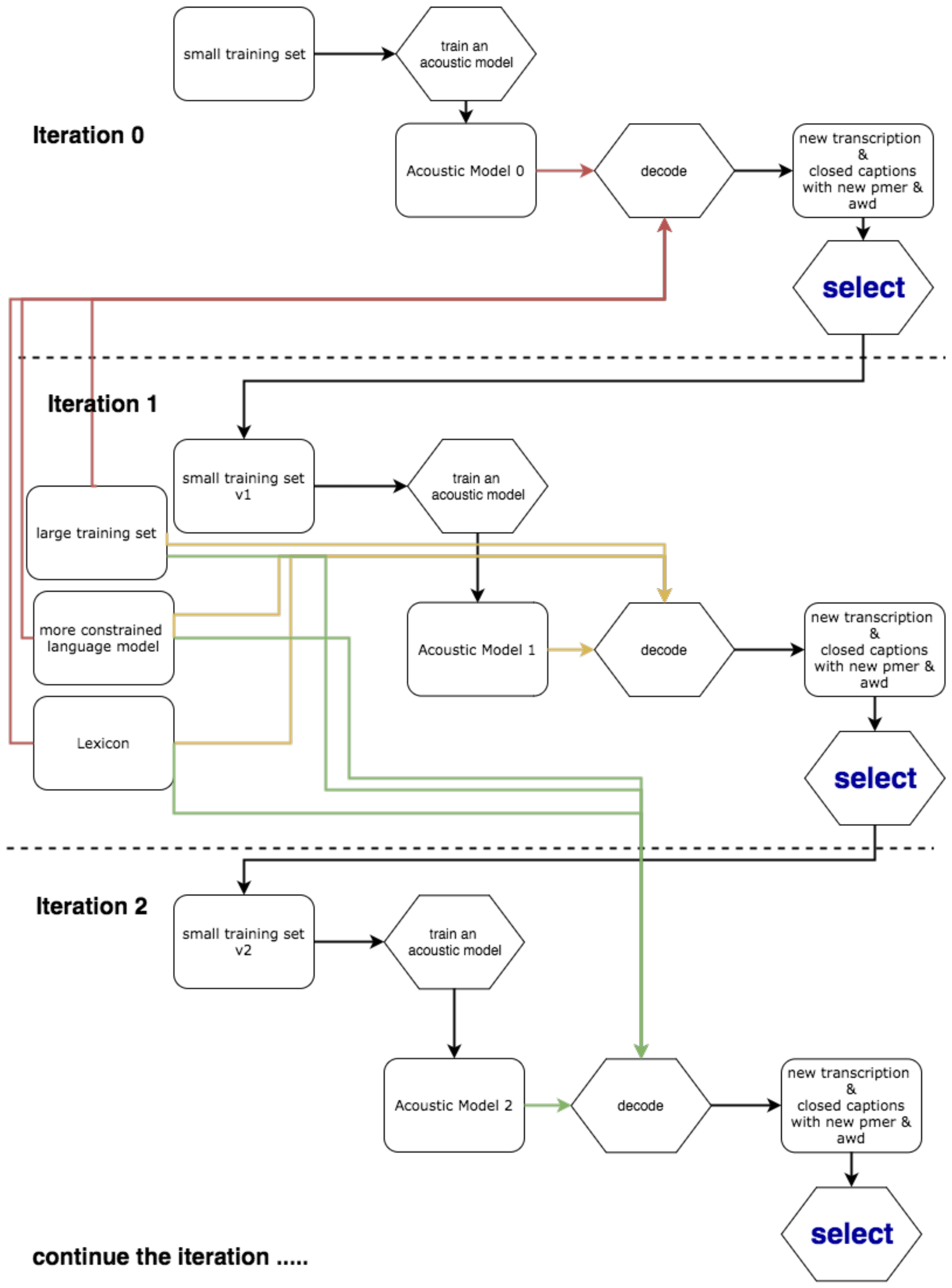


Figure 3.2: Data selection pipeline



3.4.2 Building the speech recognition systems

The previous section explains the data selection process. Nevertheless, it does not elaborate the way to train the acoustic model and decode. This section elaborates how the speech recognizer works together as illustrated in the figure 3.3.

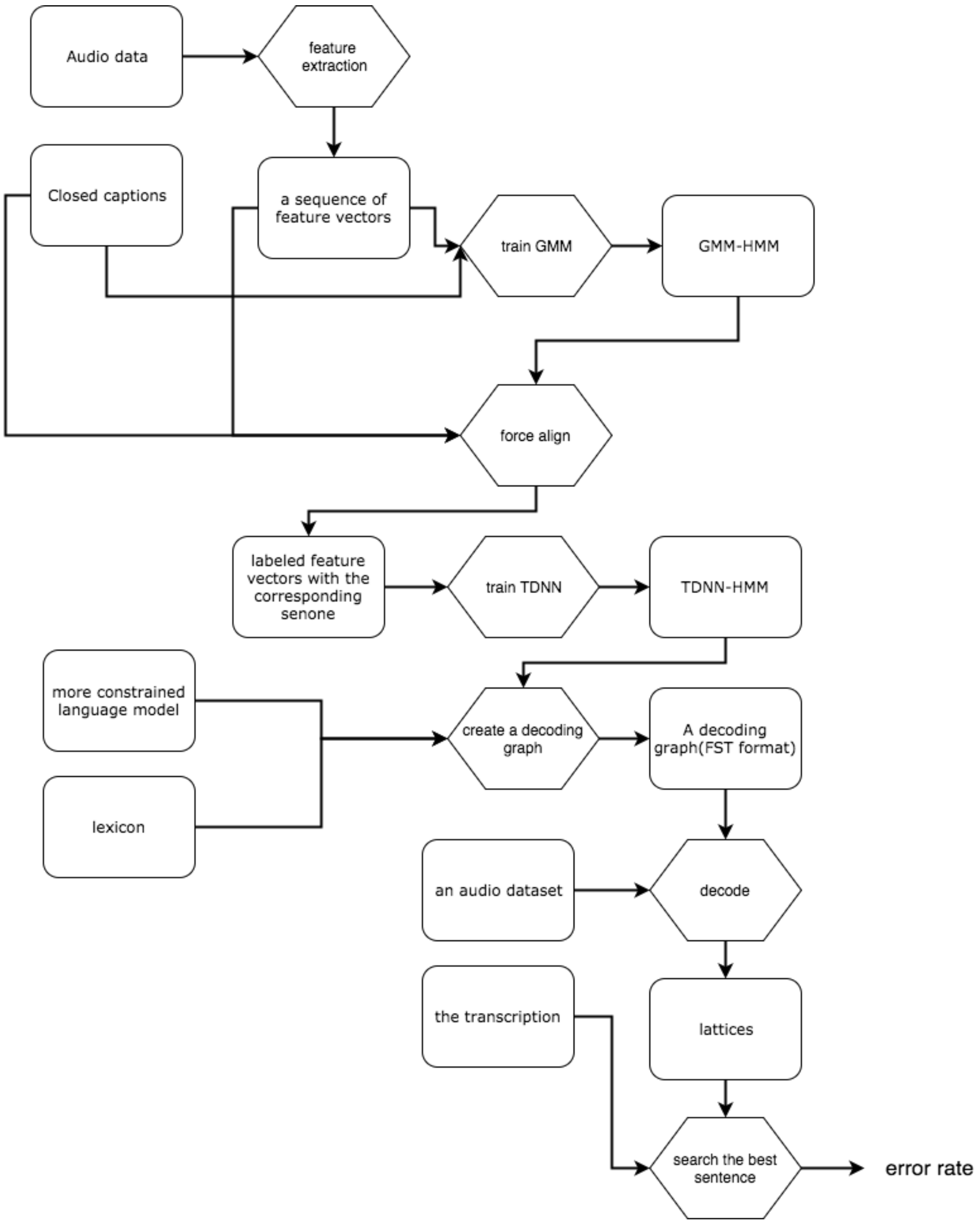
Before training an acoustic model, it is necessary to turn the audio data into a sequence of acoustic feature. The audio will be divided into small chunks and each chunk will be converted into a feature vector.

The hybrid HMM acoustic model is the approach to build the language model as explained in the section 2.1.4.3. Training TDNN needs a training dataset which has training labels. Nevertheless, the training dataset only has several utterances with start and end time. It is our job to label each acoustic feature input(each small chunk of acoustic audio) with their corresponding senone. GMM-HMM is able to be trained on the flat start. The closed captions and the audio data are the inputs of GMM acoustic model training. After training GMM, GMM forces alignment of each audio chunk and label them with their senone. Closed captions might not be perfect for GMM force alignment since they are not exactly the same as the real transcription. The word error rate may be poor in the first iteration; thus we need to do data selection in the next iteration.

After force aligning and labelling the acoustic inputs, TDNN acoustic model is trained. The TDNN acoustic model, the less constrained language model, and the lexicon are combined together to make a decoding graph. The decoding graph is based on weighted finite state transducer(WFST). WFST is a finite automata whose state transition is labeled with input, output, and a weight which encodes a probability to go to the next state. The transition maps the input symbol sequence to an output string. WFST is able to encode HMM, lexicon, and n-gram language model [22]. Thus, WFST is a perfect representation which maps the acoustic input to a word string.

When decoding an audio dataset, the decoder does not return the most probable sentence given the acoustic inputs. Instead, it returns a directed graph containing the subset of possible paths to generate a probable sentence. This directed graph is called lattice. The nodes in lattice represent points in time, while the transitions represent states of combination acoustic model, language model, and lexicon. The lattice can be used later for decoding and rescoring with different language model or acoustic model [23]. From the generated lattice, we can search the best sequence with different and more sophisticated(but slower to do) configuration or more complex language model.

Figure 3.3: Training and decoding with the speech recognition system



3.5 New proposed models

The general idea of the proposed models is to combine three different automatic speech recognition(ASR) systems by varying the language model but the same acoustic model. Combining three different ASRs is expected to complement and fix other ASR's mistake.

The language models are built with different constraints(the least, medium, and the most constraint language models). The proposed models work almost the same as the baseline model: training acoustic model with the subset of training data(audio data and their closed captions), recognizing and selecting the full training data, and repeating the steps to do more data selection. However, the difference is when recognizing and selecting data. Three different ASRs recognize the full training set, combine, and select the decoding results as following:

1. Average PMER and AWD of three ASRs.
   Three different ASRs recognize the full training set and compute new values of PMER and AWD. Each corresponding segment in three ASRs averages their PMER and AWD. The average PMER and AWD are utilized to do the data selection(the same data selection as explained in the baseline model).

2. Combine PMER and AWD of three ASRs with a proposed combination algorithm
   Here is the pseudocode how the algorithm works.

   Listing 3.1 the proposed combination algorithm

   ```
   For each segment:
       Keep only segments which are in 0.166<AWD<0.65
       and 0.03<APD(Average phone duration)<0.25 range
           If one ASR segment has zero PMER,
               Select the segment and corresponding closed
               captions
           Else
               If two segments have the same phone sequence
                   Select the segment and the corresponding
                   decoding result transcription
               Else
                   Sort by PMER. Choose top segments with
                   the lowest PMER. The closed captions will
                   be chosen.
   ```

   There are some intuitions behind the pseudocode:

   - Average phone duration(APD) is introduced to detect whether the segments are properly recognized by the speech recognizer in phone level. In contrast, AWD The duration of a phone must be between 0.03 second and 0.25 second.

$$APD = \frac{\#\text{phones in the segments}}{\text{duration of the segment}} \tag{3.1}$$

- Line 8-10 is expected to fix the closed captions with the correct transcription. The closed captions are not exactly the same as the real transcriptions. When two different ASRs yield exactly the same decoding transcription, the decoding transcriptions probably matches closer to the real transcription compared to the corresponding closed captions.

3. Utilize a randomly different training set for each iteration.
   In our baseline model, we always recognize the same full training set and select the best subset from the training set. Instead of using the same training set, we randomly choose a new full training set with more or less the same total speech duration. The intuition behind this is randomness may increase the performance of the model.

Chapter 4

Implementation

4.1 Data set

For this internship, we used TV broadcasts downloaded from the internet.

- The audio files are taken from TV broadcasts. Total duration of audio is around 200 hours.
- TV broadcast transcriptions(closed captions) of the audio files are provided. The closed captions are close to what the audio says, but not exactly the same as the speakers of the audio said. In addition to the closed captions from 200 hours TV broadcasts, we have the closed captions from 1600 hours of TV broadcasts as well.
- 640 million words of TV subtitles are provided. In addition to that, the 640M subtitles are filtered to avoid overlap with the closed captions from the TV broadcasts.

We prepared two data sets: training set and development set. The training dataset has a total duration of around 200 hours of audio. To evaluate our model, we have a development set. Some metadata are provided in our data, such as: speaker id of the transcript, genre of the show, date and time of the show, as well as television channel where the show was aired. In addition, each segment has start and end time when the segment was shown in the TV broadcast. All files in full training set have closed captions. In contrast, all files in development set have closed captions as well as manually annotated human transcription. The transcript is manually and carefully annotated by human. Thus, the transcript is believed to be the closest transcription spoken by speakers in audio files.

Table 4.1 shows the statistics of the training set and the development set. The second column represents the number of TV shows in each genre; the third column represents the total duration of each genre. Lastly, the forth column(the last column) shows the total duration of speech when the speakers spoke in the audio.

Table 4.1: Dataset duration

No	Data set	#shows	Duration(h)	Aligned speech (h)
1	training set	274	193	149
2	development set	12	8	6

After obtaining 200 hours of data, a 100 hours subset was created by selecting one from two files from 200 training set. The 200 hours and 100 hours training set are named as train.200 and train.100 respectively.

Table 4.2: Statistics of train.200 per genre

Genre	**#shows**	**Duration(h)**	Aligned speech(h)
advice	40	26	22
children	51	19	14
comedy	19	8	6
competition	30	21	16
documentary	29	22	14
drama	20	14	10
events	24	38	29
news	61	42	37

4.2 Experiment setup

In this section, we explained how we did the experiments. First, we elaborated how to build language models, then the acoustic model, and lastly how to decode the data set for the data selection and the model evaluation.

4.2.1 Language model

4.2.1.1 Library: SRILM

SRILM is a toolkit which comprises of a collection of executable programs, which are written in C++, for creating and applying statistical language models for speech recognition and other natural language applications [33]. SRILM supports several operations for language model(LM), such as: creating a language model from a text corpus, interpolating several language models into one, pruning a language model, and estimate perplexity of a language model against the test corpus.

The below code is an example how to make a 4-gram language model with input from *CORPUS* and interpolated with Kneser-ney interpolation [13]. The syntax is pretty straightforward and easy to understand. The resulted language model will be written in arpa language model format.

```
ngram-count -order 4 -kndiscount -interpolate -sort -text CORPUS -lm LM
```

4.2.1.2 Implementation

This internship experimented with three different kinds of language model for speech recognition. All language models are based on four gram language model. We will describe language model from the most constraint to the less constraint

1. LM.200
   This language model, named as LM.200, was produced from the closed captions of 200 hours training set(train.200). LM.200 was pruned by 10^{-9}. Pruning means deleting 4 gram sequences which are less than 10^{-9}. LM.200 was used for the baseline speech recognizer to select the "good" subset of data.

Table 4.3: Statistics of LM.200

No.	File	Information
1	word.200	33,914 words

2	LM.200	uni-gram=33916
		bi-gram=402299
		tri-gram=111868
		4-gram=64801

3	LM.200.1e-9	ngram 1=33916 ngram 2=402299 ngram 3=98566 ngram 4=51215

2. LM.7weeks+subtitles.limited.1e-9
   A language model was generated from the closed captions of 1600 hours transcription(7weeks TV broadcast closed captions) and interpolated with a language model from the big TV subtitles with ratio 0.9/0.1. The language model was limited by top 160,000 frequent word list and pruned by 10^{-9} resulting in LM.7weeks+subtitles.limited.1e-9. The language model and together with an acoustic model and a lexicon was utilized to compute error rate of the development set.

No.	File	Information
1	word.160k	160,000 words
2	LM.7weeks	ngram 1=91836 ngram 2=1817881 ngram 3=980925 ngram 4=847736
3	LM.subtitles	ngram 1=756644 ngram 2=27144330 ngram 3=37162518 ngram 4=57912280
4	LM.7weeks+subtitles	ngram 1=768522 ngram 2=27441072 ngram 3=37312673 ngram 4=58183256
5	LM.7weeks+subtitles.limited	ngram 1=160002 ngram 2=24926818 ngram 3=32102763 ngram 4=44253079
5	LM.7weeks+subtitles.limited.1e-9	ngram 1= 160002 ngram 2= 5251197 ngram 3= 3876171 ngram 4= 2453067

3. LM.genres
   Eight language models were generated from each genre(LM.documentary, LM.news, LM.events, LM.drama, LM.competition, LM.comedy, LM.children, and LM.advice). Then, each language model was interpolated with the big subtitle LM with ratio 0.9/0.1, limited by the top 160,000 word list, and pruned by 10^{-9} threshold. The language model was used in the new proposed algorithm only.

No.	File	Information
1	LM.documentary	ngram 1=37542 ngram 2=437631 ngram 3=135632 ngram 4=90381
2	LM.news	ngram 1=44588 ngram 2=661625 ngram 3=305473 ngram 4=255125
3	LM.events	ngram 1=23717 ngram 2=306938 ngram 3=125934 ngram 4=95203
4	LM.drama	ngram 1=21345 ngram 2=202078 ngram 3=60920 ngram 4=40073
5	LM.competition	ngram 1=33269 ngram 2=385257 ngram 3=124584 ngram 4=89642
6	LM.comedy	ngram 1=20180 ngram 2=165073 ngram 3=39353 ngram 4=23877
7	LM.children	ngram 1=27029 ngram 2=287141 ngram 3=84063 ngram 4=57851
8	LM.advice	ngram 1=30545 ngram 2=397766 ngram 3=154416 ngram 4=108794

To build language models we mentioned before, we utilized SRILM. In addition to building language models, SRILM is able to interpolate several language models, limit vocabularies of a language models, prune a language model by a threshold, and many more.

4.2.2 Acoustic model

4.2.2.1 Library: Kaldi

Kaldi is a toolkit, written in C++, which comprises several tools which are useful to experiment and build a speech recognition system [26]. Kaldi supports several operations to build a speech recognition system, such as: extracting feature vectors, acoustic modeling, training acoustic models, and creating decoding graph as well as decoding audio inputs into lattices. Kaldi provides several features, such as:

- It support finite state transducer(FST), which is based on OpenFST library. The FST is used to build a decoding graph with the inputs of acoustic model, language model, and lexicon.
- It is extensible and provides complete recipes. Kaldi gives many recipes to build acoustic model, such as: gaussian mixture model, deep neural network, etc.
- It supports linear algebra and utilizes the standard linear algebra library: BLAS and LAPACK.

- It has been tested thoroughly to ensure the quality of the library.

4.2.2.2 Acoustic model implementation

Before training the DNN acoustic model, GMM-HMM must be trained first. GMM-HMM is able to perform force alignment on the flat start(without phoneme-to-audio alignment). The labeling of each acoustic input is compulsory to train the TDNN acoustic model.

We utilized TED-LIUM's recipe(nnet3 in Kaldi) to train the TDNN acoustic model. The feature vector is extracted with MFCC, consisting of 40 feature values. The TDNN architecture has 6 hidden layers, where each hidden unit utilizes rectified linear unit(RELU) activation function. The TDNN outputs around 9000 output nodes with softmax activation function. Moreover, the TDNN architecture is implemented by leveraging the subsampling technique. Table shows the parameters and subsampling splice configuration we used in the architecture.

Table 4.4: Parameters of theTDNN acoustic model

Parameter	**Method**
initial learning rate	0.0015
final learning rate	0.00015
mini batch size	512
epoch	3
subsampling splice	
layer 1	$t-2, t-1, t, t+1, t+2$
layer 2	$t-1, t+2$
layer 3	$t-3, t+3$
layer 4	$t-7, t+2$
layer 5	$t-3, t+3$
layer 6	t

4.2.3 Recognition

The last step after training an acoustic model and building a language model is to recognize a data set. The recognition is useful for data selection(computing the new word error rate as well as phone error rate) and evaluate how good our data selection is. In overall, the recognition(or decoding) process works as following:

1. Create a decoding graph from an acoustic model, a language model, and a lexicon. In Kaldi's implementation, it transforms the acoustic model, the language model, and the lexicon into a weighted finite state transducer(WFST).

2. Decode the dataset by using the WFST. In Kaldi, the decoding process will result in a lattice. Lattice is a decoding graph which represents some variants of the recognition. Providing only one best recognition is not practical; thus lattice is the way to represent the recognition/decoding result.

3. Compute the best path of word or phone in the lattice and calculate the WER as well as PER. Kaldi produces two ctm files(word and phone ctm files) which are the best decoding results which the algorithm can obtain. The ctm file consists of words(or phone) in the transcription with its start and end time relative to the audio. By utilizing sclite, the ctm file(the hypothesized word/phone from the ASR) is compared with the transcriptions(the closed captions or the detailed transcription). Then, the code matches and computes the minimum edit distance to calculate the error rate.

4.2.3.1 Data Selection

To select the good data for the next iteration, the new matched word and phone error rate must be generated from the training set. By running the recognition steps(as explained above) on the training set and comparing the decoding result with the closed captions, a file(the file with the extension of ".prf") is generated. The file contains the number of insertion, deletion, and substitution. By using the error rate equation as bellow, the new matched error rate(WMER and PMER) are computed.

Firstly, the segments are sorted based on the new PMER. Secondly, choose the top 100 hour segments based on the PMER. These selected segments are used for the next iteration.

4.2.3.2 Evaluation

To evaluate how good our data selection is, we need to evaluate the automatic speech recognizer(especially the acoustic model). In every iteration, we train an acoustic model; furthermore, the acoustic model and together with the language model(LM.7weeks+subtitles.limited.1e-9) as well as the lexicon, we recognize the development set. After decoding and computing WER of development set, we compare the WER with the previous iteration. If the new WER is lower, we can continue the iteration; otherwise, stop the iteration.

4.2.3.3 Experiments

4.3 Experiment resources

To run our experiment, we utilized grid 5000 server cluster. Grid 5000 is a computing cluster with a focus on parallel and distributed computing [35]. It has several key features:

- It provides a large amount of resources, over 1000 nodes, and massive volume of hard disk.
- The experiment is highly configurable where we can specify number of nodes and the criteria of the nodes which we will use.
- We can do advanced monitoring of memory usage, cpu usage, and the log of our experiment which later be used to analyze the experiment.

The following is the description of resources which we leveraged for this internship based on the tasks:

1. Train acoustic models: 1 computer with GPU and 8 computers without GPU. At least one computer with GPU is compulsory because we can harness GPU to parallelization the computation of DNN training.
2. Create decoding graph only needs one computer because the Kaldi recipe does not need parallelization for this process.
3. Decoding the ASR leverages 20 computers. Usually one computer can recognize one hour of audio within one hour. Therefore, the more computer we use, the faster the decoding process will be.
4. Lastly, the error rate computation and data selection process only utilizes one computer.

Chapter 5

Experiment result

5.1 Baseline model

The data selection was run four time until it saturates in iteration number 3. Basically, the data selection works as following: train a TDNN acoustic model(AM) on 100 hour randomly selected training set(with closed caption), recognize the training set using the new acoustic model and compute new phone matched error rate, reject segments(utterances) which are not in the AWD range($0.165 \leq AWD \leq 0.66$), sort and select new data based on the lowest phone matched error rate. To evaluate how good the data selection is, the acoustic model and together with the language model as well as lexicon must recognize the development set and compute word error rate. If the word error rate decreases, the iteration can continue; otherwise, stop the iteration.

Chart 5.1 shows the average word duration of iteration 0, 1, and 2. The X-axis is the value of average word duration, while the Y-axis represents the cumulative duration of audio in percentage. All utterances are sorted based on AWD to draw this chart. Two vertical line represents boundaries to select utterances. The utterances which are inside these two lines(the value of AWD is in the range $0.165 \leq AWD \leq 0.66$) will be selected. The result shows that less than 15% data are rejected. Selection by using AWD is important to reject utterances which are badly aligned.

Chart 5.2 demonstrates the phone matched error rate of all iterations. The X-axis and Y-axis represent the value of PMER of each utterance and the duration of audio in percentage respectively. All utterances are sorted based on PMER to generate this chart. It is obvious to see that the next iteration has more zero PMER compared to the previous iteration. Moreover, the value of PMER decreases slightly over several iterations where more and more utterances have less PMER in the next iteration. Nevertheless, the iteration stops in iteration three because the data selection converges(no further improvement in PMER).

Figure 5.1: Average word duration of the iteration

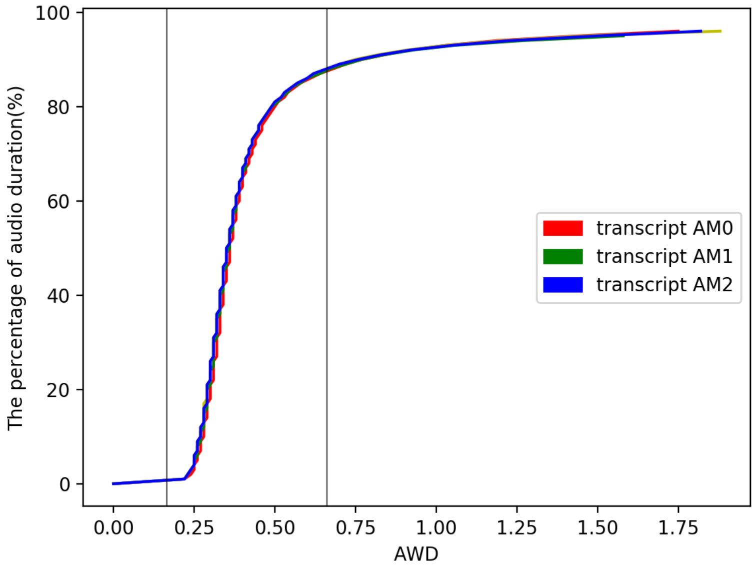


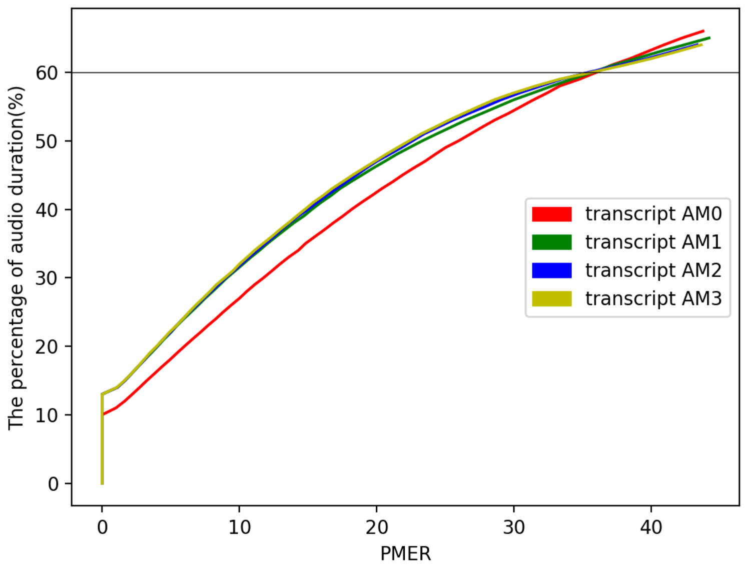
Table 5.1 shows the result of the experiments. Baseline 0(row 1) is the initial iteration(iteration 0) where the training set is randomly 100 hours from 200 hours of training set. Row number 2-4 are the next iteration.

Baseline decoding(row 2) utilizes the audio data and the decoding results(in place of the closed captions) of the previous iteration(baseline 0) as the new training set. Baseline WMER(row 3) and baseline 1(row 4) use the audio data and the closed captions(with new PMER and AWD) as the new training set. However, baseline WMER sorts utterances based on WMER instead of PMER. In contrast, baseline 1 utilized the same method as described in 3.4.1. Consequently, using closed captions and sorting by PMER are the best; thus, the next iteration only experiments with closed captions as training set and sorting by PMER. Baseline 2 and 3 are the second and third(last) iteration respectively. They have the same value of WER; hence, the iteration stops because the data selection has converged.

Table 5.1: WER and PER of acoustic models

No.	Model	WER(%)	PER(%)	PMER thresh(%)
1	baseline 0	43.9	35.8	36.0
2	baseline decoding	42.3	34.3	45.98
3	baseline WMER	41.2	34.1	
4	baseline 1	41.2	33.8	35.71
5	baseline 2	40.5	33.3	35.44
6	baseline 3	40.5	33.3	

Figure 5.2: Phone matched error rate of the iteration



5.2 Proposed models

In addition to the baseline model, we proposed three new models which we hoped it would increase the performance. Table 5.2 demonstrates the result of our technique. Baseline 1 and 2 are the same model as shown in Table 5.1, while other rows show the results of the proposed models. We put baseline 1 and 2 to compare with the proposed models.

As explained in Section 3.5, we proposed three new models:

1. Average PMER and AWD of three ASRs.
   By using the baseline acoustic model in baseline 1, three different ASRs were created by varying three different language models(as mentioned in 4.2.1.2). Each ASR from the three ASRs recognized the full training set and calculated new PMER and AWD. Each corresponding segment in three ASRs averages their PMER and AWD. The averaged training set is selected with the data selection method and trained to generate a new acoustic model. The new acoustic model with LM.7weeks+sub and the lexicon is evaluated as shown in model proposed average(row 3). Because the model does not improve the performance as good as baseline 2, we do not continue to the next iteration.

2. Combine three ASRs.
   We combined the recognition of three ASRs from Baseline 1 by using the algo-

rithm explained in Listing 3.1, and then, we train a new acoustic model from the data. We evaluated the performance of the new model as shown in row 4. The value of WER of row 4(proposed combination 1) and WER of row 2(baseline 2) are the same; but, the PER decreases by 0.1. Therefore, we continue the iteration to see if it can further reduce the error rate.

Then, three ASRs of proposed combination 1 were combined and trained a new acoustic model based on the combination algorithm(Listing 3.1). We evaluated the new model and showed the result in row 5. Unfortunately, the performance of the model becomes poorer. The reason behind this is the combination algorithm tries to fix the bad closed captions by replacing with the decoding transcriptions if two ASRs agree(has exactly the same decoding results).

3. Utilize a randomly different random training set.
We select a different full training set, and then, we used the baseline model 2 to recognize the new training set. The closed captions of the new training set and their corresponding PMER and AWD are used for training a new acoustic model. The new model is evaluated and written in row 6. It shows better performance by reducing WER as well as PER by 0.1. Again, we do another iteration, select a new training set randomly, and evaluate the new model(row 7). Unfortunately, the performance becomes worse compared to baseline 2. There is a luck factor in this technique. If we choose a very good training set in the next iteration, we may obtain a better performance model. Otherwise, the error rate can get worse as shown in row 7.

Table 5.2: WER and PER of baseline versus proposed model

No.	model name	WER(%)	PER(%)
1	baseline 1	41.2	33.8
2	baseline 2	40.5	33.3
3	proposed average	41	33.7
4	proposed combination 1	40.5	33.2
5	proposed combination 2	41.6	33.7
6	different data set 1	40.4	33.2
7	different data set 2	40.7	33.6

5.3 Execution time

Table 5.3 shows the execution time of each iteration. The most demanding process is to train an acoustic model. Furthermore, the second most demanding process is to recognize the full training set and select a good subset of data from it. This is time consuming due to a massive volume of training data(around 200 hours of TV broadcasts). In overall, the execution time of each iteration roughly takes 43 hours.

Table 5.3: WER and PER of baseline versus proposed model

Itera -tion	Train AM	Recognize & data selection	Calculate WER	Decoding graph with LM.200	Decoding graph with LM.7weeks+subs	Total
0	21h	16h	1h	6m	5h	43h
1	24h	14h	1h	5m	3h	43h

5.4 Discussion

This chapter provides the results of experiments of baseline and proposed approaches. In this section, we give a summary of our experiments and offer possible explanations for the obtained results.

In baseline model, we showed a chart of PMER(see 5.2) produced by four iteration baseline models. The chart shows that the next iteration obtains more segments with zero PMER and lower PMER. The value of PMER decreases slightly in each iteration until it converges in the last iteration(iteration 2 and 3). The reason behind this is the data selection method has converged in Baseline 2. Furthermore, we compared the selected segments for training Baseline 2 and 3. They have the same subset of selected segments/utterances(with slightly reduced PMER values in Baseline 3) which explains why they have almost the same PMER.

In addition to comparing PMER in the cumulative chart, we also evaluate each baseline model by computing word error rate of the development set. We did four iterations(baseline 0, 1, 2, and 3). The WER of the baseline model keeps decreasing in each iteration. In other word, we has shown that the data selection technique(inspired from [17]) works well in our data. Model baseline 2 and 3 have the same WER which denotes that the data selection has converged. Moreover, we experimented by modifying the baseline technique.

The first modification is we tried to use the decoding result as the training set for the next iteration. The result is not as good as using our original technique(using closed captions as the training set). The intuition of the bad performance is the PER of baseline 0 is 35.8%; i.e. around 1 from 3 phones are incorrectly recognized. Therefore, the decoding result may be more terrible than the real transcriptions or even more terrible than the closed captions. The second modification is we tried to sort the segments by WMER in place of PMER. The result is worse than the baseline method. This is because we train an acoustic model which predicts sub phones(senone) from the acoustic inputs. Therefore, sorting by PMER is more likely to produce a better acoustic model(and of course better ASR because we use the same language model and lexicon) than sorting by WMER.

Beside the baseline model, we experimented with three different proposed models. The idea behind the proposed models is by combining three different ASRs(the same acoustic model but different language models). The first proposed model is by averaging PMER and AWD and do data selection the same as the baseline method. The result is slightly worse than the baseline. The second proposed model is by combining the recognition of three ASRs by the algorithm proposed in Listing 3.1. We obtained slightly better PER and exactly the same value of WER compared to Baseline 2. Thus, we continued the iteration which resulted in the bad performance model. The reasoning is our combination algorithm tried to fix the closed captions by replacing with the decoding results if two ASRs agree. We hypothesize that our three ASRs are not different because we only utilize different language models(built with somewhat the same text corpus) and the same acoustic model. Therefore, if one ASR makes a mistake, the other probably makes the same mistake. The last proposed model is to recognize randomly different training set for each iteration and do data selection from the training set. The first iteration shows better model than Baseline 2. However, the second iteration performs slightly worse than Baseline 2. We believe there is a luck factor in selecting the random training set. If the randomly selected set is well chosen, we obtain a better model; otherwise, the model will perform poorer.

In conclusion, we have experimented with the baseline model and we also proposed some methods. However, our proposed method does not work as well as we expected. The main reason behind it is we do not use and combine highly different automatic speech recognitions. Instead, we used the same acoustic model and three different language models, which are built from somewhat the same corpus. Therefore, it is necesarry to experiment and investigate more the combination of totally different acoustic model(trained on different architecture, different acoustic input, etc) and totally different language model(different text corpus, different parameters, etc). Due to the limited time constraint of the internship(the execution time of one iteration is long, see 5.3), we do not have much opportunity to experiment further.

Chapter 6

Conclusion and future works

6.1 Conclusion

In this internship, we explored two

6.2 Future works

6.2.1 Short term

6.2.2 Long term

Training an automatic speech recognition requires a large amount of training data. However, manual transcription is expensive in term of time and manpower. There are unlimited supply of audio data in the internet, TV, and radio. Mostly, they have no corresponding exact transcriptions(closed captions) or even no transcription at all. Therefore, we experimented with the TV broadcasts with closed captions(not the same as the detailed transcriptions). Closed captioning is still time-consuming and costly. Some data, such as TV broadcasts, are usually closed-captioned, but other rich sources of data, such as: audio data from internet, are not closed captioned. These data are usually in massive volume and richer in variety and content.

We hope that in the future we can utilize the data without transcriptions for training a high performing automatic speech recognition. ......

Bibliography

[1] Joakim Anden and Stephane Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, aug 2014.

[2] Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 5(2):179–190, February 1983.

[3] Claude Barras, Edouard Geoffrois, Zhibiao Wu, and Mark Liberman. Transcriber: Development and use of a tool for assisting speech corpora production. *Speech Communication*, 33(1-2):5–22, jan 2001.

[4] Herve A. Bourlard and Nelson Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[5] H.Y. Chan and P. Woodland. Improving broadcast news transcription by lightly supervised discriminative training. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE.

[6] George E. Dahl, Dong Yu, Li Deng, and Alex Acero. Large vocabulary continuous speech recognition with context-dependent DBN-HMMS. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, may 2011.

[7] Hynek Hermansky and Sangita Sharma. Temporal patterns (traps) in asr of noisy speech. In *in Proc. ICASSP*, pages 289–292, 1999.

[8] Navdeep Jaitly, Patrick Nguyen, Andrew Senior, and Vincent Vanhoucke. Application of pretrained deep neural networks to large vocabulary speech recognition. In *Proceedings of Interspeech 2012*, 2012.

[9] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.

[10] Thomas Kemp and Alex Waibel. Unsupervised training of a speech recognizer: Recent experiments. In *in Proc. EUROSPEECH*, pages 2725–2728.

[11] Dietrich Klakow and Jochen Peters. Testing the correlation of word error rate and perplexity. *Speech Commun.*, 38(1):19–28, September 2002.

[12] J. Klann and P. Szolovits. An intelligent listening framework for capturing encounter notes from a doctor-patient dialog. In *2008 IEEE International Conference on Bioinformatics and Biomeidcine Workshops*. IEEE, nov 2008.

[13] Reinhard Kneser and Hermann Ney. Improved clustering techniques for class-based statistical language modelling. In *Third European Conference on Speech Communication and Technology, EUROSPEECH*, September 1993.

[14] Karen Kukich. Technique for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, dec 1992.

[15] Lori Lamel, Jean-Luc Gauvain, and G Adda. Lightly supervised and unsupervised acoustic model training. 16:115–129, 08 2002.

[16] P. Lanchantin, M. J. F. Gales, P. Karanasou, X. Liu, Y. Qian, L. Wang, P.C. Woodland, and C. Zhang. The development of the cambridge university alignment systems for the multi-genre broadcast challenge. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, dec 2015.

[17] P. Lanchantin, Mark J.F. Gales, Penny Karanasou, X. Liu, Y. Qian, L. Wang, P.C. Woodland, and C. Zhang. Selection of multi-genre broadcast data for the training of automatic speech recognition systems. In *Interspeech 2016*. ISCA, sep 2016.

[18] Benjamin Lecouteux. Imperfect transcript driven speech recognition. In *In: Proceedings of InterSpeech?06. Pitsburg*, 2006.

[19] Andrew L. Maas, Peng Qi, Ziang Xie, Awni Y. Hannun, Christopher T. Lengerich, Daniel Jurafsky, and Andrew Y. Ng. Building dnn acoustic models for large vocabulary speech recognition, 2014.

[20] L. Mathias, G. Yegnanarayanan, and J. Fritsch. Discriminative training of acoustic models applied to domains with unreliable transcripts. In *Proceedings. (ICASSP 05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* IEEE.

[21] Nima Mesgarani, Shihab Shamma, and et al. Speech discrimination based on multiscale spectro-temporal modulations. In *IN PROC. ICASSP*, pages 601–604, 2004.

[22] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer Berlin Heidelberg, 2008.

[23] H. Murveit, J. Butzberger, V. Digalakis, and M. Weintraub. Large-vocabulary dictation using sri's decipher speech recognition system: progressive search techniques. In *IEEE International Conference on Acoustics Speech and Signal Processing*. IEEE, 1993.

[24] Ibrahim Patel and Y. Srinivas Rao. Realtime speech processing for automated cue-generation. In *2010 INTERNATIONAL CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING TECHNOLOGIES*. IEEE, oct 2010.

[25] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *INTERSPEECH*, 2015.

[26] 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*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[27] Arik Poznanski and Lior Wolf. CNN-n-gram for HandwritingWord recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016.

[28] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco. Connectionist probability estimators in HMM speech recognition. volume 2, pages 161–174. Institute of Electrical and Electronics Engineers (IEEE), jan 1994.

[29] Frank Rosenblatt. Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, mar 1960.

[30] Hasim Sak, Andrew Senior, and Franoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014.

[31] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

[32] Silicon Intelligence. Acoustic model. [Online; accessed July 25, 2017].

[33] Andreas Stolcke. Srilm - an extensible language modeling toolkit. pages 901–904, 2002.

[34] Andreas Stolcke, Wen Wang, Dimitra Vergyri, Venkata Ramana, Rao Gadde, and Jing Zheng. An efficient repair procedure for quick transcriptions. In *in Proc. ICSLP*, pages 1961–1964, 2000.

[35] Grid5000 Team. Grid5000:home. https://www.grid5000.fr/mediawiki/index.php/Grid5000:Ho cited August 2017.

[36] Samuel Thomas, Sriram Ganapathy, and Hynek Hermansky. Phoneme recognition using spectral envelope and modulation frequency features. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, apr 2009.

[37] Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Readings in speech recognition. chapter Phoneme Recognition Using Time-delay Neural Networks, pages 393–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[38] P. C. Woodland, X. Liu, Y. Qian, C. Zhang, M. J. F. Gales, P. Karanasou, P. Lanchantin, and L. Wang. Cambridge university transcription systems for the multi-genre broadcast challenge. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, dec 2015.

[39] George Zavaliagkos and Thomas Colthurst. Utilizing untranscribed training data to improve performance. 1998.