Data Selection For The Training Of Deep Neural Network In The Framework of Automatic Speech Recognition

Juan Karsten

Inria Loria Nancy

Universite de Lorraine

A thesis submitted for the degree of

Master

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

This internship used 200 hours of speech data and their corresponding closed captions obtained from BBC. However, only closed captions are provided by the challenge. In other word, the closed captions are not exactly the same as what audio says. Utilizing the whole dataset generates a bad acoustic model owing to imperfect closed captions. Thus, the objective of the internship is to develop a data selection method to obtain a high performance automatic speech recognition(ASR). Simply put, the ASR must reduce word error rate as small as possible.

A baseline system was developed from a 200 hours random subset of whole training set(train.200). Firstly, a deep neural network was trained on the 100 hours subset of data from train.200 and recognized train.200 to calculate phone matched error rate(PMER), word matched error rate(PMER), and average word duration(AWD). The audio segments were sorted and selected based on PMER and AWD for the next iteration. According to our experiments, baseline system shows decreasing trend of word error rate as well as phone matched error rate threshold. Moreover, I proposed a new algorithm which combines three ASR systems by varying language models. These novel approaches are compared with the baseline system to assess whether it achieves better performance.

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. One of of the practical applications of speech recognition is automatic transcription on television broadcasts.

An acoustic 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.

To train an ASR, one needs a massive amount of training audio and their exact transcriptions. Many audio data are available in the internet; however, many of them only possess few corresponding transcriptions or no transcription at all. Transcribing audio manually is labor intensive and also time consuming. 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. Therefore, selecting ”good” data(audio data with very close captions) might be a solution to achieve a high performance ASR.

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.

1.2 About the project

Kaldi framework helps to train acoustic models with deep neural network in this internship. The goals of the internship are as following:

1. Learn the state-of-the-art automatic speech recognition. I learn how to train and employ deep neural network for recognizing audio as well as build N-Gram language models. Moreover, I utilized a cluster computing server to train the acoustic models with massive amount of data.

2. Select the subset of data which are close to the real transcription and evaluate how good our data selection is. The good data must have the closed captions which are close to the real transcription.

3. Propose new solutions which improve the current state-of-the-art.

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.

1.3 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.

1.4 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

Review of literature

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 [2]. ASR has been applied in many applications and domains, for example: automatic closed captioning for hearing-disabled persons [8], taking notes of conversations between doctors and patients [4], automatic closed captioning TV broadcasts [15], 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 [12]. N-gram language model has been applied in many applications, such as: speech recognition[15], handwriting recognition [10], as well as spelling correction [6].

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 [5] 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 [1] [3]:

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

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:

- $Q = q_1q_2...q_N$ 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.

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

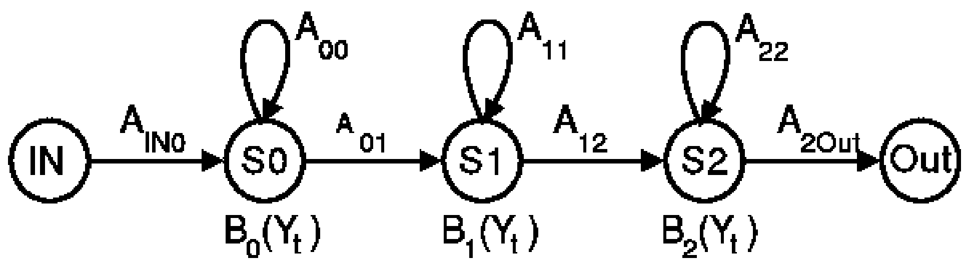
2.1.3.3 Applying HMM to acoustic model

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.

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 [13]



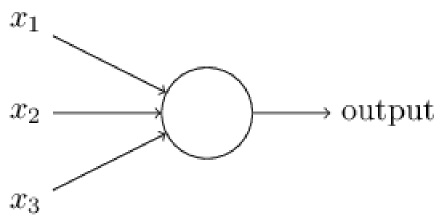
2. Transition matrix

   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. Acoustic feature vector is a 39-dimension which is generated based on Mel-frequency cepstral coefficient(MFCC). Each observation is a slice of 10 milliseconds audio. For one second audio, there will be 100 acoustic feature vectors where each vector has 39 features.

Figure 2.2: A perceptron



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

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 [9]. 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 [11]. 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
$$a(x) = \frac{1}{1 + e^{-x}} \tag{2.11}$$

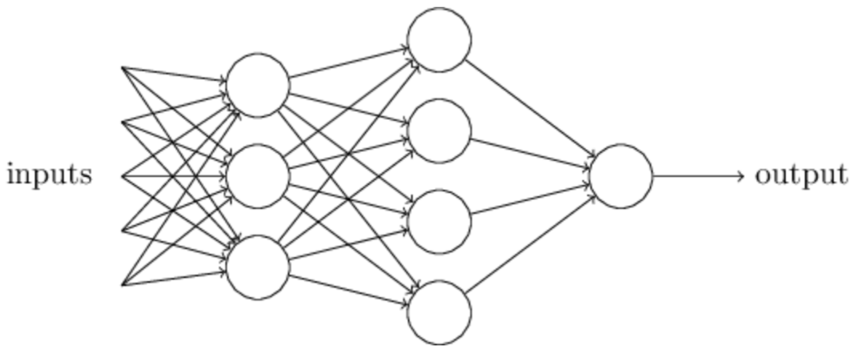
2. Rectified linear unit(relu)
$$a(x) = max(0, x) \tag{2.12}$$

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



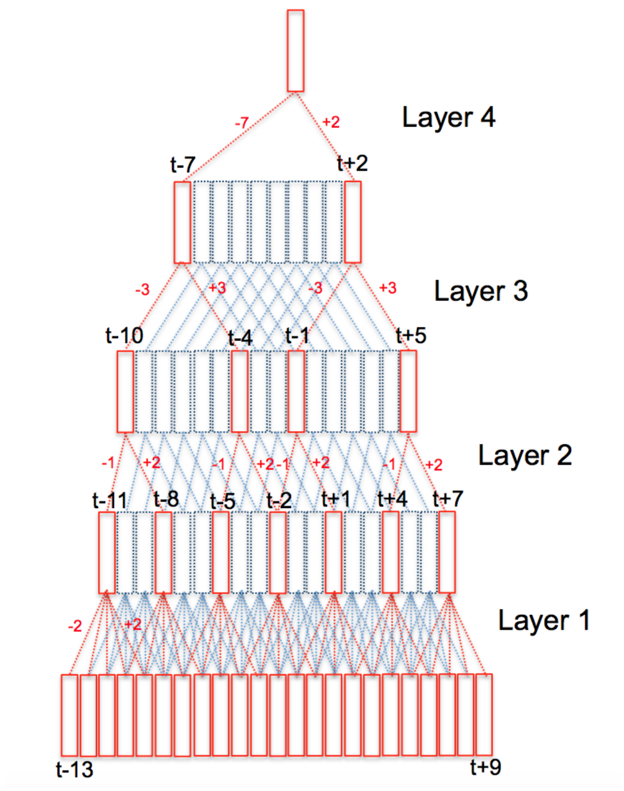
2.1.4.2 Time delay neural network(TDNN)

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.

Time delay neural network(TDNN) was first proposed by Waibel et al in 1990[14]. TDNN is suitable to model long term temporal relationship within the range of N delays in short term speech features(like acoustic inputs in the speech recognition problem). The acoustic feature is called short term feature because it is an MFCC feature vector generated from a slice of 10 milliseconds acoustic input.

To learn sequential inputs, TDNN must learn sequence of cues. Hence, Waibel et al introduced a time delay D_1 to D_N to learn long term temporal dependencies. The input feature vector is 40 dimensional MFCC feature vector. Figure 2.4 shows the

Figure 2.4: An example of a TDNN architecture. [9]



TDNN architecture. The input layer receives 4 delays($N = 4$), i.e. $\{t-2, t-1, t, t+1, t+2\}$. Therefore, each unit has 200 weights(5×40) from input layer. The next layer(hidden layer 1) receives delay from $t-1$ through $t+2$. The higher layer will receive wider temporal contexts compared to the lower layers. For example, the final layer receives inputs which are time delay from $t-7$ through $t+2$ context.

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.

Training a TDNN is still time consuming. Peddinti et al found that there existed large overlaps between input contexts at neighboring time steps; thus, they proposed subsampling [9]. Figure 2.4 shows the procedure of subsampling. It subsamples $t-1$ and $t+2$ time(without including t and $t+1$ time samples) in the second layer, subsamples $t-3$ and $t+3$ in the third layer, and subsamples $t-7$ and $t+2$ in the final layer. It applies wider splice when going to higher layers in the networks. The reason is we force the higher layer to learn wider temporal contexts. The advantage of TDNN is it can reduce computational time and model size.

2.2 Data selection: Lightly supervised approach

One of the main challenge of speech recognition is to reduce development cost when adapting the speech recognition to other task or language. The biggest development cost is to transcribe the audio with exact transcriptions. To generate a ”good” speech recognition, one needs a massive amount of training audio and their exact transcriptions. However, transcribing audio is labor intensive and also time consuming. The internet has unlimited volume of audio data; but, they usually have closed captions only. 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.

Unsupervised acoustic model training was proposed by Zavaliagkos and Colthurst in 1998 [16]. 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 acoustic model.

Lightly supervised approach was proposed in 2002 by Lamel et al [7]. 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. The closed captions are usually produced manually in real time and many are already provided. 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 and deleted.

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; thus, it provides less supervision. We can interpolate these data with the closed captions to build a more general language model.

Chapter 3

Methodology

This chapter mainly discusses the methodology. First, we explain the data which we used. After that, the baseline method is elaborated. Lastly, we propose new data selection algorithms.

3.1 Data

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

- The audio files are taken from TV broadcasts over period of 1 April 2008 - 19 may 2008(7 weeks). Total duration of audio is 1,600 hours.
- TV brodcast transcriptions(7 weeks 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.
- 640 million words of TV subtitles are given over the period of 1979-2013. In addition to that, the 640M subtitles are filtered to avoid overlap with the 7 weeks closed captions.

We prepared two data sets: training set and development set. The training dataset has a total duration of 1600 hours of audio. To evaluate our model, we have a development set. To accelerate acoustic model training, we selected roughly 200 hours of data from the training 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.

How to select 200 hours(train.200) and 100 hours(train.100) subset of data:

1. The files in train.full were sorted by their size.
2. Select the biggest file out of 8 files.
3. Write the selected data into 200 training set file.

Because the duration of train.full is more or less 1600 hours, 200 hours of data were obtained by using this random selection.

No	Data set	#shows	Duration(h)	Aligned speech (h)
1	full training set	2,193	1,580	1,197
2	dev.short	12	8	6
3	train.200	274	193	149

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.

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.

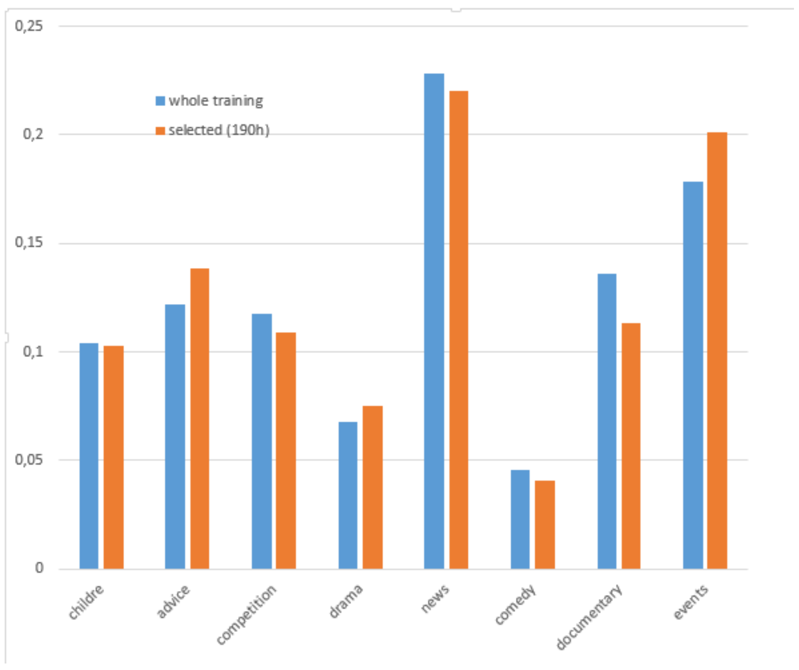
Table 3.1: Statistics of train.200 per genre

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

3.2 Baseline model

todo insert overview here

Figure 3.1: Bar chart of each genre from all training data and 200 hours training data



3.2.1 Error rate and average word duration

The standard evaluation of speech recognition is word error rate(WER). WER compares and counts the number of edits between the real transcription and the decoding result(the hypothesis transcription which is produced by the speech recognizer). The first step to calculate WER is by computing the number of edits with minimum edit distance algorithm. The algorithm produces the number of substitution, insertion, and deletion. Then, word error rate can be calculated as following:

$$\text{Word error rate} = 100 \times \frac{\text{Insertion} + \text{Substitution} + \text{Deletion}}{\text{Total words in correct transcript}} \tag{3.1}$$

Phone error rate(PER) is a metric which compares the hypothesis transcriptions and the real transcriptions in phone level. Instead of comparing words(calculating WER), we need to compare phone in data selection problem. The main reason is we want to improve the acoustic model which outputs a sequence of hypothesized phones.

When selecting the subset of data, we compared the closed captions and the output of speech recognition. To avoid confusion with WER and PER(error rate between the hypothesized transcriptions and the real transcriptions), we used term word matched error rate(WMER) and phone matched error rate(PMER), which is an error rate between the closed captions and the decoding results in word and phone level respectively.

One of the standard framework of running minimum edit distance and calculating error rate is sclite. sclite is able to do alignment and calculate error rate. Moreover, it provides some statistic summary, e.g. total insertion, total deletion, overall error rate, etc, and calculate error rate for each speaker. It also provides insertion, deletion, and substitution for each segment which later can be used to calculate WMER and PMER for each segment.

3.2.2 Our baseline method

Our baseline was inspired from the Cambridge method. In overall, our baseline technique works as following:

1. Train an acoustic model(AM.100-v0) on 100 hours of training data(train.100)

2. Build a language model(e.g. LM.200.1e-9) from 200 hour transcriptions. This language model was built from closed captions of train.200. Then, the language model is pruned by 10^{-9}. In other word, if the probabilities of N-Gram word sequence is less than 10^{-9}, they will be excluded. We will elaborate the language model in the next section.

3. By utilizing the acoustic model(AM.100-v0), the language model(LM.200.1e-9), and a lexicon(e.g. Lexicon.200), recognize a development set(dev.short, 8 hours) to know the overall PER and WER. Our lexicon is from Combilex British English lexicon. Lexicon.200 is a subset of Combilex lexicon where only words which exist in the 200-hour training set(train.200) are used.

4. By using the same acoustic model(AM.100-v0), language model, and lexicon, recognize the 200 hour training set. From the recognition process, calculate PMER(phone matched error rate), WMER(word matched error rate), and AWD(average word duration) of each segment.

5. Sort segments according to its PMER. Choose 100 hour segments with top PMER value if the segments has AWD with range between 0.165 and 0.66. This range follows the AWD range from the Cambridge method.

6. Train a new acoustic model(AM.100-v1) with the 100 hour data(100-v1).

7. Utilizing AM.100-v1, LM.200, and Lexicon.200, recognize 200 hours data to get 100 hour data(100-v2). Retrain an acoustic model(AM.100v2) by using 100-v2 and re-recognize the 200 hour data. Reiterate the process of selecting(100-v3 and so on) and training data(AM.100v3 and so on) until no further improvement is found.

To train an acoustic model, firstly a gaussian mixture model(GMM) HMM was trained. **todo continue this**

3.2.3 Language model

This internship experimented with three different kinds of language model. All language models are based on four gram language model.

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}. LM.200 was used for the baseline speech recognizer to select the "good" subset of data.

Table 3.2: 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 7 weeks transcription 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 proposed new 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

One of the tools to build language model is 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.

3.2.4 Acoustic Model

3.3 Proposed new algorithm

We proposed three new algorithms which combined three different automatic speech recognizers(ASR) to select data.

1. Average PMER and AWD of three speech recognizers and select subset of data the same as baseline algorithm.

2. Use the baseline algorithm, but different random 200 hours of training set in each iteration.

3. Combine three different speech recognizers and select data with the combination and selection algorithm which we proposed. This algorithm will be explained in this section.

Before elaborating the proposed combination and selection algorithm, three different speech recognizers(ASR) are explained first.

- ASR1(the most constrained): AM1, LM.200, and Lexicon.200

- ASR2: AM1, LM.7weeks+subtitle, and Lexicon.7weeks+subtitle
- ASR3(the least constrained): AM1, LM.7weeks+subtitle, and Lexicon.7weeks+subtitle

The different combination can be obtained by varying acoustic models and language models. The following is the algorithm how three different ASR are combined and selected:

```
1. for each segment:
2.     If one ASR has zero PMER or zero WMER
3.         choose the original segment
4.     Else If two segments have the same phone sequence
5.         choose the decode segment
6.     Else
7.         Reject segment which is not in 0.166¡AWD¡0.65 and 0.03¡APD¡0.2 range
8.         Sort by PMER. Choose top segments with the lowest PMER. The original
   segments will be chosen
```

3.3.1 Evaluating speech recognizers

1. Create a decoding graph from an acoustic model, a language model, and a lexicon. The usage of language model and lexicon must be the same for all acoustic models. For example in this internship: use LM.7weeks+subtitles.limited.1e-9 and Lexicon.7weeks+subtitles as the language model and the lexicon respectively.
2. Decode audio of development set(dev.short) with the graph and compare the decoding result with manually annotated transcription(transcript_human).
3. Calculate WER(word error rate) and PER(phone error rate).

3.4 Baseline model

3.4.1 Acoustic model AM.200 and Language Model Version 1 (LM.200.1e-9)

Calculating PMER and WMER for each segment by utilizing AM.200 and LM.200.1e-9:

1. GMM(GMM.200): GMM is trained by using the 200 hours data

2. TDNN(TDNN.200): the result of GMM training(GMM.200) is utilized to build a TDNN(time delay neural network).

3. TDNN graph creation
   The language model used is the LM.200.1e-9 and the acoustic model is TDNN.200; furthermore, the dictionary used(Lexicon.200) is the modification of the library provided by MGB(Lexicon.MGB). Words in Lexicon.MGB which do not exist in word.200 are removed from Lexicon.200.

4. TDNN 200 hours of training data recognition
   We need to recognize 200 hours data to calculate PMER, WMER, and AWD. This data will be used for data selection of the next iteration.

5. Sclite score PMER and WMER
   Use the lattice and graph(produced at step 5 and 3 respectively) to calculate overall PMER and WMER. However, the decoding program does not provide PMER, WMER, and AWD for each segment. Fortunately, the program generates prf file which contains number of correction, insertion, deletion, and substitution for each segment. Two prf files exist after decoding:(ctm_words.filt.filt.prf contains word match error rate and ctm_phones.filt.filt.prf contains phone match error rate). From there, we are able to calculate PMER, WMER, and AWD of each segments. For the development set, we only need to know the overall PER and WER. In contrast, training set recognition needs PMER, WMER, and AWD for each segment.

version AM0

No.	Description	Result	execution time	extra info
1	GMM.100		3h 30m	20 hosts
2	TDNN.100 training: LM.200.1e-9, GMM.100, Lexicon.200		17h 7m	3 GPU hosts
3	Create a graph from TDNN.200, LM.200.1e-9, Lexicon.200		6m	1 host
4	Create a graph from TDNN.200, LM.7weeks+subtitles.1e-9, Lexicon.200		4h 49m	1 host
5	TDNN.100 recognized train.200		16h 6m	20 hosts
6	TDNN.100 & LM.7weeks+subtitles.1e-9 recognized dev.short	WER 43.9% PER 35.8%	45m	20 hosts

version AM1

No.	Description	Result	execution time	extra info
1	GMM.100		4h 7m	20 hosts
2	TDNN.100 training: LM.200.1e-9, GMM.100, Lexicon.200		20h 1m	3 GPU hosts
3	Create a graph from TDNN.200, LM.200.1e-9, Lexicon.200		5m	1 host
4	Create a graph from TDNN.200, LM.7weeks+subtitles.1e-9, Lexicon.200		3h 7m	1 host
5	TDNN.100 recognized train.200		14h 18m	20 hosts
6	TDNN.100 & LM.7weeks+subtitles.1e-9 recognized dev.short	WER 41.2% PER 33.8%	46m	20 hosts

version AM2

No.	Description	Result	execution time	extra info
1	GMM.100		3h 3m	20 hosts
2	TDNN.100 training: LM.200.1e-9, GMM.100, Lexicon.200		23h 58m	3 GPU hosts
3	Create a graph from TDNN.200, LM.200.1e-9, Lexicon.200		5m	1 host
4	Create a graph from TDNN.200, LM.7weeks+subtitles.1e-9, Lexicon.200		3h 13m	1 host
5	TDNN.100 recognized train.200		15h 51m	20 hosts
6	TDNN.100 & LM.7weeks+subtitles.1e-9 recognized dev.short	WER 40.5% PER 33.3%	36m	20 hosts

version AM2

No.	Description	Result	execution time	extra info
1	GMM.100		3h 3m	20 hosts
2	TDNN.100 training: LM.200.1e-9, GMM.100, Lexicon.200		25h 27m	3 GPU hosts
3	Create a graph from TDNN.200, LM.200.1e-9, Lexicon.200		7m	1 host
4	Create a graph from TDNN.200, LM.7weeks+subtitles.1e-9, Lexicon.200		4h 49m	1 host
5	Create a graph from TDNN.200, LM.genres.1e-9, Lexicon.200		2h 1m each genre	1 host
6	TDNN.100 recognized train.200		15h 52m	20 hosts
7	TDNN.100 & LM.7weeks+subtitles.1e-9 recognized dev.short	WER 40.5% PER 33.3%	38m	20 hosts

3.4.2 Acoustic model(AM.200) and Language Model version 3(LM.genres)

3.4.3 Acoustic model AM.100-v1 and Language Model version 1 (LM.200.1e-9)

3.4.3.1 100 hours selection(100-v1)

How to select 100 hours subset of data for baseline model v1: Section 3.4.1 calculates PMER, WMER, and AWD for each segments. Consequently, we can extract segments(of 200 hours transcriptions) which satisfy $0.165 \leq AWD \leq 0.66$. In the other word, the segments which does not satisfy are removed. Then, segments are sorted based on their PMER in ascending. We can deduce PMER threshold and its total duration by sorting PMER as presented in the following table:

PMER	Total duration	Percentage of total segment duration
0.0	14.83 h	10%
5	29.66 h	20%
10	44.49 h	30%
16	59.32 h	40%
24	74.15 h	50%
33	88.98 h	60%
47	103.81 h	70%
82	118.64 h	80%

The Table 3.4.3.1 shows that 60% of data has 33% PMER with total duration of 88.98 hours of audio. From the table, we conclude that threshold 33 is selected as the threshold to select data for the next iteration.

WMER	Total duration	Percentage of total segment duration
0.0	14.83 h	10%
8	29.66 h	20%
15	44.49 h	30%
23	59.32 h	40%
32	74.15 h	50%
43	88.98 h	60%
58	103.81 h	70%
98	118.64 h	80%

3.4.3.2 Training Acoustic model AM.100-v1

3.4.4 The Second Acoustic model(AM.100-v2) and Language Model version 1 (LM.200.1e-9)

3.5 Comparison

Comparison between different speech recognizer:

No.	Acoustic model	Language Model	PER	WER
1	TDNN.100	LM.7weeks+subtitles.limited.1e-9 Lexicon.7weeks+subtitles	35.8	43.9
2	TDNN.100-v1_decode	LM.7weeks+subtitles.limited.1e-9 Lexicon.7weeks+subtitles	34.3	42.3
3	TDNN.100-v1	LM.7weeks+subtitles.limited.1e-9 Lexicon.7weeks+subtitles	33.8	41.2
4	TDNN.100-v2	LM.7weeks+subtitles.limited.1e-9 Lexicon.7weeks+subtitles	33.3	40.5
5	TDNN.100-v3	LM.7weeks+subtitles.limited.1e-9 Lexicon.7weeks+subtitles	33.3	40.5

Figure 3.2: Average word duration

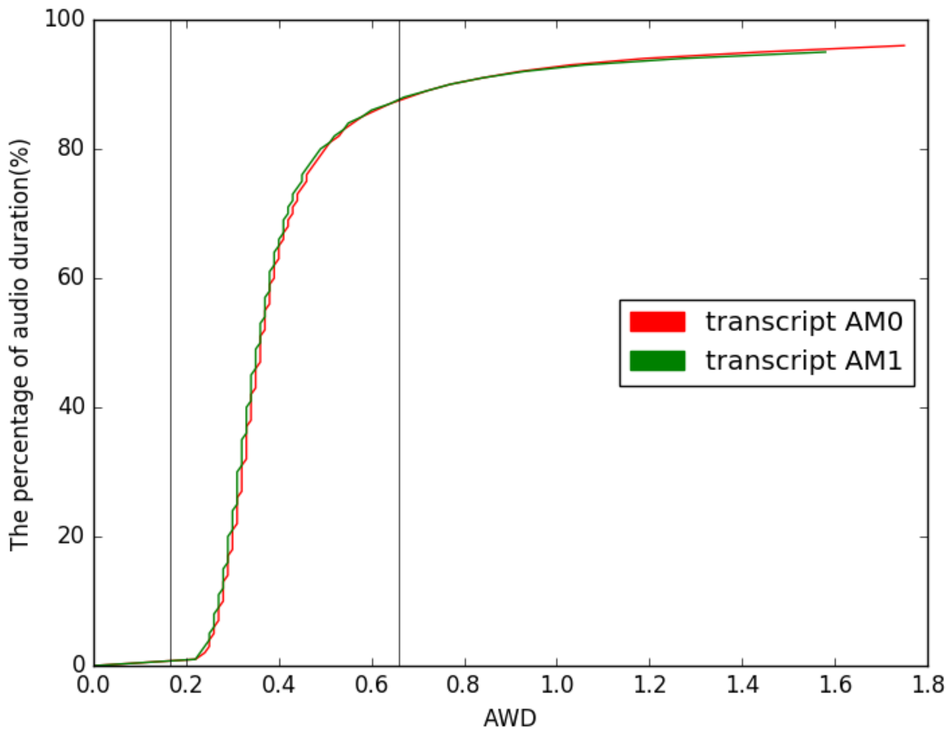


Figure 3.3: Phone match error rate

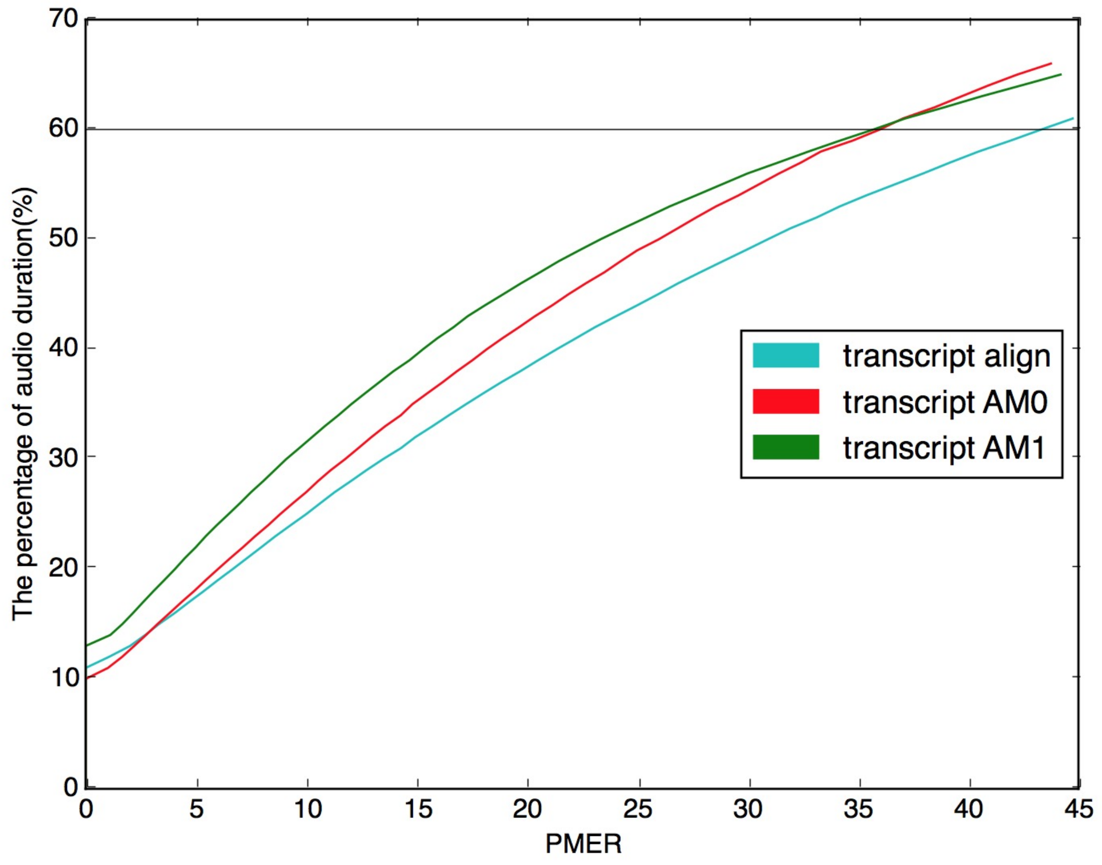
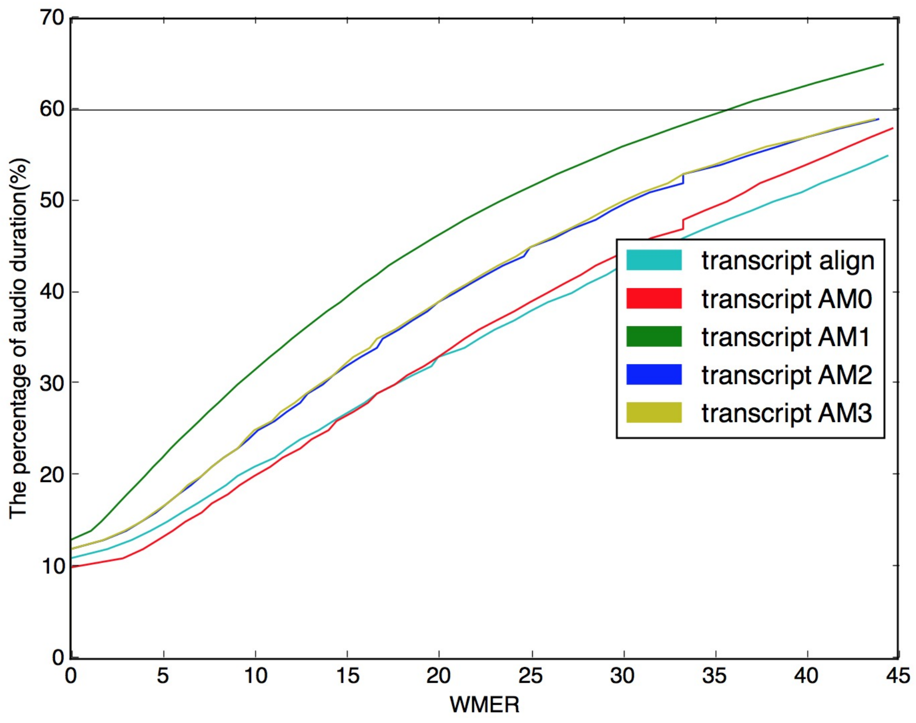


Figure 3.4: Word match error rate



Chapter 4

Experimental Setup

4.1 Tools

4.2 Code pipeline

4.3 Execution

Chapter 5

Sample Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Appendix A

Sample Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Appendix B

Sample Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Bibliography

[1] 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.

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

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

[4] 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.

[5] 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.

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

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

[8] 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.

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

[10] 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.

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

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

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

[14] 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.

[15] 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.

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