NTNU
Norwegian University of
Science and Technology

Speech Recognition by Human and Machine

Vegard Gulaker

Master of Science in Engineering Cybernetics
Submission date: February 2010
Supervisor: Sverre Hendseth, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

The motivation for this thesis is the possibility to divide the task of speech recognition between a computer and a deaf person. The computer is able to do the signal processing, feature-detection and analyzation, but the interpretation of the output, which is difficult for a computer, can be done by the human after sufficient training.

The ambition of the thesis is to explore the concept, and investigate if a divide as the one mentioned above is possible.

*A literature study on automatic speech recognition by computers, with especial care given to signal processing, feature-detection and algorithms will be conducted.

*A set of features must be decided, which can be detected by the computer and gives a foundation to recognize the speech.

*Experiments can be conducted on existing programs, application programmers interface or toolkits to evaluate the learning benefits and possibility as a starting point for an application.

Assignment given: 27. September 2009
Supervisor: Sverre Hendseth, ITK

Abstract

Several feature extraction techniques, algorithms and toolkits are researched to investigate how speech recognition is performed.

Spectrograms were found to be the simplest feature extraction techniques for visual representation of speech, and are explored and experimented with to see how phonemes are recognized.

Hidden Markov models were found to be the best algorithms used for speech recognition. Hidden Markov model toolkit and Center for Spoken Language Understanding Toolkit, which are based on hidden Markov models, were not found to be suitable for the intentions of the thesis.

Contents

Acknowledgements

I would like to thank my supervisor Sverre Hendseth for his encouragement and guidance while writing this thesis. Even though speech recognition has never been done at the Department of Cybernetics, he listened to my ideas and gave good feedback.

My girlfriend Siri Holthe Mathisen also deserves a lot of thanks and love, for getting me up in the morning and helping me through a tough part of my life.

Lastly my family deserves to be thanked, for always being there for me with their support.

List of Figures

Chapter 1

Introduction

1.1 Overview

A speech recognition system uses material from a lot of various disciplines, e.g., statistical pattern recognition, communication theory, signal processing and mathematics.

The main problem with recognizing speech is the variable nature of speakers. When a word is pronounced even by the same speaker two times, the speech pattern is never the same. Speech is also different for every speaker; women usually speak with a higher frequency than males. A word can be uttered fast, slow or with variable speed and noise from other sources can be heard simultaneously. There is also a problem with dialects and non native speakers whom pronunciation differs severely.

These problems are especially clear when a computer is used for speech recognition. Most automatic speech recognition systems are speaker dependent, where a program learns how the target speaks.

1.2 Problem Description

The motivation for this thesis is the possibility to divide the task of speech recognition between a computer and a deaf person. The computer is able to do the signal processing, feature-detection and analyzation, but the interpretation of the output, which is difficult for a computer, can be done by the human after sufficient training.

The ambition of the thesis is to explore the concept, and investigate if a divide as the one mentioned above is possible.

- A literature study on automatic speech recognition by computers, with especial care given to signal processing, feature-detection and algorithms will be conducted.
- A set of features must be decided, which can be detected by the computer and gives a foundation to recognize the speech.
- Experiments can be conducted on existing programs, application programmers interface or toolkits to evaluate the learning benefits and possibility as a starting point for an application.

1.3 Background

A course about speech recognition has never been a taught at the Department of Engineering Cybernetics. All the algorithms, methods and programs explored in this thesis was not known prior to writing, but some knowledge about signal processing, probability, statistic, advanced calculus and programming helped to understand the algorithms and toolkits.

1.4 Disposition

Chapter 2 will look at how human speech is produced, and the basic building blocks of the English language with emphasis on how phonemes are

produced.

Chapter 3 is a literature study on how the various ways speech can be processed on the computer. Existing methods for feature extraction is explored, and three algorithms used for automatic speech recognition.

Chapter 4 explores various programs and toolkits that can be used to learn or use methods discussed in Chapter 3. Personal experience with the programs and toolkits are given to review how well they are suited for the purpose of this thesis.

Chapter 5 explores how to understand spectrograms. The different spectral clues and example are given, as are the experience with spectrograms and problems encountered while trying to understand them.

Chapter 6 is a discussion looking at the various feature extractions and algorithms explored in Chapter 3. It also discusses future work based on this thesis.

Chapter 2

The Human Speech Process

2.1 How Speech is Produced

An average speaker speaks two to three words per second, which is equivalent to ten to twelve phonemes [8]. The vocabulary consists of 50.000 to 100.000 words.

The mechanics of human speech is given in Figure 2.1. One technical term not present in the figure is the palate, which is the bone and skin separating the mouth cavity and nasal cavity.

A better representation of how the speech mechanism works is given in Figure 2.2.

Speech is produced in the following way [1, 9]:

1. Air is forced from the lungs.

2. If the vocal chords are tense, they vibrate and voiced speech is produced. When the vocal chords are relaxed, there is still a constriction causing the air to become turbulent producing unvoiced speech.

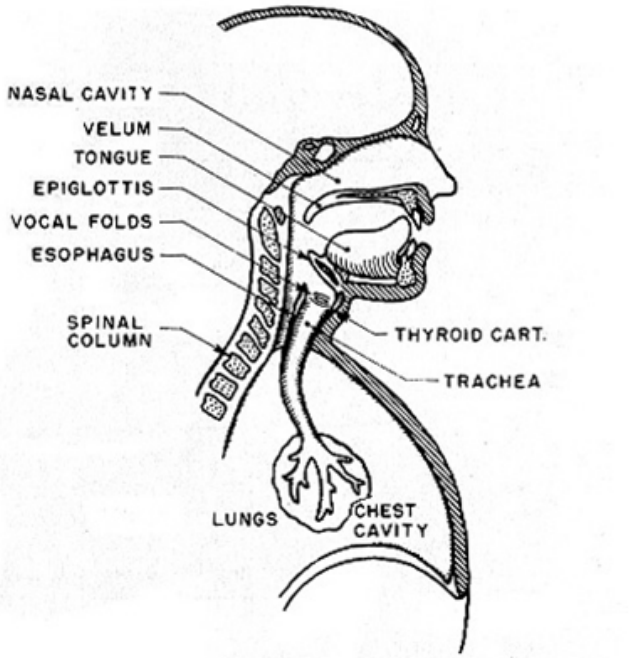

Figure 2.1: The human vocal mechanism, retrieved from [1]

3. Quasi-periodic pulses are modulated in frequency by the pharynx, i.e, throat cavity, mouth cavity and possibly the nasal cavity.

4. Based on the different position of the jaw, tongue, lips and mouth different sounds are produced.

2.2 Languages

A language consists of linguistic units called phonemes, where changing a phoneme in a word will give it another meaning [9]. The changes can be subtle, and a may take a skilled person in the language to recognize. In some Chinese dialects, the pitch is very important to distinguish words. But in English, raised pitch can be used at the end of sentences to indicate a question.

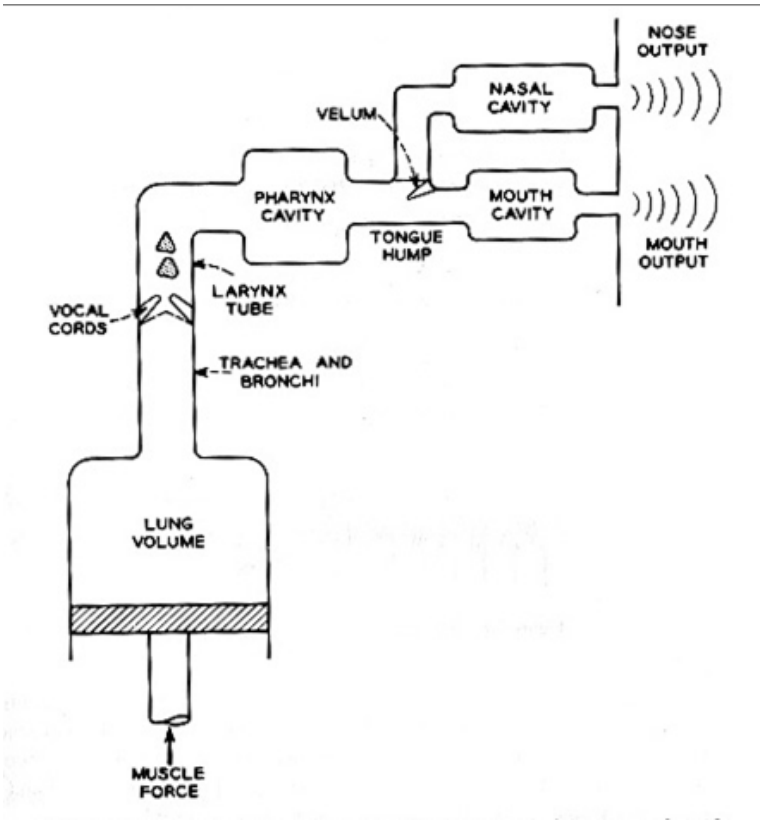

Figure 2.2: Representation of the complete physiological mechanism of speech, retrieved from [1]

2.2.1 Syllables

There are over 10000 different syllables in the English language, but a native speaker uses only 500 of these 80 percent of the time when speaking[8].

A syllable structure is built up of four parts, where two are obligatory [10]. The structure can be seen in Figure 2.3

The rhyme and nucleus, which is the central segment of the syllable, is obligatory. The onset and coda are starting and ending segments of the syllable, respectively.

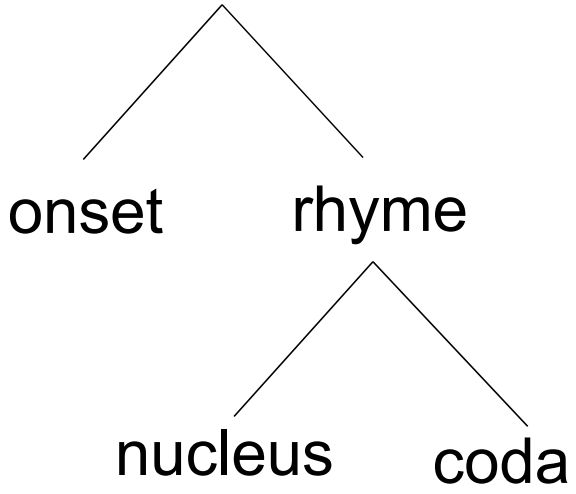

Figure 2.3: The structure of a syllable

2.2.2 Phonetics

The information and terminology regarding phonetics is mainly gathered from "Fundamentals of Speech Recognition" [1] and "Speech Analysis Synthesis and Perception" [9]. Lecture notes from the course "Automatic Speech Recognition" at MIT taught in spring 2003 [11] and the book: "An Introduction to English Phonology" [12] are also used.

A phoneme is a distinct speech sound. Words consist of phonemes which describe accurately how the word is pronounced.

A complete list of phonemes is given in Appendix A, but a classification of different phonemes will be given here:

The following list shows a the definition of where the different sounds are generated.

Definition	**Place where sound is made**
Labio-dental	The upper lips and lower lip
Dental	Tongue behind the teeth
Alveolar	Tongue to the gum ridge
Palatal	Tongue against the hard palate
Velar	Tongue against the soft palate
Glottal	Vocal chords constricted and fixed
Labial	Constricted by the lips

Vowels

There are three different types of vowels depending on where the tongue is position when the sound is uttered; front, mid and back. The vocal tract has a stable position when a vowel is pronounced.

Constriction in mouth	**Tongue position**		
	front	central	back
High	/i/ eve, /ɪ/ It	/ɝ/ bird	/u/ boot, /ʊ/ foot
Medium	/e/ hate, /ɛ/ met	/ʌ/ up	/o/ obey, /ɔ/ all
Low	/æ/ at		/a/ father

The tongue position and constriction can be experienced by pronouncing the words succeeding each phoneme.

Diphthongs

A diphthong is a vowel which transforms to another vowel before the pronunciation is complete. /aɪ/ (my), /aʊ/ (now), /ei/ (day), /oʊ/ (no), /ɔi/ (boy) and /iʊ/ (few) are the diphthongs in the English language.

Approximants

The approximants are very vowel and fricative like. They are split into two groups: glides and semivowels. The glides are dynamic sounds which precede a vowel. The semivowels cause more constriction in the oral channel than vowels.

Place of articulation	**Glides**	**Semivowels**
Palatal	/w/ we	/r/ read
Labial	/j/ you	
Alveolar		/l/ let

Nasal Consonants

There is a total closure is present in the front of the mouth, either by the lips, the tongue and the gum ridge, or the tongue and the palate. The velum is opened, and the nasal tract is the route out for the air. This cause the nostrils to vibrate when pronouncing a nasal consonant.

Place of articulation	
Labial	/m/ me
Alveolar	/n/ no
Palatal/velar	/ŋ/ sing

Fricatives

A fricative can be recognized by the hissing sound made by friction when air leaves the mouth. There are two types of fricatives: voiced and unvoiced. The difference between voiced and unvoiced fricatives is the vibration in the vocal cords when a voiced fricative is pronounced.

Place of articulation	Voiced	Unvoiced
Labio – dental	/v/ vote	/f/ for
Dental	/ð/ then	/θ/ thin
Alveolar	/z/ zoo	/s/ see
Palatal	/ʒ/ azure	/ʃ/ she
Glottal		/h/ he

Stops

A stop is produced by stopping the airflow in the mouth. Depending on the stop, it might be behind the lips, behind the teeth or in the back of the mouth. For voiced, a small vibration can be seen in the waveform before the outlet. The unvoiced have a complete lack of signal during the build up for the outlet.

Place of articulation	Voiced	Unvoiced
Labial	/b/ be	/p/ pay
Alveolar	/d/ day	/t/ to
Paletal/velar	/g/ go	/k/ key

Affricates

Affricates are the combination of a stop and fricative. The two used in English are: /tʃ/ (chew) and /dʒ/ (judge).

2.3 Speechreading

Visual image of the speaker improves speech recognition [13]. Speechreading give a high association between recognizing isolated words and words in sentences, but low to moderate association between phoneme and word recognition.

The visual image of the talker influences how the word is interpreted. The McGurk effect illustrates this. If a syllable /bi/ is heard, while a video of a talker saying the syllable /gi/ is seen, the syllable is perceived as /di/. This effect is also present when there is a male voice with a female face in the video, or if the sound and video are not synchronous.

Chapter 3

Human Speech on the Computer

3.1 Speech Waveform

When a word is spoken, it creates a variation in the air pressure [14]. The human ear is capable of noticing a variation of 0.00002 Pascal. A computer needs to convert an analog signal into a digital to be able to process them. This conversion takes place in three steps [4].

1. Sampling: The signal is made discrete by taking samples at a time interval. The Nyquist rate is the sampling rate which conserves the higher frequencies given by:

$$F_N = 2 * F_{MAX} \tag{3.1}$$

   Where F_N is the sampling rate, and F_{MAX} is the maximum frequency it is possible to distinguish.

2. Quantization: The value which represents the signal by a digital value. It is chosen from a finite set of possible values.

3. Coding: Each discrete value is represented by a b-bit binary sequence. The numbers of bits are the standard for the computer, 8, 16 or 32.

Most of the frequencies for regular speech fall below $3000Hz$ for , which is why the telephone range can be: $200-3400Hz$[15], and still be understandable.

It is possible to classify several characteristics of speech from the waveform [1]. Silence is where no sound is recorded. There are natural silences while talking, e.g., commas and punctuation. There are also natural silences before specific phonemes, such as stops, where the pressure is building in the mouth. Voiced, where the vocal chords are vibrating, and unvoiced, where they are not vibrating is also possible to determine.

3.2 Feature Extraction from Speech

After speech is recorded, there are a lot of possible ways to extract relevant information from it. This section will look at different methods which is used in later algorithms and methods.

3.2.1 Short-Time Fourier Transform

A problem with the normal Fourier transform is that it does not show how the frequency changes over time [16]. Short time Fourier transform cuts the signal up in windows and computes the Fourier transform, usually with the fast Fourier transform, for each window.

Let $x(n)$ be a signal, and $X_n(e^{j\omega_k})$ is the short-time Fourier transform of $x(n)$ evaluated at time n and frequency ω_k. Given a window function, $w(n)$, the short-time Fourier transform can be defined as [17]:

$$X_n(e^{j\omega_k}) = \sum_{m=-\infty}^{\infty} w(n-m)s(m)e^{-j\omega_k m} \tag{3.2}$$

The size of the window, defined as L, influences the function in the following way [1]:

- Large L relative to the signal periodicity gives good frequency resolution but gives a bad time resolution.
- Small L relative to the signal gives poor frequency resolution but gives a good time resolution.

To improve the result of the short-time Fourier transform, the windows can be overlapping [16].

For more information about calculating the reverse of the short-time Fourier transform and sampling rate in correspondence to the windows, "A Unified Approach to Short-Time Fourier Analysis and Synthesis" [17] should be read.

3.2.2 Window functions

To create a spectrogram, different window shapes can be used. Matlab uses Hamming window by default [18].

Praat[2] gives the choices of:

- Gaussian, Square (none, rectangular)
- Hamming (raised sine-squared)
- Bartlett (triangular)
- Welch (parabolic)
- Hanning (sine-squared)

and argues that the Gaussian window is superior because it doesn't show sidelobes in the spectrogram. But the computation for Gaussian window is twice as expensive as the other windows.

Four window functions will be shown here, rectangular for reference, Hamming and Gaussian because those are the best suited for creating spectrograms and triangular window because it is used for computing Mel-frequency cepstral coefficients. All figures are retrieved from Wikipedia after the Matlab code to create them was checked for correctness. The definition of the windows is retrieved from [19].

Rectangular Window

A Rectangular window is given by the formula.

$$w(n) = 1 \tag{3.3}$$

A problem with the rectangular window is that it creates discontinuities of the signal at the start and end of the window [20]. This is caused because the math assumes the signal to be periodic.

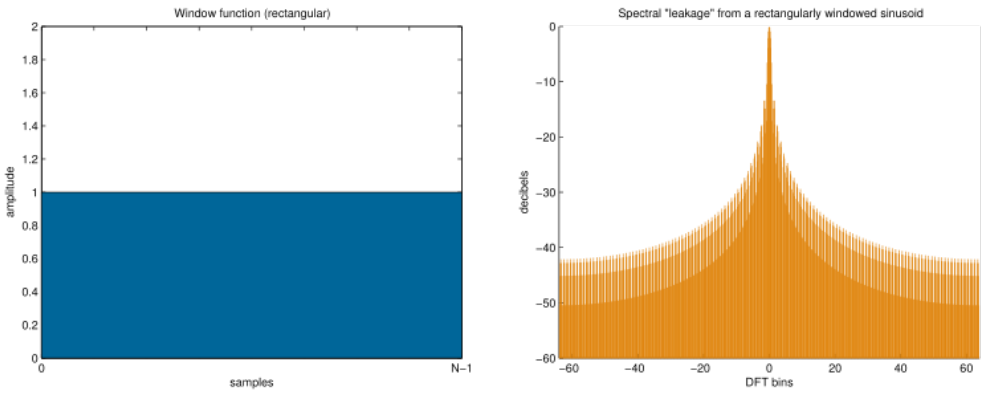

Figure 3.1: The window function and spectral leakage for the rectangular window.

Hamming Window

A Hamming window is defined by:

$$w(n) = 0.54 - 0.46\cos(\frac{2\pi n}{N-1}), \quad 0 \leq n \leq N-1 \tag{3.4}$$

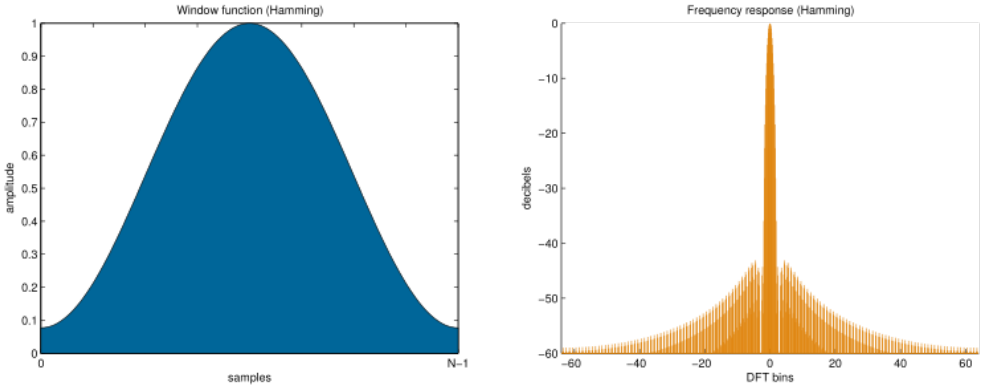

Figure 3.2: The window function and spectral leakage for the Hamming window.

Gaussian Window

The Gaussian window is defined by:

$$w(n) = \exp(-\frac{1}{2}(\alpha\frac{\pi}{N/2})^2) \tag{3.5}$$

Where $-\frac{N}{2} \leq n \leq \frac{N}{2}, \quad 2.5 \leq \alpha$ and window length $L = N + 1$. α is the reciprocal of the standard deviation.

Triangular Window

The triangular window is defined by:

$$w(n) = \begin{cases} \frac{n}{\frac{N}{2}}, & n = 0, 1, \cdots, \frac{N}{2} \\ w(N-n), & n = \frac{N}{2}, \cdots, N-1 \end{cases} \tag{3.6}$$

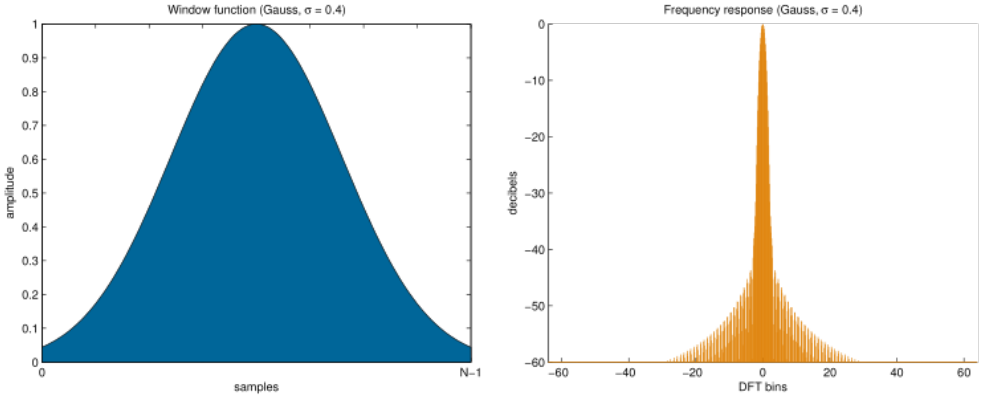

Figure 3.3: The window function and spectral leakage for the Gaussian window.

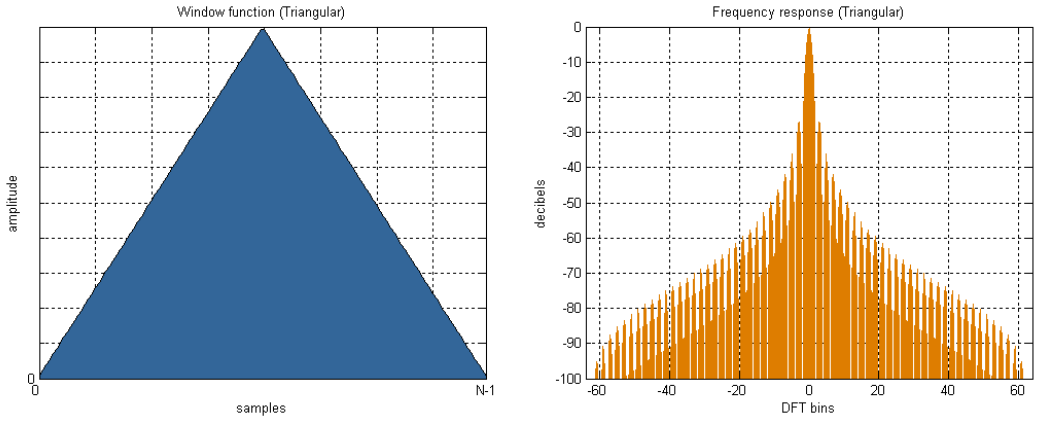

Figure 3.4: The window function and spectral leakage for the triangular window.

3.2.3 Spectrograms

A spectrogram is defined as an intensity plot of the short time Fourier transform magnitude [21].

The spectrogram is calculated by:

$$spectrogram(x,t) = \mid STFT(x,t) \mid^2 \tag{3.7}$$

Where the windows in the short-time Fourier transform is allowed to overlap

by $25 - 50$ percent.

An example of a spectrogram can be seen in Figure 3.5.

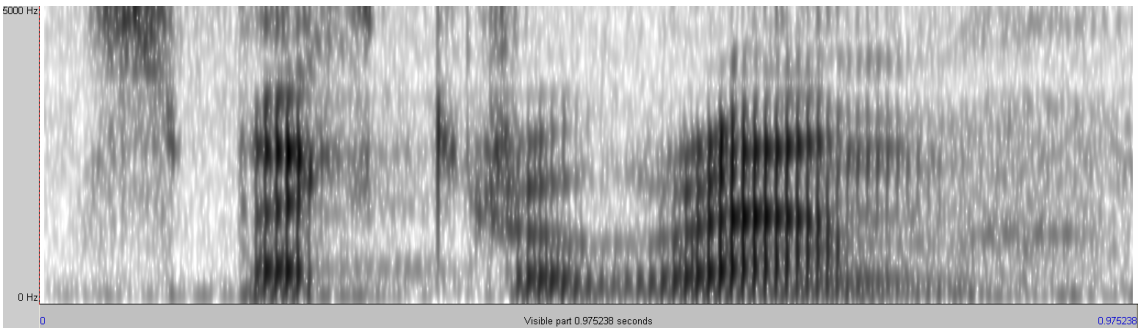

Figure 3.5: Spectrogram of the word "spectrogram", created with Praat [2]

An experienced spectrogram reader is able to recognize which words were uttered by looking at a spectrogram [14]. Understanding speech from spectrograms will be closer examined in Chapter 5.

3.2.4 Frequency Formants

A formant is the resonant frequency of the spoken phoneme. Identifying some phonemes based on the formant is possible for a trained individual. Vowels have three formants, F_1, F_2 and F_3. Each of these formant occur at 1000Hz intervals, i.e. F_1 at 0-1000Hz, F_2 at 1000-2000Hz [22].

An example of a formants in a spectrogram can be seen in Figure 3.6.

One problem with formants are that every talker have a variance from the mean values G.E. Peterson and H.L. Barney recorded 76 speakers and made a graph of the variances in formants by the speakers; shown in Figure 3.7.

The vowels are mostly separate, and can be distinguished. But some overlap, e.g., /ʊ/ and /ɝ/. Semivowels adapt to the surrounding phonemes, so their formants vary depending on the context.

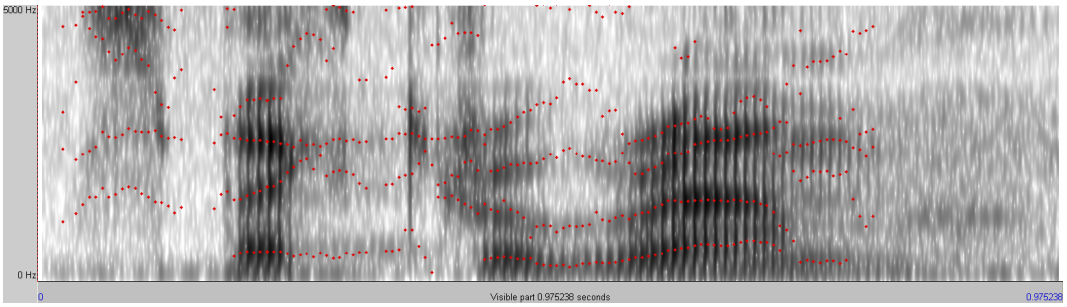

Figure 3.6: Spectrogram with formants of the word "spectrogram", created with Praat [2]

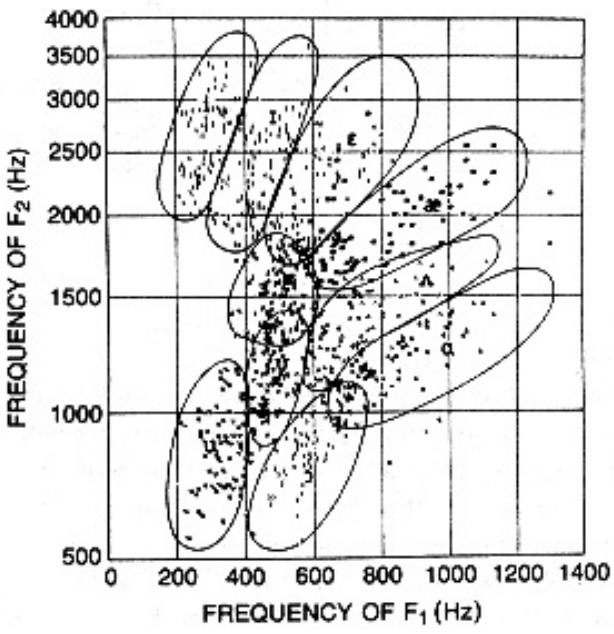

Figure 3.7: Difference in formant for different talkers, retrieved from [3]

3.2.5 Linear Predictive Coding

Linear predictive coding is a low bitrate encoding based on a speech coding system [4]. Since speech signals change rapidly, the model is used for very short segments, i.e., $10 - 20ms$. To smooth out the discontinuities from frame to frame, the model parameters measured from the previous segments The idea is to model the vocal tract as a linear all-pole filter, as shown in Figure 3.8. The unvoiced and voiced speech is generated by a random noise generator and periodic impulse train with a period equal to the desired pitch, respectively.

The transfer function for the filter is given by:

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{3.8}$$

Where a_k is a predictor coefficient for each pole, and G is a gain factor. The model improves the more poles which are used, but the 10 poles is an acceptable number to use [1, 23].

Given an input signal $x(n)$, an estimate $\hat{x}(n)$ can be calculated by linearly combining the previous signals:

$$\hat{x}(n) = \sum_{k=1}^{p} a_k x(n-k) \tag{3.9}$$

The error between the observed value and estimate can be calculated as:

$$e(n) = x(n) - \sum_{k=1}^{p} a_k x(n-k) \tag{3.10}$$

a_k can be solved from a set of linear equations by using the least-square criterion.

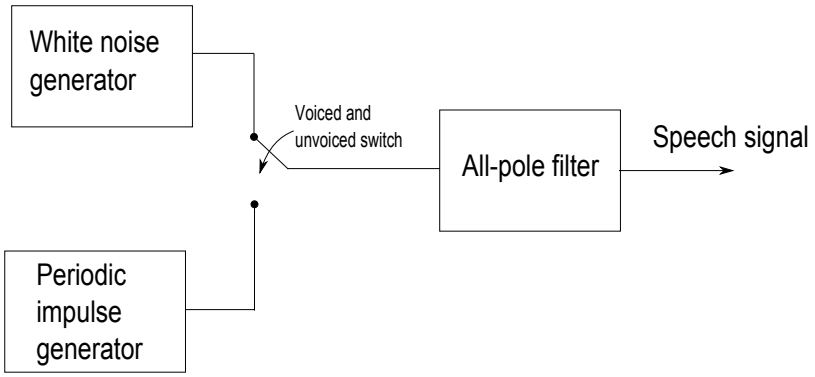

Figure 3.8: Block diagram of the vocal tract model, inspired by [4]

$$\sum_{k=1}^{p} a_k r_{xx}(l-k) = r_{xx}(l) \tag{3.11}$$

Where $l = 1, 2, \ldots, p$ and $r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n)x(n+l)$ is the time average autocorrelation of the sequence $x(n)$.

The gain G can be calculated by:

$$G^2 = r_{xx}(0) - \sum_{k=1}^{p} a_k r_{xx}(k) \tag{3.12}$$

More information on linear prediction can be read about in "Linear Prediction: A Tutorial Review" [23]. A linear predictive coding processor for speech recognition is explained in "Fundamentals of Speech Recognition" [1].

3.2.6 Perceptual Linear Predictive

The principle behind perceptual linear predictive is the same as for linear prediction, to approximate the auditory spectrum of speech by an all-pole model. Perceptual linear predictive analysis is more consistent with human hearing than linear predictive coding [5]. The consistency is achieved by warping the frequency into the Bark frequency scale. The Bark frequency scale corresponds to the first 24 critical bands of hearing, i.e., the frequencies with the most information related to hearing.

The idea behind perceptual linear predictive analysis can be seen in Figure 3.9.

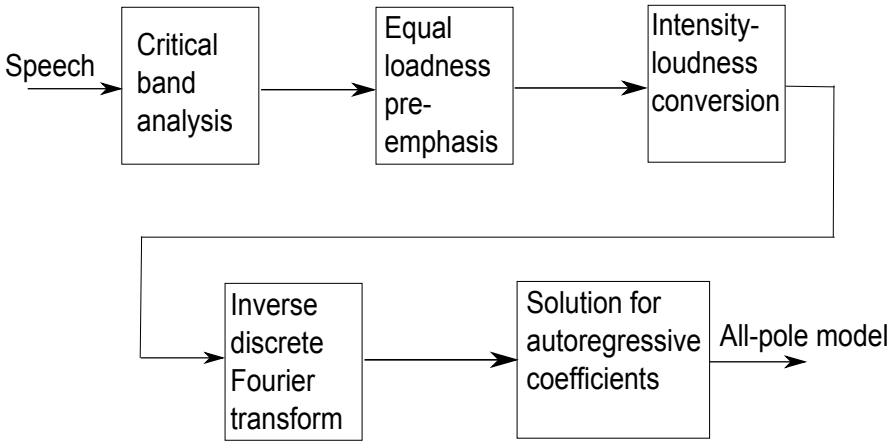

Figure 3.9: Block diagram of perceptual linear predictive speech analysis, inspired by [5]

More info about perceptual linear predictive can be read about in [5]. The paper proves a 5th order pole model to be the most consistent with the human sensitivity to the frequency changes of the first three formants. The advantage of perceptual linear predictive analysis for speaker independent automatic speech recognition is also demonstrated.

3.2.7 Mel-Frequency Cepstral Coefficients

The Mel scale was devised through human perception experiments to adjust the frequency scale so it was better suited for human hearing [24].

$Mel(f)$ is a logarithmic scale of the of the normal frequency scale. It has linear properties for frequencies under $1000Hz$, and logarithmic above, and is calculated by[25]:

$$Mel(f) = 2595log_{10}(1 + \frac{f}{700}) \tag{3.13}$$

The Mel-Frequency cepstral coefficients are computed from fast Fourier transform power coefficients filtered by triangular windows. Triangular windows were explored in Section 3.2.2.

An example of Mel-frequency cepstral coefficients can be seen in Figure 3.10, but extracting information from a visual image of the coefficients are not of much use.

3.2.8 Cepstrum

Cepstrum is a Fourier analysis of a logarithmic amplitude spectrum of a waveform [26]. The cepstrum have peaks corresponding to the spacing between the harmonics of the signal. It is possible to calculate the formant frequencies by using the cepstrum and linear predictive coding.

The conversion from a speech signal to cepstrum can be seen in Figure 3.11.

3.3 Dynamic Time Warping

Dynamic time warping is a dynamic programming algorithm which use induction to compute the similarity of two sequences [20]. The algorithm compares an input sequence to several references, i.e., stored words, to find

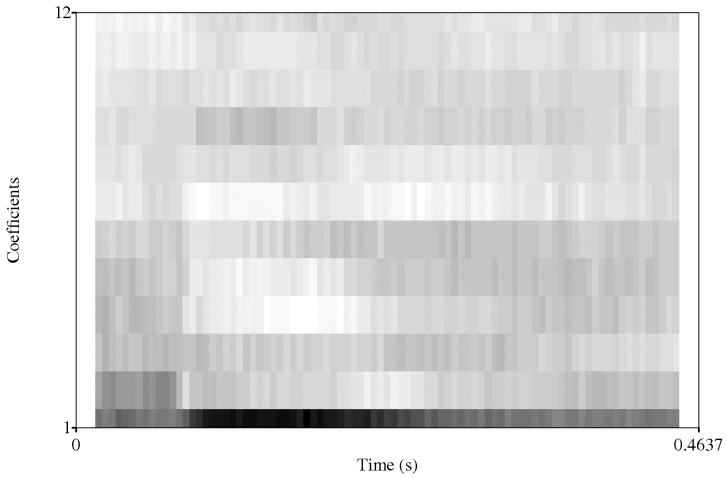

Figure 3.10: Mel-frequency cepstral coefficients of the word "Buy", created with Praat [2]

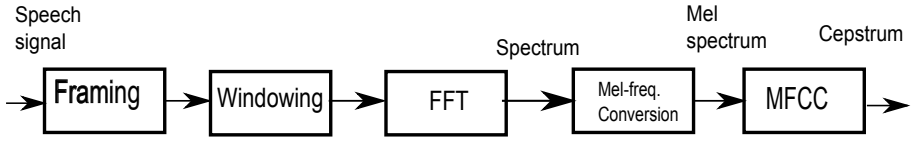

Figure 3.11: Block diagram for speech signal to cepstrum, inspired by [6]

the best match. For speech recognition, the sequence consists of features extracted from equally spaced frames of a word.

One problem with using dynamic time warping for speech recognition is the variable speaking rate, even when it is the same person who utters the word [27].

3.3.1 An Algorithm for Dynamic Time Warping

Speech is not proportional to the duration of the utterance, so it is not advisable to take a linear time normalization of the sequences [1].

The two speech pattern X and Y can be represented by two sequences; $(x_1, x_2, \ldots x_{T_x})$ and $(y_1, y_2, \ldots y_{T_y})$, where i_x and i_y are used to denote time indices. The idea is to minimize the distortion between the sequences, where the distortion is defined as:

$$d(i_x, i_y) \tag{3.14}$$

To normalize the sequences, two warping functions:

$$i_x = \phi_x(k), \quad i_y = \phi_y(k) \tag{3.15}$$

where $k = 1, 2, \ldots, T$, and T is a common time for both sequences, are introduced.

By introducing a nonnegative weighting coefficient: $m(k)$ and a normalizing factor: $M_\phi = \sum_{k=1}^{T} m(k)$, a global pattern dissimilarity measure can be defined:

$$d_\phi(X, Y) = \sum_{k=1}^{T} \frac{d(\phi_x(k), \phi_y(k))m(k)}{M_\phi} \tag{3.16}$$

To find the best path, i.e., the one with the least dissimilarity, the minimum of all possible paths can be found:

$$d(X,Y) \triangleq \min_{\phi} d_{\phi}(X,Y) \tag{3.17}$$

In Figure 3.12, the basic idea of how dynamic time warping is shown. The optimal alignment path is drawn where the distortion between i_x and i_y is minimum.

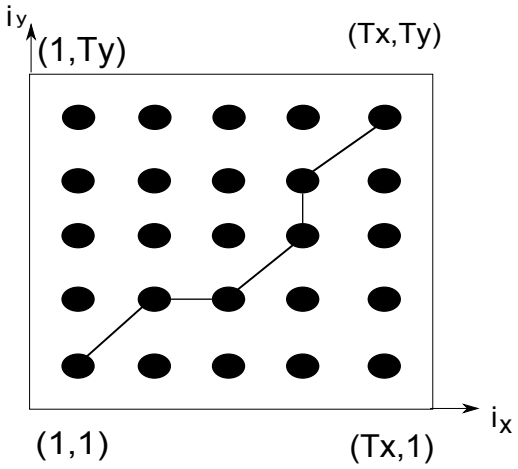

Figure 3.12: Example of dynamic time warping

Warping Constraints

There are several constraints that ϕ must satisfy.

1. **Endpoints constraints:** Even if the utterances have different lengths, the endpoints will be the same.
   Start:
   $$\phi_x(1) = 1, \quad \phi_y(1) = 1 \tag{3.18}$$
   End:
   $$\phi_x(T) = T_x, \quad \phi_y(T) = T_y \tag{3.19}$$

2. **Monotonicity:** The path, as shown in Figure 3.12, can not turn back on itself, so the path must be monotonically increasing:

$$\phi_x(k) \leq \phi_x(k+1), \quad \phi_y(k) \leq \phi_y(k+1) \tag{3.20}$$

3. **Local continuity:** To ensure that no important information, e.g., a phoneme, is ignored, a local continuity constraint is used. There are several different types of local constraints where the type used determines the alignment flexibility [11]. The one used by [27] and recommended by [1] is:

$$\phi_x(k+1) - \phi_x(k) \leq 1, \quad \phi_y(k+1) - \phi_y(k) \leq 1 \tag{3.21}$$

4. **Global path constraints:** These constraints determine which regions the optimal warping path can traverse. A constraint given by [27] is:

$$\mid \phi_x(k) - \phi_y(k) \mid \leq r \tag{3.22}$$

   This constraint ensures that the time difference between the sequences does not wander too far from each other.

   Another constraint shown in [11] is a slope of $\frac{1}{2}$ and 2 from the points $(1,1)$ and (T_x, T_y) which reduces the legal range of the global path to a parallelogram.

5. **Slope constraint:** To ensure the path does not create an unrealistic correspondence between the two sequences, a slope constraint is utilized. A way to realize this constraint is to only allow the path to take m steps horizontally before it takes n steps in the vertical direction.

Computing the Total Distortion

The algorithm for finding the best path from $(1,1)$ to (T_x, T_y) is given by:

Defining a weighted accumulated distortion between point (x', y') and (x, y):

$$\zeta((x', y'), (x, y)) = \sum_{l}^{L} d(x_l, y_l) m(l) \tag{3.23}$$

Where l is a legal path, satisfying the constraints, from (x', y') to (x, y) from 1 to L.

Initialization:

$$D(1,1) = d(1,1)m(1) \tag{3.24}$$

Recursion:

$$D(x,y) = \min_{(x',y')} [D(x',y') + \zeta((x',y'),(x,y))] \tag{3.25}$$

Computing the total distortion for the optimal path:

$$D(X,Y) = \frac{D(T_x, T_y)}{M_\phi} \tag{3.26}$$

3.4 Hidden Markov Models

Most of the material gathered here are from "Fundamentals of Speech Recognition" [1] and "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition" [28]. The article should be read to get a better understanding of hidden Markov models. The lectures of "Center for Spoken Language Understanding" [20] and "Automatic Speech Recognition" at MIT[11] is also used.

3.4.1 Markov Models

A Markov model can be described as a finite state machine with transitions between states at discrete time events. The system can only be in one state at the time, and the transition between states are expressed as possibility, i.e., $0 \leq p \leq 1$.

The terminology used here is based on [28].

- N is the number of states in the Markov chain, in Figure 3.13 $N = 4$.
- S is the different states. A number $1, 2, \ldots, N$ is used on each state.
- Time instants associated with state change is denoted $t = 1, 2, \cdots$.
- q_t is the state at time instant t.
- a_{ij} is the transitional probability from state $i \rightarrow j$. The states must also be $1 \leq i, j \leq N$.
- $a_{ij} \geq 0$
- $\sum_{j=1}^{N} a_{ij} = 1.$
- $\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N$ is the initial state probabilities.
- O is used for observations of the model, e.g., $O = \{S_1, S_2, S_2, S_3\}$ is a possible observation in Figure 3.13.

3.4.2 Example of Markov Chain

To understand how a hidden Markov model works, it is essential to have a grasp of how a Markov chain works.

Using $O = \{S_1, S_2, S_2, S_3\}$, where the first observation $\pi_1 = S_1$,and the state transition probabilities are:

$$A = \{a_{ij}\} = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 \\ 0 & 0.2 & 0.3 & 0.5 \\ 0 & 0 & 0.6 & 0.4 \\ 0.7 & 0.3 & 0 & 0 \end{bmatrix}$$

The probability of the series of states is calculated this way:

$$\begin{aligned} P(O) &= P[S_1, S_2, S_2, S_3] \\ &= P[S_1] \cdot P[S_2 \mid S_1] \cdot P[S_2 \mid S_2] \cdot P[S_3 \mid S_2] \\ &= \pi_1 \cdot a_{12} \cdot a_{22}, \cdot a_{23} \\ &= 1 \cdot 0.5 \cdot 0.2 \cdot 0.3 = 0.03 \end{aligned}$$

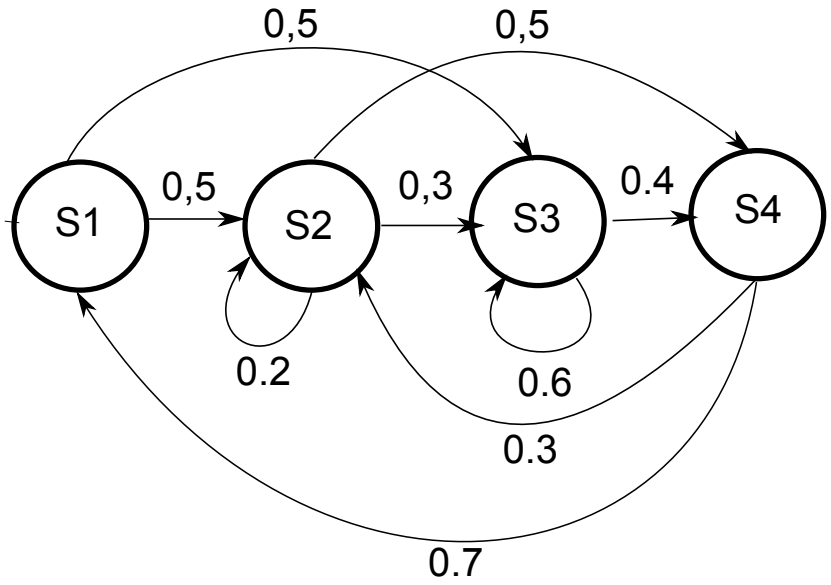

Figure 3.13: An example of how a Markov model might look.

3.4.3 Hidden Markov Models

A hidden Markov model is a Markov model where the states are hidden. There are N sates in the model, but all of these are hidden. All states are connected to each other, but other connections models are often of interest. For some practical application, the states can correspond to actual physical states.

The terminology is the same as for Markov models, with some additions:

- M is the number of distinct observation symbols for each state.
- $V = \{v_1, v_2, \cdots, v_M\}$ is the individual symbols.
- $B = \{b_j(k)\}$ $\quad b_j(k) = P[v_k \text{ at } t \mid q_t = S_j] \quad 1 \leq j \leq N \quad 1 \leq k \leq M$ at state j, is the observation symbol probability distribution.

The information can be used as generator to give an observation sequence, i.e., $O = o_1, o_2, \cdots, o_T$, in this manner:

1. Choose initial state $q_1 = S_j$, according to the initial state distribution π:
2. $t = 1$:
3. Choose $o_t = v_k$ according to $b_i(k)$.
4. $q_{t+1} = S_j$ according to a_{ij}.
5. $t = t + 1$, and return to step 3 as long as $t < T$. End otherwise.

The notation:

$$\lambda = (A, B, \pi) \tag{3.27}$$

is used to describe a complete set of parameters for the model.

There are three problems associated with hidden Markov models.

1. **How to efficiently calculate $P(O \mid \lambda)$.** When a state transition from every state to any other state is possible, the runtime for an observation sequence is $O(2TN^T)$. Calculating this is infeasible with the number of observation and states which are expected. The problem can be solved by using the forward part of the Forward-backward algorithm. The backward part is used in problem 3.
2. **Given an observation sequence, how to optimally choose a state sequence.** There are several solutions to this problem depending on how optimal state sequence is defined. The solution to the best state sequence path can be achieved by maximizing $P(Q, O \mid \lambda)$. The Viterbi algorithm is used to accomplish this.
3. **How to adjust the model parameters $\lambda = (A, B, \pi)$ to best describe the observation sequence.** This is how the hidden Markov model trains to adjust to the surroundings. It is considered the most difficult of the problems because no algorithm exists which accomplishes this. The closest is the Baum-Welch algorithm that adjust λ to locally maximize $P(O \mid \lambda)$.

3.4.4 Forward-backward Algorithm

The forward variable is defined as:

$$\alpha_t(i) = P(o_1 o_2 \cdots o_t, q_t = S_i \mid \lambda) \tag{3.28}$$

The probability of the partial observation sequence $o_1 \cdots o_t$ and the state S_i at time t with the hidden Markov model λ.

With this definition the variable $\alpha_t(i)$ can be solved with:

1. **Initialization:**

$$\alpha_1(i) = \pi_i \cdot b_i(o_1), \qquad 1 \leq i \leq N \tag{3.29}$$

   Checks which state is the most likely to be in given the observation o_1.

2. **Induction:**

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(j) a_{ij}\right] b_j(o_{t+1}) \tag{3.30}$$

   Where $1 \leq t \leq (T-1), 1 \leq j \leq N$.

   Equation (3.30) iterates over the rest of the observation. $\alpha_t(j)a_{ij}$ calculates the possibility of the state to change to S_j from S_i, this is summed and multiplied to the probability of being in state S_j given the current observation.

3. **Termination:**

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{3.31}$$

   Because $\alpha_T(i) = P(o_1 \cdots o_T, q_T = S_i \mid \lambda)$ the probability $P(O \mid \lambda)$ is calculated by summing the possibilities at time step T.

The runtime for this algorithm is $O(N^2T)$, which is feasible to calculate.

Both the results of the forward and backwards calculations are used in the Baum-Welch algorithm which will be explored in Section 3.4.6.

The backward variable is defined as:

$$\beta_t(i) = P(o_{t+1}o_{t+2}\cdots o_T \mid q_t = i, \lambda) \tag{3.32}$$

The calculation is initialized with:

$$\beta_T(i) = 1, \qquad 1 \leq i \leq N \tag{3.33}$$

Where the value for T is chosen to one so the value of the computation does not affect the result.

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \tag{3.34}$$

Where $t = T-1, T-2, \cdots, 1, \quad 1 \leq i \leq N$

For examples several good examples to understand the algorithm, the CSLU lecture folder [20] should be read.

3.4.5 Viterbi Algorithm

The Viterbi algorithm is used to calculate the most likely path of states given an observation sequence. It does not compute the probability of being in a given state. This is the main difference from the forward algorithm.

A variable which represents the highest probability along a single path at time t is defined as:

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1 q_2 \cdots q_t = i, o_1 o_2 \cdots o_t \mid \lambda] \tag{3.35}$$

It accounts for all earlier observations, and ends up in state S_i.

For any value of t, we have:

$$\delta_t(i) = \left(\max_i \delta_{t-1}(i)a_{ij}\right) b_j(o_t) \tag{3.36}$$

The algorithm can be divided in to four steps:

1. **Initialization:**

$$\delta_1(j) = \pi_i b_i(o_1) \tag{3.37}$$
$$\psi_1(i) = 0 \tag{3.38}$$

   Where $1 \leq i \leq N$.

2. **Recursion:**

$$\delta_t(j) = \left[\max_{1\leq i\leq N} \delta_{t-1}(i)a_{ij}\right] b_j(o_t) \tag{3.39}$$
$$\psi_t(j) = \arg\max_{1\leq i\leq N}[\delta_{t-1}a_{ij}] \tag{3.40}$$

   Where $2 \leq t \leq T, \quad 1 \leq j \leq N$.

   $\psi_t(j)$ is the best path prior to S_j at time t.

3. **Termination:**

$$P^* = \max_{1\leq i\leq N}[\delta_T(i)] \tag{3.41}$$
$$q_T^* = \arg\max_{1\leq i\leq N}[\delta_T(i)] \tag{3.42}$$

4. **State sequence backtracking:**

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \cdots, 1 \tag{3.43}$$

   The backtracking is done because the best sequence can change as t increases.

The results of the Viterbi algorithm yield very small values. So the calculations should be done in the log-domain to avoid underflow errors.

3.4.6 Baum-Welch Algorithm

The algorithm uses the results of the forward-backward algorithm to reestimate the model $\lambda(A, B, \pi)$ to locally maximize $P(O \mid \lambda)$.

To reestimate the model, the probability of being in state S_i at time t and S_j at time $t+1$ is introduced using the results from the forward-backward algorithm:

$$\xi(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) \tag{3.44}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \tag{3.45}$$

Given the observation sequence and the model, the probability of being in state S_i at time t can be expressed as:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{3.46}$$

The Re-estimated variables can be calculated:

- **Re-estimation of initial state probabilities:**

$$\bar{\pi}_i = \gamma_1(i) \tag{3.47}$$

- **Re-estimation of the state transition probabilities:**

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{3.48}$$

- **Re-estimation of the observation symbol probability distribution:**

$$\bar{b}_j(k) = \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state j}} \tag{3.49}$$

$$= \frac{\sum_{\substack{t=1 \\ s.t\, o_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{3.50}$$

By iterating this calculation with the re-estimated values, a local maximum will be reached.

3.4.7 Hidden Markov Model for Speech Recognition

A hidden Markov model usually corresponds to a phoneme or a word, where the states can be associated with:

- **Sub-phoneme:** A phoneme is usually split into three parts. This is advantageous because the first and third is influenced by the surrounding phonemes.
- **Phoneme:** The whole phoneme is used as a state, and can transitions can build words.
- **Sub-word:** This approach is designed to recognize words, and states can be defined to best match the word.

The hidden Markov models can be combined to create higher levels systems, e.g., sequence of hidden Markov models for phonemes $\Rightarrow$ word level system.

Gaussian Mixture Models

Gaussian mixture models are used to estimate the probability of an observation given the state [20], i.e., $b_j(o_t)$. The probability of being in a state is independent from the probabilities of being in other states.

The probability is given by:

$$b_j(o_t) = \sum_{k=1}^{M} c_{jk} N(o_t; \mu_{jk}, \sigma_{jk}) \tag{3.51}$$

Where M is the number of mixture components and c_{jk} is a mixture weight.

3.5 Artificial Neural Network

Artificial neural networks is the way a computer mimics the human brain [29]. It is not well suited for automatic speech recognition, but have been shown to yield good results for short isolated speech units [30]. There have been several applications that uses a hybrid hidden Markov model and artificial neural network for automatic speech recognition.

There are several different type of artificial neural networks, three popular are [6]:

- **Multilayer Perceptron:** The network tries to minimize the differences between expected and real outputs from the system.
- **Elman Network:** Uses back propagation as a learning for the system.
- **Probabilistic Neural Network:** Use a probability distribution function to calculate the network connections weights.

An example of how a artificial neural network works is shown in Figure 3.14. Mel-frequency cepstral coefficients, discussed in Section 3.2.7, is usually used

as input to the model. The number of input, hidden and output layers can varies in different implementations, and the example shown is an extremely simplified case.

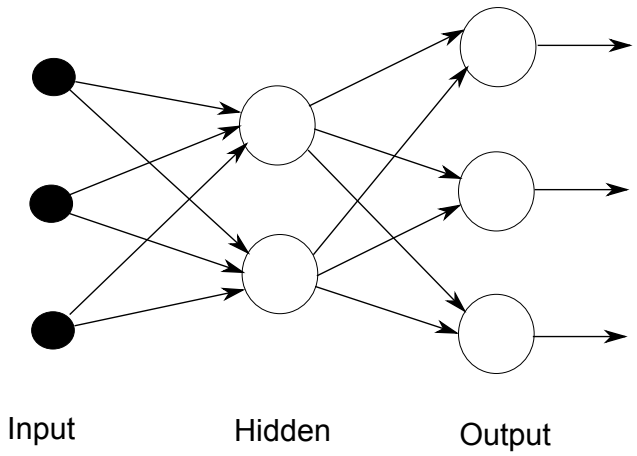

Figure 3.14: Example of a multilayer perceptron artificial neural network, inspired by [7]

An artificial neural network which have not learned anything is ready for work, but it has not experienced any stimulus yet, so it is not able to do anything [31]. A multilayer perceptron is trained by giving it an input pattern where the correct response is known. The output is then observed, and the error between the expected output and the networks output is computed. This error is used to adjust the weights within the network.

Chapter 4

Programs and APIs for Speech Recognition

4.1 Introduction

There are several different toolkits that are designed for speech recognition. Each uses different methods and is designed for different purposes. This chapter will look at some promising programs and toolkits used for speech recognition, and evaluate the purpose and how effective they are.

4.2 Free Toolkits and APIs for Speech Recognition

4.2.1 Praat: Doing Phonetics by Computer

Praat [2] is a program developed by Paul Boersma and David Weenink. The program is used for speech analysis, it is possible to record sound, create spectrograms and analyze formats, pitch, intensity, Mel-frequency cepstral

coefficients, linear predictive and several other features.

Getting Started

All objects, e.g., sounds, spectrograms formants, spectrums, intensity plots, are stored in the Praat objects window, which is shown in Figure 4.1. From here, new sound files can be recorded, or already existing sound files can be opened.

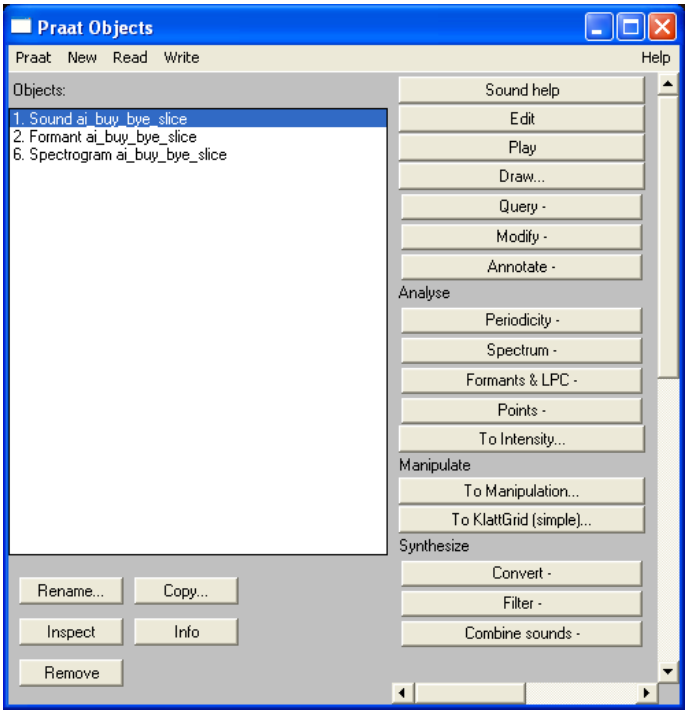

Figure 4.1: The Praat objects window

The menu on the right gives options to manipulate the object. Sound files, e.g., a .wav file, can be analyzed in several ways and create new object files:

- **Formants:** As shown in Figure 4.2, the formants of a sound file can be extracted. Praat gives the options of drawing a picture of the formants, or a list of the frequency of all the different formants at a various times.
- **Spectrogram:** Figure 4.3 shows a spectrogram created by Praat.

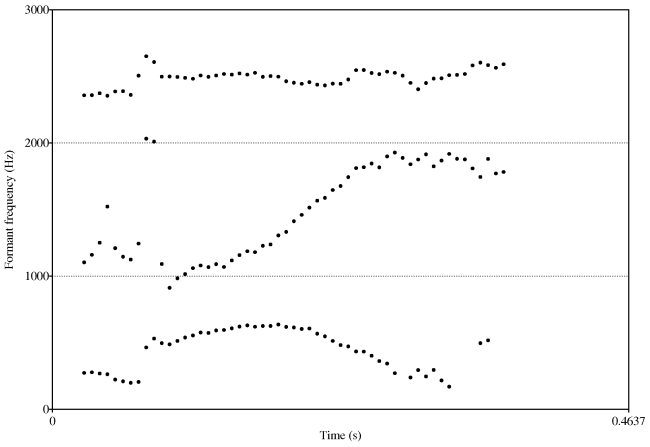

Figure 4.2: The Praat formant object picture of the word "Bye"

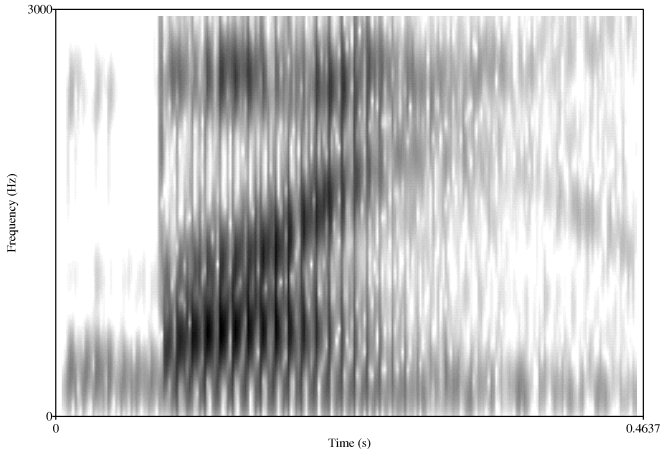

Figure 4.3: The Praat spectrogram object picture of the word "Bye"

4.2.2 Hidden Markov Models Tool Kit (HTK)

HTK[32] is a tool kit for building and manipulating hidden Markov models. The library modules and tool kit is written in C, and the source code is available on their web site after registering and agreeing to their license.

HTK is primary aimed at recognizing words or sub-words by using hidden Markov models. Forward-backward algorithm, Viterbi algorithm and Baum-Welch re-estimation is used to recognize single words. For continuous speech, several hidden Markov models are connected together. It is also possible to build a phoneme recognizer or other types of recognizers.

The different programs generated when compiling HTK can be read about in [33].

4.2.3 Center for Spoken Language Understanding Toolkit (CSLU)

CSLU is a research center at Oregon Graduate Institute of Science and Technology. They have created a large toolkit with several applications and development opportunities.

Content of the Toolkit

- Users, i.e., people who use the finished product to learn a language.
  - **Rapid Application Developer (RAD):** Is a graphical tool for creating structured dialogs between the user and a computer. RAD makes it possible to quickly create a speech interface without any prior knowledge. Text-to-speak is a part of this program.
  - **Baldi:** Animated head that is anatomically correct. It can be made transparent, so the user can watch how the mouth looks when a word is produced. The face can be used within RAD and the user can record voice and have Baldi speak with the voice of the user. Baldi is primary made for the teaching aspect of CSLU toolkit.

- **Tutor:** Available in English and Spanish. Is designed to help people recognize words and pronounce them.
- **SpeechView:** SpeechView is similar to Praat in that speech can be recorded and analyzed. The SpeechView program is shown in Figure 4.4. The top spectrogram is the default spectrogram, the second is manually manipulated to remove noise. The panel at the button shows where the different phonemes are located, this has been entered manually. 2D, 3D and color 3D spectrograms can be shown, pitch contour and energy contour can also be displayed.

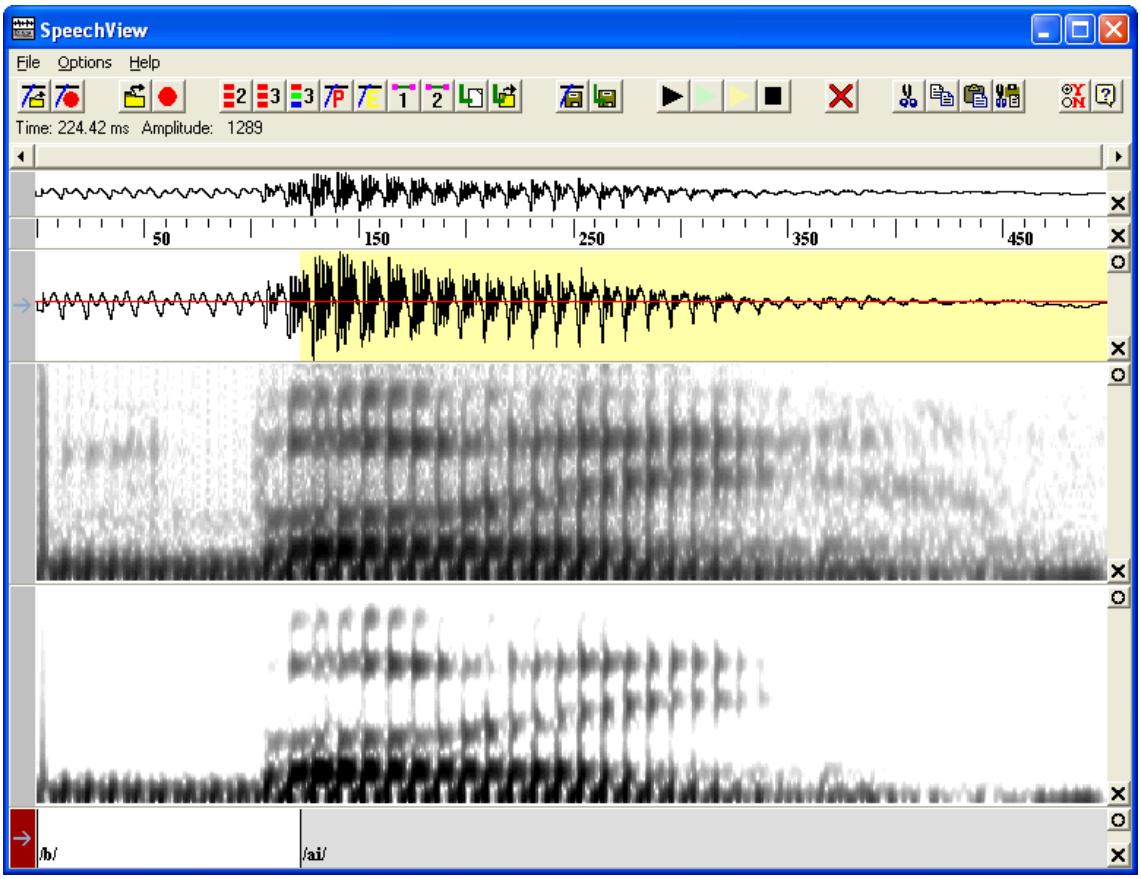

Figure 4.4: SpeechView, a part of the CSLU Toolkit. The word uttered was "Bye" recorded at 8000 samples per second.

- Developers.
  - **Tool Command Language (Tcl):** CSLU toolkit supports scripting with Tcl. Several packages are included in the toolkit,

and it is possible to develop word based recognizers as demonstrated in the Viterbi package [34], or more complex applications as demonstrated in the Toolkit Development Guide[35].

– **C library:** The library gives the possibility to develop speech recognition programs with C code.

The feature extractions possible with the library are [36]:

* Power spectral analysis (FFT)
* Linear predictive analysis (LPC)
* Perceptual linear prediction (PLP)
* Mel scale cepstral analysis (MEL)
* Relative spectra filtering of log domain coefficients (RASTA)
* First order derivative (DELTA)
* Energy normalization

Phoneme probability models used for word models, lexical trees and grammar can also be created.

The English subset of Worldbet[37] is used to write phonetic symbols because IPA symbols are difficult to write in plain text.

In stead of having a model for each phoneme, CSLU argues that phonemes are influenced by the adjacent phonemes [38]. They group phonemes in eight broad context groups depending on what they spectrally look like:

- **Front:** Phoneme that resembles a front Vowel.
- **Middle:** Phoneme that resembles a middle Vowel.
- **Back:** Phoneme that resembles a back Vowel.
- **Sil:** Silence.
- **Nasal:** Nasal phoneme.
- **Retro:** Retroflexe semivowel. The only English phoneme in this category is /r/ but many vowels becomes retroflexed when an /r/ is succeeding.
- **Fric:** A fricative vowel.

- **Other**

To recognize a phoneme in a word, three models are needed. One for the preceding phoneme, one for the context free part of the model, and one for the succeeding phoneme. There are eight possibilities for each proceeding and succeeding phoneme, so each phoneme have 17 categories defined. This results in a total of 576 categories for English phonemes.

The toolkit uses both hidden Markov model and artificial neural networks for automatic speech recognition. It is possible to train a recognizer for specific words. Each word must be phonetically described, and new context groups can be specified depending on the words which will be recognized. Each phoneme then needs to be split into three parts where the first and third part is dependent on the preceding or succeeding phoneme, respectively. The more training data the model have, the more precise the end result will be.

4.3 Experience with the Toolkits

4.3.1 Experience with Praat

Praat is easy to use and is excellent for learning the various methods used in speech recognition. Praat is not open source, which mean it is not possible to look at the implementations of the methods or use it as a basis for an implementation.

4.3.2 Experience With HTK

An error was encountered when trying to compile HTK on Windows XP Professional x64, with Microsoft Visual Studio 2005. After some research, it was discovered that HSLab is dependent on "X Window System" for GUI, which is only supported on UNIX systems. The dependency can be removed in the makefile, but the program HSLab will not be compiled.

To overcome this problem, other solutions were tried:

- **Ubuntu 8.04 64 bit:** HTK was made for a 32 bit system, a lot of problems were encountered while trying to compile it on a 64 bit system.
- **Ubuntu 9.10 32 bit through Virtualbox:** Trying to run Ubuntu with Virtualbox on 64 bit XP was caused the computer to restart 80-90 percent of the time. When Ubuntu booted, no sound was heard and it was impossible to record anything.
- **Ubuntu 9.10 32 bit:** HTK compiled without problem, but the there was a problem with the sound card so recording sound did not work.
- **Cygwin:** There same problem with 64 bit and 32 bit was encountered while trying Cygwin.

After using several weeks learning about HTK, trying to install it with various methods and operating systems further experimentation was abandoned.

4.3.3 Experience With the CSLU Toolkit

The CSLU-C Toolkit for automatic speech recognition i interesting because it is possible to create own recognition models for specific words, but it is heavily integrated in the phonetic model used by CSLU, and described in Section 4.2.3. The toolkit is not open source, but the header files are available with some commentary. One example program of a digit recognizer is also available and compiles without any problems, but the lack of tutorials and source code to examine make it difficult to use.

A tutorial on the training of hybrid hidden Markov models/artificial neural networks is given at [39]. This method requires a lot of data for training, which is given for the tutorial, but to train the model for new words, would require a lot of work.

One error in the tutorial was found.

Corpora.txt should be changed from:

```
corpus: numbers
    wav_path    /tutorial/data/speechfiles
    txt_path    /tutorial/data/txtfiles
    phn_path    /tutorial/data/phnfiles
    format      {NU-([0-9]+)\.[A-Za-z0-9_]+}
    wav_ext     wav
    txt_ext     txt
    phn_ext     phn
    cat_ext     cat
    ID:         {regexp $format $filename filematch ID}
```

to:

```
corpus: numbers
    wav_path    tutorial/data/speechfiles
    txt_path    tutorial/data/txtfiles
    phn_path    tutorial/data/phnfiles
    format      {NU-([0-9]+)\.[A-Za-z0-9_]+}
    wav_ext     wav
    txt_ext     txt
    phn_ext     phn
    cat_ext     cat
    ID:         {regexp $format $filename filematch ID}
```

Chapter 5

Understanding Spectrograms

5.1 Introduction

This chapter will focus on how to recognize speech with spectrograms. The theory will be presented first,where the sources are "Spectrogram Reading" [14] and the website of Robert Hagiwara, assistant professor at the department of linguistics university of Manitoba examples[40].

All speech files are recordings of the authors voice using the program Praat [2]. All words were recorded with stereo sound at 44100 Hz, which is enough to represent the whole human hearing frequency. Spectrograms were also made by Praat, using a Gaussian window.

Since the author is not a native English speaker, the figures might vary from spectrograms found at other sources.

5.2 Analyzing Spectrograms

Spectrogram reading can be difficult because two spectrogram of the same utterance can always have little variations. The best way is to look for clues

of different phonemes.

5.2.1 Vowels

Vowels are not important for comprehension of written text, but many speech recognition systems rely on vowels to perform well.

Vowels are considered the easiest phonemes to recognize. Three formants are visible in the spectrogram as concentration of energy moving horizontally. The two lowest formants are needed to recognize a vowel. Usual formants frequencies for the vowels are shown in the table below:

Vowel	F_1	F_2	F_3
Collected from	[1]/[14]	[1]/[14]	[1]/[14]
/i/	270/280	2290/2250	3010/2900
/ɪ/	390/400	1990/1900	2550/2550
/ɛ/	530/550	1840/1770	2480/2490
/æ/	660/690	1720/1660	2410/2490
/ʌ/	520/640	1190/1190	2390/2390
/a/	730/710	1090/1100	2440/2640
/ɔ/	570	840	2410
/ʊ/	440/450	1020/1030	2240/2380
/u/	300/310	870/870	2240/2250
/ɝ/	490	1350	1690

The phoneme /ɝ/ is actually an r-coloring of the phoneme /ʌ/.

Figures 5.1, 5.2 and 5.3 shows spectrograms of the vowels while pronouncing the example words given in Appendix A. The arrow points to where the phoneme is located in the word.

5.2.2 Diphthongs

Diphthongs are similar to vowels in having a very clear intensity for the formants. The way a diphthong moves from one vowel to another can be

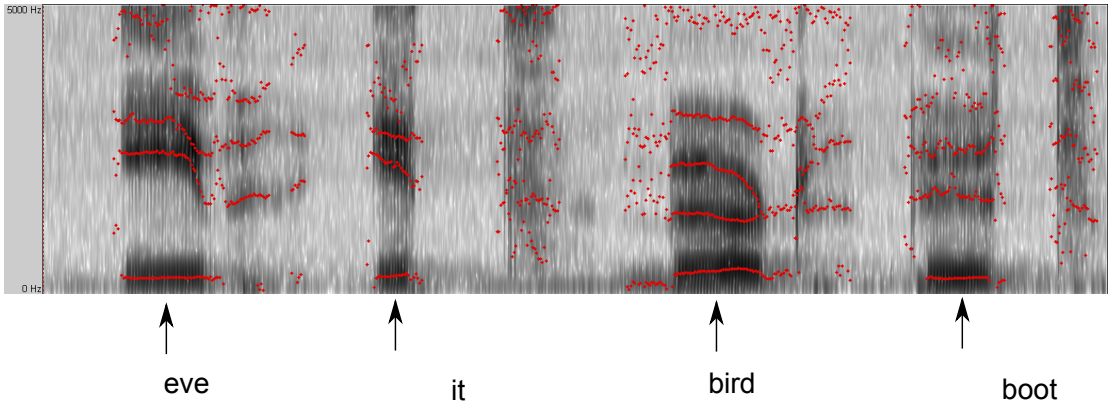

Figure 5.1: Spectrogram example of the first four vowels

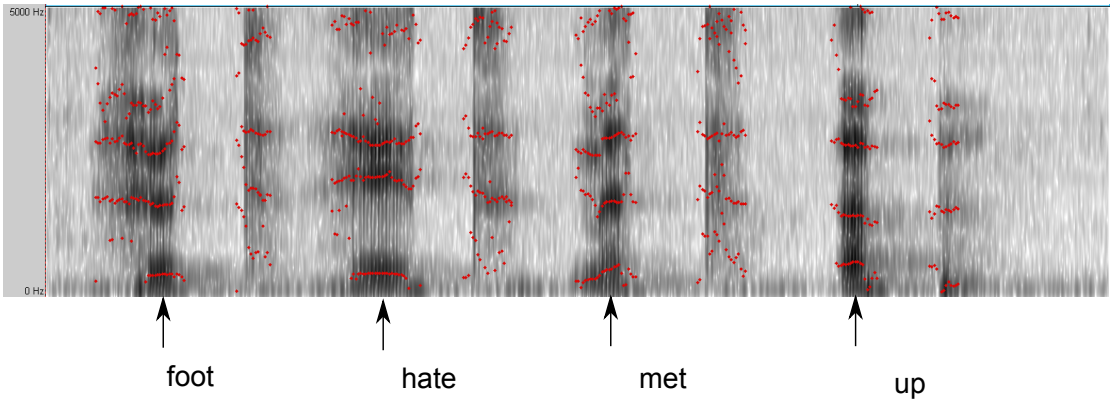

Figure 5.2: Spectrogram example of the four mid vowels

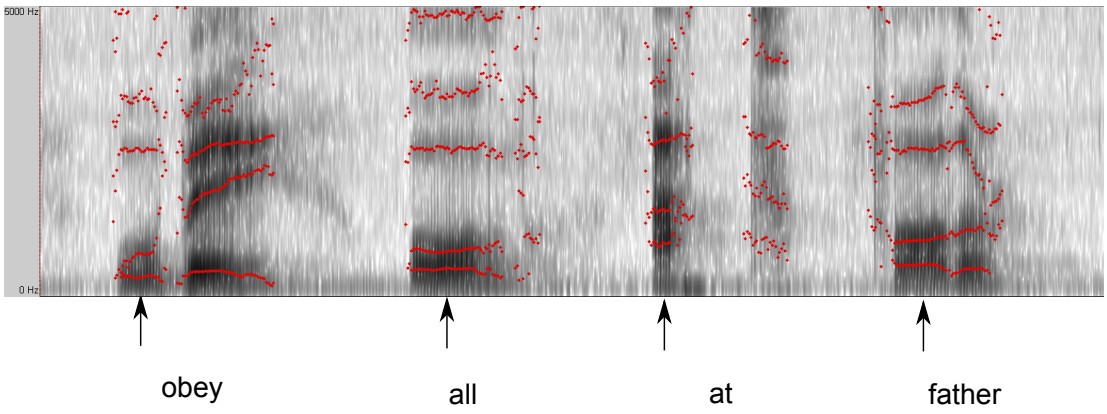

Figure 5.3: Spectrogram example of the four end vowels

seen in the spectrogram.

Even though the diphthongs are written in IPA as the vowels they move from to the ones they move to the spectrogram clues for recognizing each diphthong is a bit more complex:

- **/aɪ/:** Begins near /ʌ/, moves to /i/. F_2 has a sharper rise than /ei/.
- **/aʊ/:** Begins near /ʌ/, moves to /u/.
- **/ei/:** Begins near /ɛ/ and /ɪ/, moves to /i/.
- **/ɔi/:** Is very similar to /aɪ/, but F_1 and F_2 are lower.
- **/iʊ/:** F_1 contains higher energy than the other formants, and F_2 sinks towards F_1.
- **/oʊ/:** F_1 and F_2 can merge together, and are lower in value than /aʊ/.

Figures 5.4 and 5.5 show spectrograms of all the diphthongs respectively.

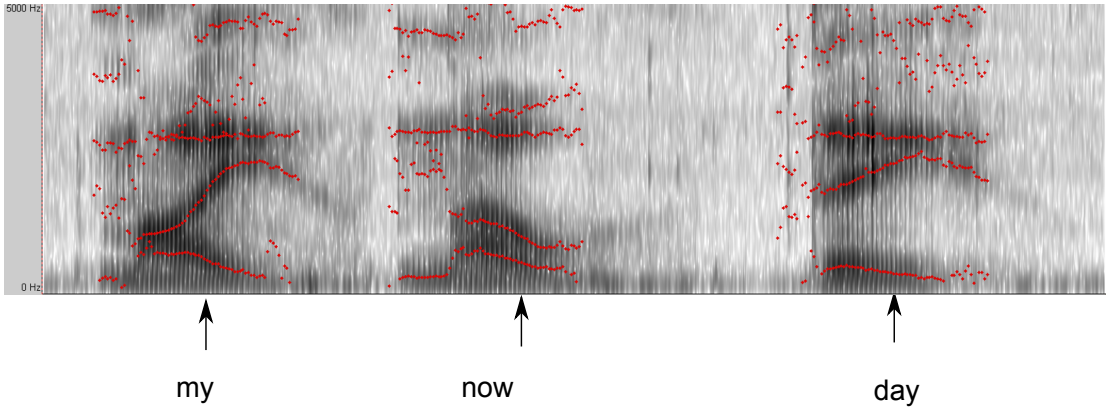

Figure 5.4: Spectrogram example of the first three diphthongs

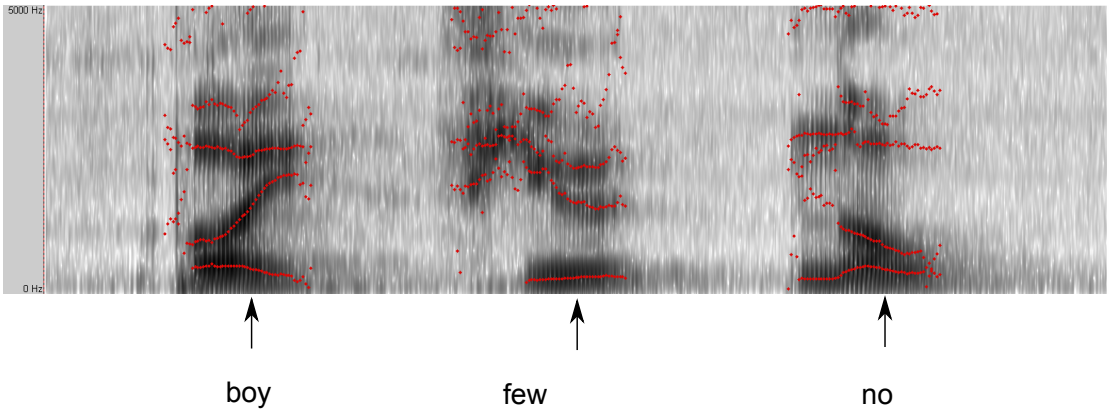

Figure 5.5: Spectrogram example of the latter three diphthongs

5.2.3 Approximants

Approximants are more difficult to recognize than vowels, mostly due to their nature to adapt to the nearby phonemes. The intensity of the formants are less than the vowels.

All approximants have in common that F_1 is in a mid to high position and the formants are lesser intense than for vowels. The specific clues about the phonemes are:

- **/l/:** Can be identified by the lack of resonance in the $1500 - 2000Hz$ range. F_2 and F_3 diverge through the /l/.
- **/r/:** F_3 is always below $2000Hz$, and has the same shape as F_2, they might be overlapping.
- **/w/:** Starts out week with only F_1 visible, F_2 and F_3 becomes clearer later if at all with F_3 usually above $2000Hz$.
- **/j/:** F_2 and F_3 nearly collide and which causes the pattern to look like an X. Some friction can also be seen.

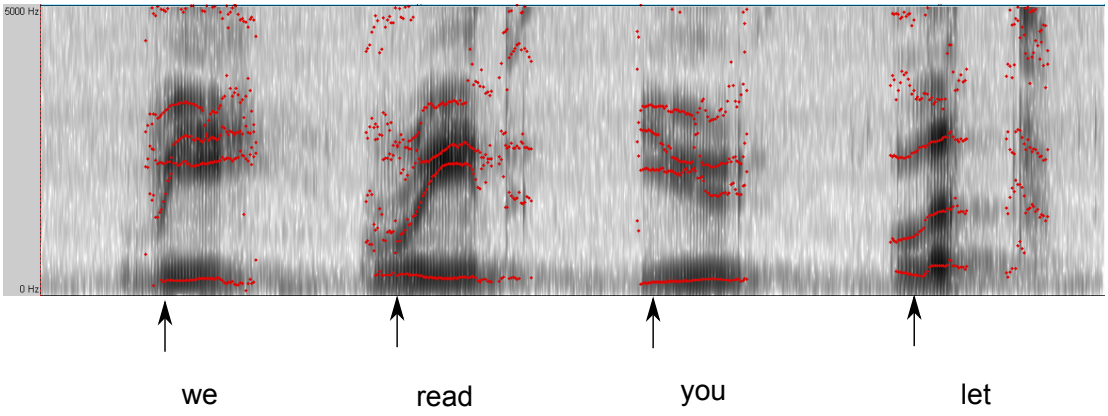

Figure 5.6: Spectrogram example of the approximants

5.2.4 Nasal Consonants

Nasal consonants are difficult to recognize. The spectral clues they have in common is less energy than vowels, and the areas between the formants contain almost no energy at all. It is also very little energy in the higher formants.

The formant F_2 of nasal consonants is the one most affected by the preceding and succeeding phonemes. Clues for each phoneme are listed below:

- **/m/:** The formants are descending F_2 is usually $900 - 1400Hz$.
- **/n/:** Formants are discontinuous at the beginning and end of the phoneme. F_2 is usually $1500 - 1800Hz$.
- **/ŋ/:** F_2 rises to $1900 - 2000Hz$, F_3 and F_4 get pinched together. Depending on the speaker, the /g/ may be pronounced which shows up as a stop.

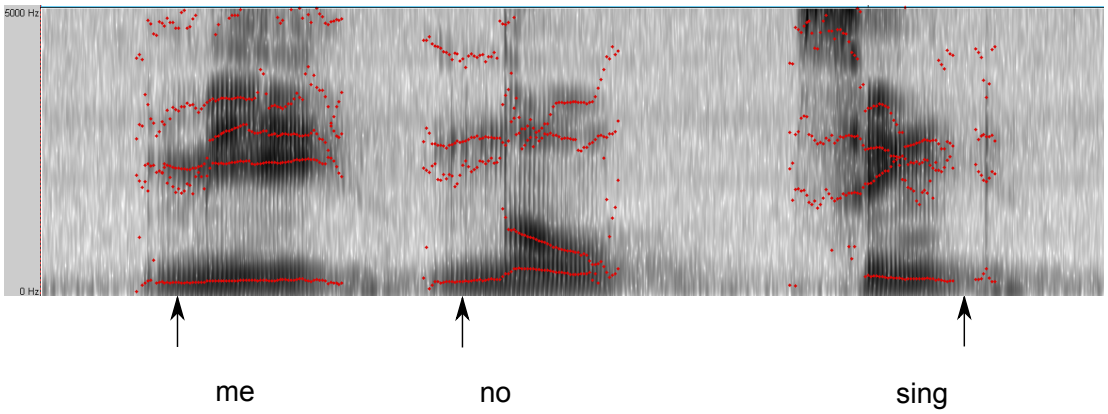

Figure 5.7: Spectrogram example of the nasal consonants

5.2.5 Fricatives

To recognize a phoneme as a fricative is not that difficult, the problem is to distinguish them from each other.

Fricatives have a lot of energy in the higher frequency bands. If a formant is present in the spectrogram, it is not caused by the harmonics of the phoneme, but by the noise which is caused by the friction.

- **/f/:** An upwards triangle shape is present around $1200Hz$. The energy of the higher frequencies are also stronger than for /v/, /θ/ and /ð/.
- **/s/:** Contains the most energy of the fricatives and is considered the easiest to recognize. There is a lot of energy between $3000 - 4000H$ and $8000Hz$.
- **/ʃ/:** Similar to /s/, but the strong energy begins at $2500Hz$.
- **/θ/:** Similar to /f/, but the energy will start at $1500 - 2500Hz$ with a highest concentration above $3000HZ$.
- **/h/:** Contains weak energy which looks like noise, and is a very volatile phoneme. The best clue is given by the succeeding phoneme, where the transition from an /h/ is very strong.
- **/v/:** There are several possible ways a /v/ can look in a spectrogram:
  - A strong noise friction around $4000Hz$.
  - Several burst-like noise friction.
  - No friction at all.

  One common factor is that /v/ is usually similar to /f/, but with the triangle pointing downwards around $1200Hz$.
- **/ð/:** The fricative with the least energy. If it is visible the energy is concentrated around $1500 - 2500Hz$ with some higher energy noise.
- **/z/:** Similar to /s/ with friction in the same area, but will usually have a glottal stops below $500Hz$.

- **/ʒ/:** Similar to /ʃ/ with friction in the same area, but there is glottal stops below $500Hz$.

Figures 5.8, 5.9 and 5.10 shows the examples of the fricatives as mentioned respectively. The formants are not shown because they are not that essential when recognizing fricatives. The spectrogram is represented in the range $0 - 8000Hz$ to illustrate the energy in the higher frequency bands.

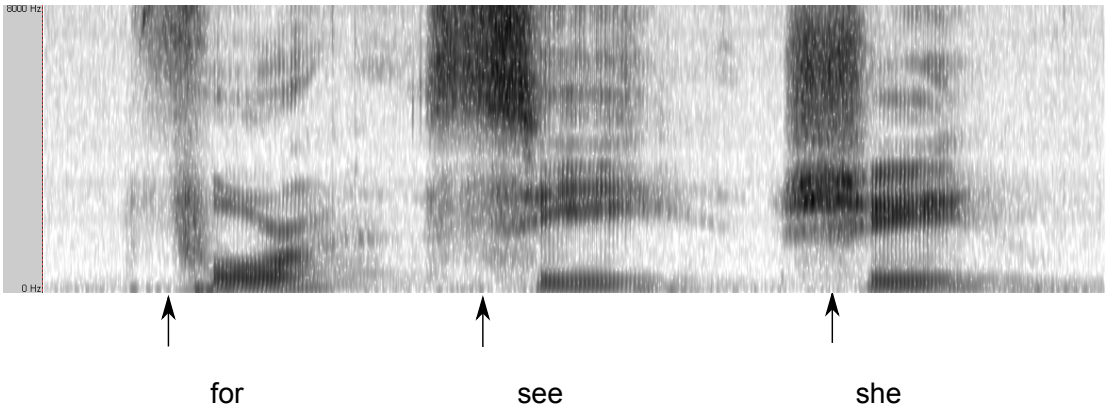

Figure 5.8: Spectrogram example of the three first fricatives

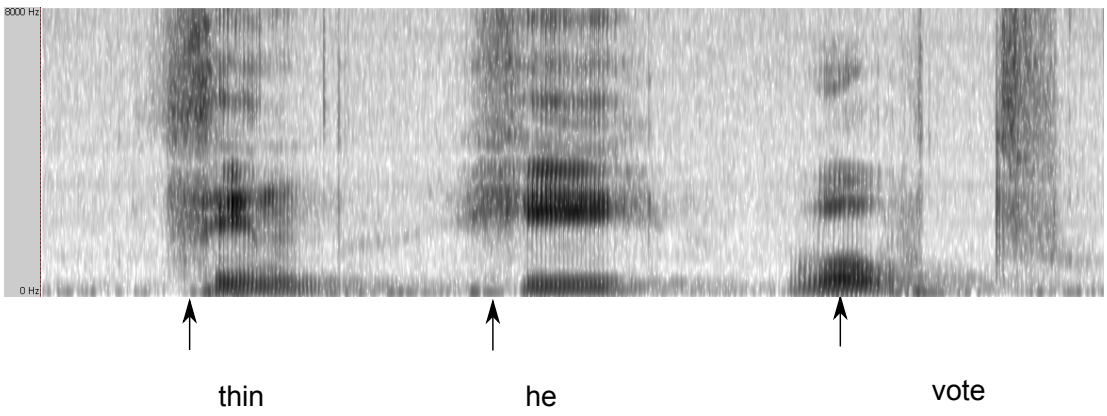

Figure 5.9: Spectrogram example of the three mid fricatives

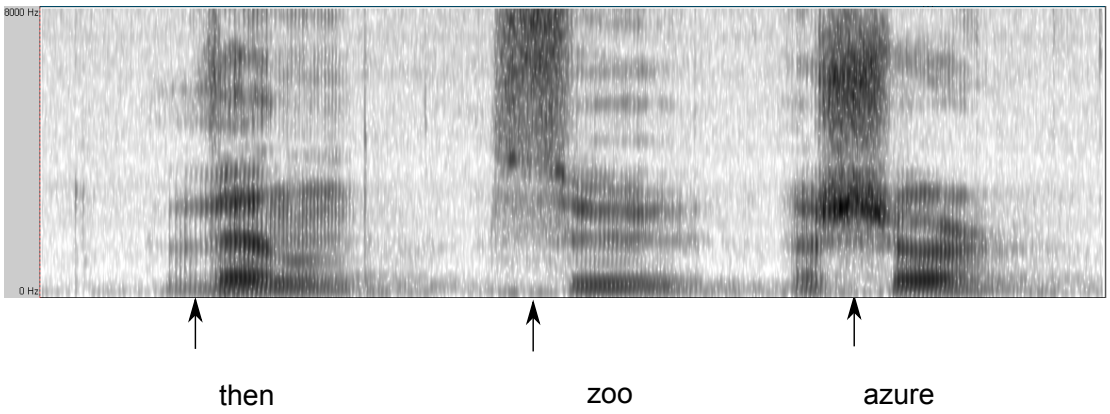

Figure 5.10: Spectrogram example of the three latter fricatives

5.2.6 Stops

Stops goes from almost total silence, also called a closure, to the release of the words. This can be seen in the spectrogram as a period of almost no energy, to a sudden burst of high energy.

Stops also influence the formants of the preceding and succeeding vowels. Each voiced/unvoiced, i.e., /b/-/p/, /d/-/t/ and /g/-/k/, pair have the same effect on the vowel.

- **/b/ and /p/:** /b/ is the weakest of the voiced stops, which is the best way to recognize it if the spectrogram shows a closure and burst. F_2 and F_3 of the transition phonemes is downward pointing.

- **/d/ and /t/:** Formants starts forming at $1800Hz$ and $2800Hz$ which moves to vowel, nasal or approximant.

- **/g/ and /k/:** /g/ is the strongest of the voiced stops, and can contain multiple bursts of energy. The formants usually come together in a triangular pattern, also called a velar pinch.

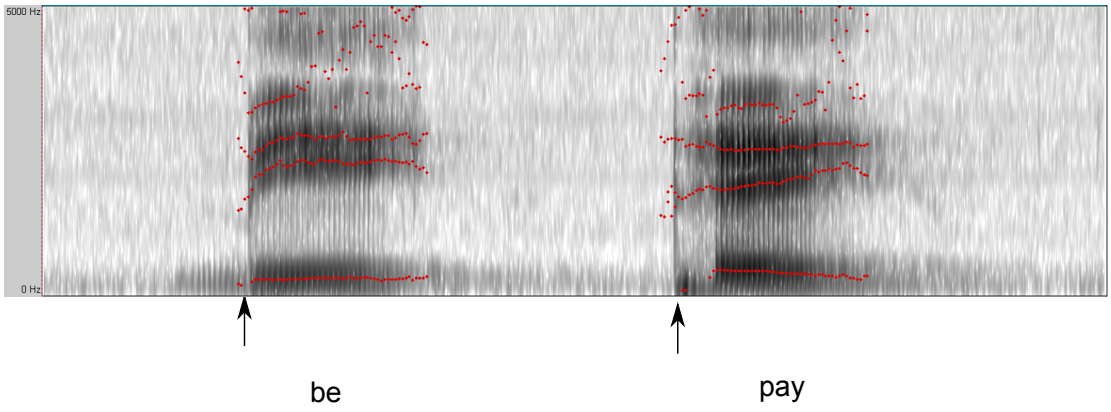

Figure 5.11: Spectrogram example of /b/ and /p/

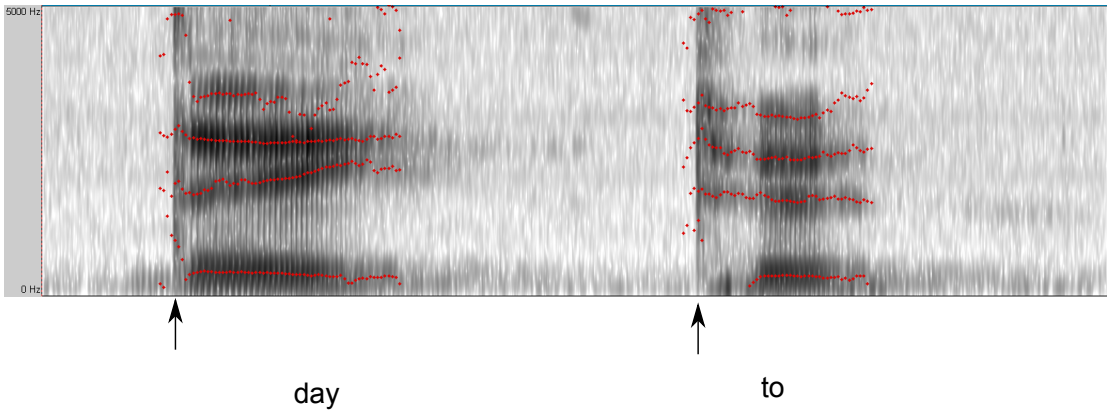

Figure 5.12: Spectrogram example of /d/ and /t/

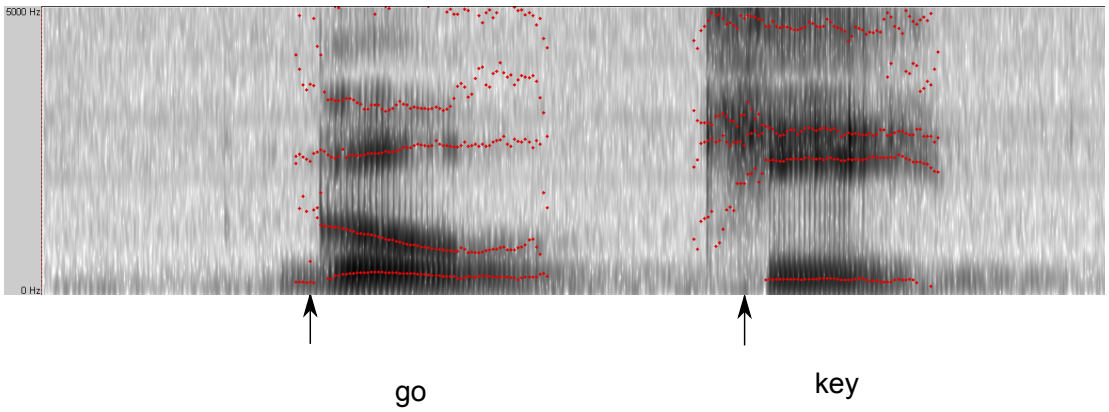

Figure 5.13: Spectrogram example of /g/ and /k/

5.2.7 Affricates

Since affricates starts as a stop, and end as a fricative, the spectrogram has similar clues to those groups.

- **/tʃ/:** A short burst which looks like a /t/ followed by a frication as an /ʃ/ where the frication is strongest in the range $2500 - 3000Hz$.
- **/dʒ/:** Starts as a /d/, ends in a /ʒ/, except the frication is asymmetrical, and the strongest energy is in the range $2500 - 3000Hz$.

5.3 Experiments with Spectrograms

This section will look at some of the problems experienced with spectrogram reading. The word "see" will be used in most examples because of the high energy in the phoneme /s/ and the easily recognizable formants of the phoneme /i/. All examples are made with the program Praat[2], recorded with stereo sound at 44100 samples per second. The spectrogram is shown in the range $0 - 8000Hz$ to see the energy in the higher frequencies.

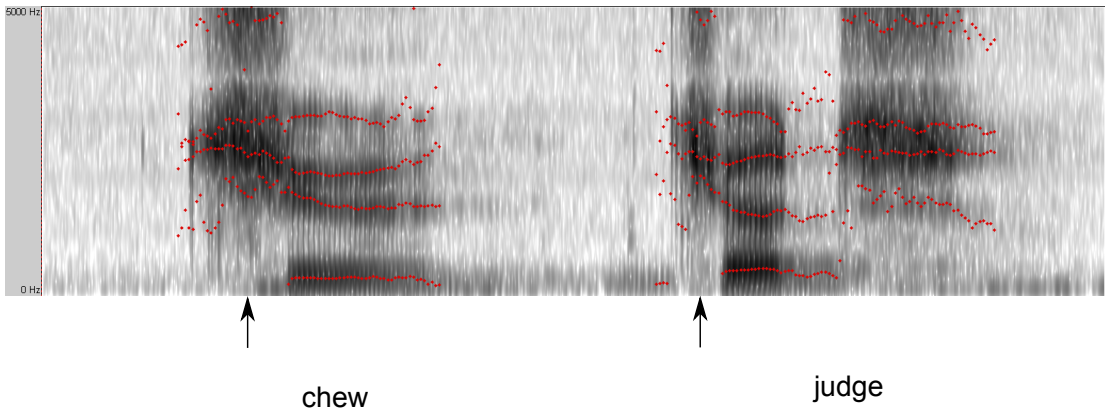

Figure 5.14: Spectrogram example of the affricates

5.3.1 Distance From Source

As the distance increases from the microphone, the signal recorded decreases in energy. Figure 5.15 and Figure 5.16 were recorded in a quiet environment with no background noise from heard by the human ear. The arrows show where the word starts and ends.

Phonemes /s/ and /i/ can be recognized in both samples, but the reduced energy does not make the word stand particularly out, causing Praat to compute formants of the background noise as well. Comparing the phoneme /i/ with the one produced in Figure 5.1, the formants are still in the correct ranges, but the energy is much less.

5.3.2 Noise

The background noise used in these experiments was hard rock music. Noise will be present at the spectrogram. If the noise is at the same loudness as the speaking source, as seen in Figure 5.17, the relevant information will seem to blend with the noise. It was still possible to hear the word ”see” in the audio example without problems.

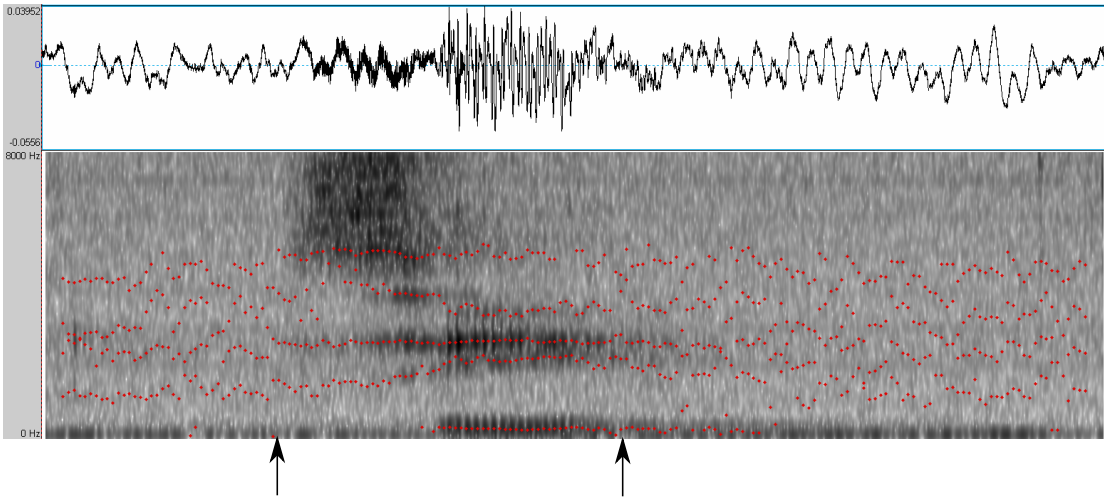

Figure 5.15: Spectrogram and waveform of the word ”see” with the microphone 1 meter from source

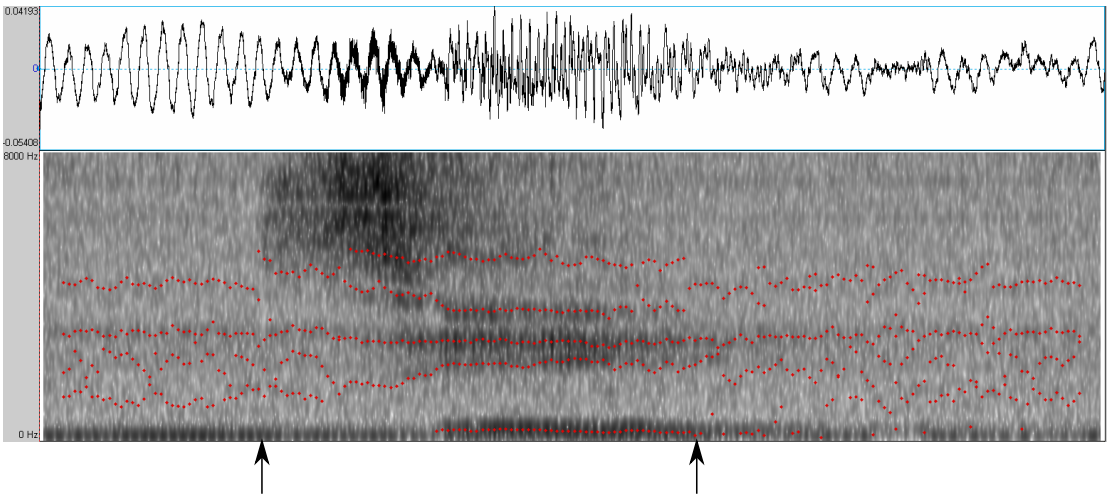

Figure 5.16: Spectrogram and waveform of the word ”see” with the microphone 2 meter from source

In Figure 5.18, the noise was not as loud as in the previous example. It is possible to distinguish the word "see" in the spectrogram without much difficulty. But the spectral noise is still present in the spectrogram.

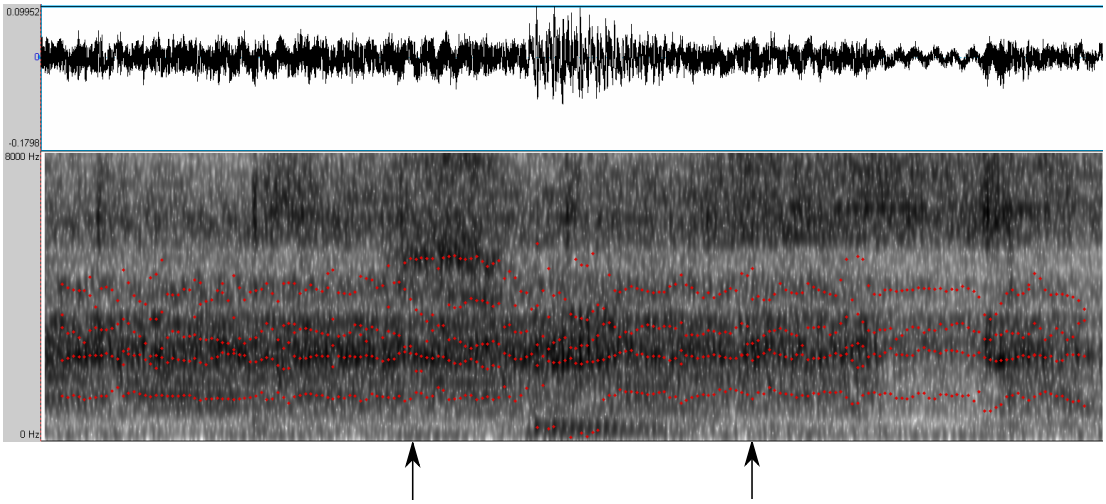

Figure 5.17: Spectrogram and waveform of the word "see" with loud noise

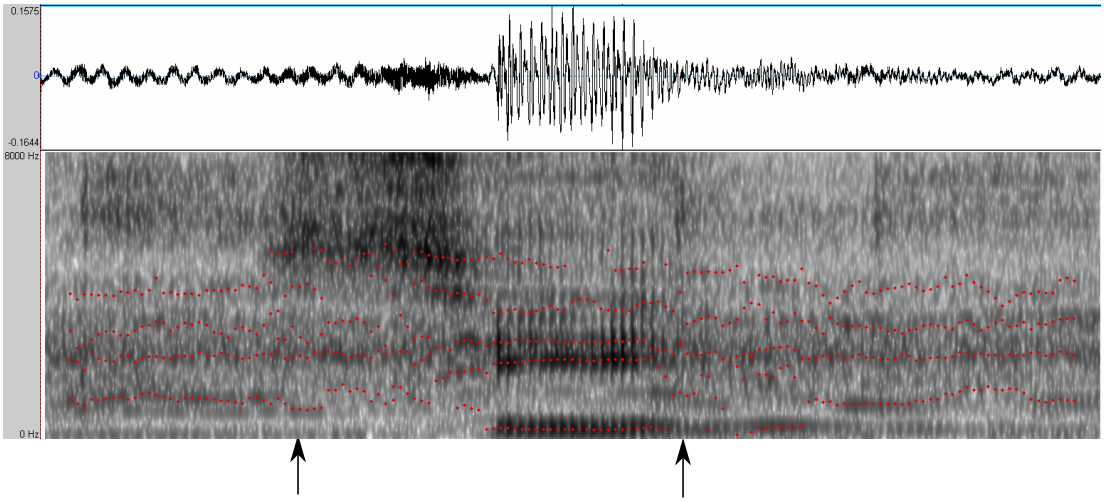

Figure 5.18: Spectrogram and waveform of the word "see" with the microphone 2 meter from source

5.3.3 Length of Utterance

The length of the utterance affects the spectrogram. In Figure 5.19, the first utterance was pronounced in 0.21 seconds, the second took 1.38 seconds. The spectral clues are still the same, but the various speaking rate of different people must be considered when looking at spectrograms.

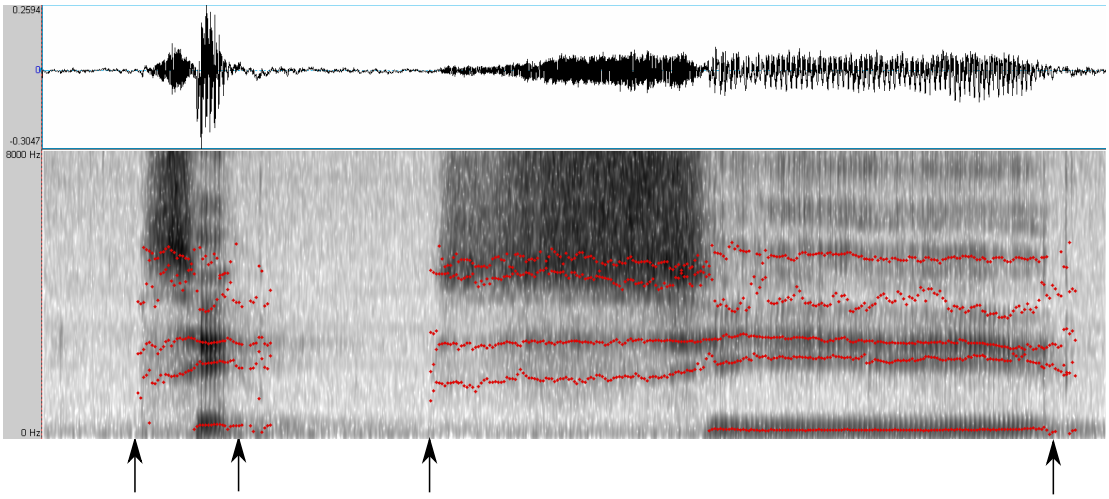

Figure 5.19: Spectrogram and waveform of the word ”see”, pronounced fast then slow

Chapter 6

Discussion

6.1 Which Features are Best Suited

6.1.1 Waveforms

Waveforms are one of the simplest features of speech on the computer, but it is impossible to recognize speech directly through a waveform. Some elements can be recognized, as stops and fricatives, but there are several problems. Two waveforms for the same word uttered by the same speaker are often very different. The amplitude and length of the waveform will vary a lot depending on how loud and fast the speaker is speaking. It does not show enough speech relevant data.

6.1.2 Mel-Frequency Cepstral Coefficients and Cepstrum

These features extract a lot of speech relevant data, but offer very little visual meaning when looked at. They are often used as input to recognition algorithms.

6.1.3 Spectrograms and Formants

A spectrogram also varies when created from two different samples, but a spectrogram contains enough information to recognize spectrograms with some training. Spectrograms offer a good visual recognition system.

But a spectrogram is very affected by noise. The speaker should speak directly into a microphone one at the time to get the best results. Different people pronounce words at different rates and frequencies, especially apparent with men and women, some samples are needed to fully differentiate the vowels and other spectral clues.

Spectrograms are the feature extraction technique chosen as the one with the most potential, and a lot of work has gone into learning how to recognize spectrogram to see if this is a feasible technique. The downside to recognizing speech by a spectrogram is that the words should be spoken directly into the microphone without any background noise for the best results. The practicality of using a real time spectrogram reader for a person who is deaf on a daily basis is also in question. The person has to look at a screen or interface to get the information, and the training required is extensive.

6.2 Algorithms for Speech Recognition

Because speech is very dynamic, most algorithms are based on the software learns and adapts to the speaker. This is very apparent in hidden Markov models and artificial neural networks. Algorithms based on these algorithms will tend to recognize speech similar to what they were taught, and not be very speaker independent.

Hidden Markov models are the most used algorithm in speech recognition at the present time. This is reflected in the extensive examination of the algorithm in Section 3.4. Artificial neural networks and hybrid systems are also in use, but the extensive learning aspect present when using artificial neural networks make hidden Markov models a better algorithm to achieve speaker independence.

Dynamic time warping compares two samples independent from time. Even

though there is over 10000 syllable in the English, as explored in Section 2.2.1, only 500 of these are used 80 percent of the time. It could be feasible to compare 500 samples for each syllable, but there was not enough time to calculate the run time for dynamic time warping, or if it is feasible to extract syllable in continuous speech.

6.3 The Split Between Human and Machine

Humans are superior to machine when it comes to understanding speech. When young children speak, humans are often able to comprehend the meaning, even though the sentences are incomplete and phonemes are pronounced wrong. An advice for teachers and parents whom child have not fully developed speech or have a speaking disability, is to carry context card which lists the things the child likes to talk about to help comprehension [41]. The same idea can be applied to a person who is deaf. Context cards for different situations and persons to assist an application.

Praat is able to record and analyze speech files, but there is no real time support. It is well suited for learning spectrogram reading, and see the results of the various feature extraction techniques discussed in Section 3.2.

Hidden Markov model toolkit showed a lot of potential when studying how it worked before trying to implement it, but as a result of not being able to get it to work properly; it could not be fully explored.

CSLU toolkit is a bit complex for the idea behind this thesis, and the lack of open source and documentation make it difficult to create an application based on feature extraction. It is not designed to work on real time data, so making a stand alone application based on the toolkit is not considered feasible.

6.4 Future Work

This thesis lay the groundwork for another master thesis or Ph.D. All feature extraction techniques, algorithms and programs are well documented by the

scientific community, and the sources given should be consulted for more information.

A real time spectrogram reader can be developed, but the negative aspects discussed under Section 6.1.3 means this is better used as a training tool than a practical application.

To use Mel-frequency spectral coefficients can be used as input to an application which extracts the speech relevant information and displays them to a user as vibrations, colors or another interface. More research about how little information is needed to preserve comprehension is needed if this approach is pursued.

Hidden Markov models are the most used algorithms at a higher level than feature extraction. The implementation of hidden Markov models for speech recognition is difficult, and an already existing toolkit could be used as a basis for further implementation.

6.5 Conclusion

A split between human and machine is possible, as demonstrated by spectrograms that was found to be the best simple feature extraction method for visual recognition of speech through a computer interface. An application for a deaf person to recognize speech is feasible based on real time spectrogram reading, but such an application would not be practical. Getting the computer to perform more of the recognition based on hidden Markov models is the most promising approach.

Bibliography

[1] Lawrence Rabiner and Biing_Hwang Juang. *Fundamentals of Speech Recognition.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[2] Paul Boersma and David Weenink. Praat. `http://www.fon.hum.uva.nl/praat/`, Accessed 04.01.2010.

[3] Gorden. E. Peterson and Harold L. Barney. Control methods used in a study of the vowels. *The Journal of the Acoustical Society of America*, 24(2):175–184, March 1952.

[4] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processes.* Pearson Education Inc, Upper Saddle River, New Jersey 07458, 4th edition, 2007.

[5] Henyk Hermansky. Perceptual linear predictive(plp) analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752, April 1990.

[6] Gülin Dede and Murat Hüsnü SazlI. Speech recognition with artificial neural networks. *Digital Signal Processing*, In Press, Corrected Proof:–, 2009.

[7] Speaker& Speech Recognition. Philip jackson. `http://info.ee.surrey.ac.uk/Teaching/Courses/eem.ssr/`, Accessed 07.02.2010.

[8] Willem J.M. Levelt. Models of word production. *Trends in Cognitive Sciences*, 3(6):223–232, June 1999.

[9] James L. Flanagan. *Speech Analysis Synthesis and Perception.* Springer-Verlag, 1972.

[10] Eugene E. Loos, Susan Anderson, Dwight H. Day Jr., Paul C. Jordan, and J. Douglas Wingate. Glossary of linguistic terms. `http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/contents.htm`, Accessed 23.02.2010.

[11] Dr. James Glass and Prof. Victor Zue. Automatic speech recognition. http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-345Automatic-Speech-RecognitionSpring2003/CourseHome/index.htm, Accessed 01.10.2009.

[12] April McMahon. *An Introduction to English Phonology*. Edinburgh University Press, 22 George Square, Edinburgh, 2002.

[13] David G. Stork and Marcus E. Hennecke. *Speechreading by Humans and Machine*, volume 150 of *F*. Springer, 1995.

[14] Tim Carmell. Spectrogram reading. http://cslu.cse.ogi.edu/tutordemos/SpectrogramReading/spectrogram_reading.html, Accessed 04.10.2010, 2009. Oregon Health & Science University.

[15] Dave Marshall. Implications of sample rate and bit size. http://www.cs.cf.ac.uk/Dave/Multimedia/node150.html, Accessed 15.02.2010.

[16] Ivan Selesnick. Short time fourier transform. http://cnx.org/content/m10570/latest/, Accessed 14.01.2010.

[17] Jont B. Allen and Lawrence R. Rabiner. A unified approach to short-time fourier analysis and synthesis. *Proceedings of the IEEE*, 65(11):1558–1564, November 1977.

[18] Spectrogram. http://www.mathworks.com/access/helpdesk/help/toolbox/signal/spectrogram.html, Accessed 15.01.2010.

[19] Fredric J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1), January 1978.

[20] John_Paul Hosom. Automatic speech recognition with hidden markov models. http://www.cslu.ogi.edu/people/hosom/cs552/, Accessed 09.01.2010.

[21] Julius O. Smith. Mathematics of the discrete fourier transform (dft) with audio applications, second edition. https://ccrma.stanford.edu/~jos/mdft online book, Accessed 15.02.2010.

[22] Sidney Wood. What are formants? http://person2.sol.lu.se/SidneyWood/praate/frames.html, Accessed 08.01.2010.

[23] John Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 62(4):561–580, April 1975.

[24] Ben J. Shannon and Kuldip K. Paliwal. A comparative study of filter bank spacing for speech recognition. *Microelectronic Engineering Research Conference*, 2003.

[25] Shin'ichi Satoh Kiyoharu Aizawa, Yuichi Nakamura. *Advances in Multimedia Information Processing - PCM 2004*. Springer-Verlag, 2004.

[26] UCL Department of Phonetics and Linguistics. Lecture 10: Speech signal analysis. `[http://www.phon.ucl.ac.uk/courses/spsci/matlab/lect10.html`, Accessed 07.02.2010.

[27] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Ieee Transactions on Acoustics, Speech, and Signal Processing*, 26 Issue 1:43–49, February 1978.

[28] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), February 1989.

[29] Chee Peng Lim, Siew Chan Woo, Aun Sim Loh, and Rohaizan Osman. Speech recognition using artificial neural networks. *Web Information Systems Engineering*, 1(1):419–423, June 2000.

[30] Franco Mana Roberto Gemello and Dario Albesano. Hybrid hmm/neural network based speech recognition in loquendo asr. `http://www.loquendo.com/en/brochure/Speech_Recognition_ASR.pdf`, Accessed 07.02.2010.

[31] Tariq Mehmood Abdul Ahad, Ahsan Fayyaz. Speech recognition using multilayer perceptron. *Students Conference, ISCON '02. Proceedings. IEEE*, 1:103–109, August 2002.

[32] Cambridge University Engineering Department. Htk web site. `http://htk.eng.cam.ac.uk/`, Accessed 16.11.2009.

[33] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. *The HTK Book (for HTK Version 3.4)*. Cambridge University Engineering Department, March 2009.

[34] John-Paul Hosom. Cslu toolkit packages: Viterbi. `http://cslu.cse.ogi.edu/toolkit/docs/2.0/pkg/viterbi/viterbi.html`, Accessed 05.02.2010.

[35] The Toolkit Developers Guide. Johan schalkwyk. `http://cslu.cse.ogi.edu/toolkit/old/old/version2.0a/documentation/devsh/develop.html`, Accessed 05.02.2010.

[36] Johan Schalkwyk and Mark Fanty. The cslu-c toolkit for automatic speech recognition. http://cslu.cse.ogi.edu/toolkit/old/old/version2.0a/documentation/csluc/Cdoc.html, Accessed 10.02.2010.

[37] Center for Spoken Language Understanding Oregon Graduate Institute of Science & Technology. Ipa, worldbet, and ogibet english broad phonetic labels. http://cslu.cse.ogi.edu/tutordemos/nnet_training/Worldbet-English.pdf, Accessed 05.02.2010.

[38] Ron Cole John-Paul Hosom and Mark Fanty. Speech recognition using neural networks at the center for spoken language understanding. http://www.cslu.ogi.edu/tutordemos/nnet_recog/recog.html, Accessed 04.02.2010.

[39] John-Paul Hosom, Jacques de Villiers, Ron Cole, Mark Fanty, Johan Schalkwyk, Yonghong Yan, and Wei Wei. Training hidden markov model/artificial neural network (hmm/ann) hybrids for automatic speech recognition (asr). http://cslu.cse.ogi.edu/tutordemos/nnet_training/tutorial.html, Accessed 05.02.2010.

[40] Robert Hagiwara. How to read spectrograms. http://home.cc.umanitoba.ca/~robh/index.html, Accessed 16.02.2010.

[41] Gwen Lancaster. *Developing speech and language skills: a resource book for teachers*. David Fulton Publishers, 1 edition, August 2007.

Appendices

Appendix A

IPA for English

/i/	eve	/l/	let
/ɪ/	it	/m/	me
/ɝ/	bird	/n/	no
/u/	boot	/ŋ/	sing
/ʊ/	foot	/v/	vote
/e/	hate	/f/	for
/ɛ/	met	/ð/	then
/ʌ/	up	/θ/	thin
/o/	obey	/z/	zoo
/ɔ/	all	/s/	see
/æ/	at	/ʒ/	azure
/a/	father	/ʃ/	she
/aɪ/	my	/h/	he
/aʊ/	now	/b/	be
/ei/	day	/p/	pay
/oʊ/	no	/d/	day
/ɔi/	boy	/t/	to
/iʊ/	few	/g/	go
/w/	we	/k/	key
/r/	read	/tʃ/	chew
/j/	you	/dʒ/	judge