Automatic Speech Recognition of Native Dutch Speakers with Different Age and Gender

Wendy Clerx
Delft University of Technology

October 2007

Man-Machine Interaction Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands
E-mail: wendy@mmi.tudelft.nl
Student number: 1105469

Date of Presentation
14th November 2007

Graduation Committee
ir. P. Wiggers drs. dr. L.J.M. Rothkrantz
dr.ir. J.F.M. Tonino
prof. dr. C.M. Jonker

Keywords: Automatic Speech Recognition, Language Porting,
Dutch Language, Flemish, Phonetics, Sonic, N-Best,
Perceptual Minimum Variance Distortionless Response, PMVDR

This work has been typeset with LaTeX 2_ε

Abstract

Humans are capable of estimating speaker ages by only hearing them speak. It is also well known from the field of phonetics that speaker age influences the speech signal. This has however not yet been researched for the Dutch language. In this research, the influence of age on speech is researched for both genders separately and compared with the gender differences, using Perceptual Minimum Variance Distortionless Response features. The influences of age are minimal for these features but greater than the differences between speech from different genders. Different spectral features are influenced for different phonemes. It seems unlikely that adapting speech recognizers using Perceptual Minimum Variance Distortionless Response features will lead to much improvement.

Furthermore, this thesis describes the process of creating a Dutch automated speech recognition system, using the Sonic large vocabulary continuous speech recognition system as a basis. The system achieves a recognition rate of 64.6% on the broadcast news task from the N-Best project. The porting process is described in detail and provides an accurate introduction to the practice of porting speech recognition systems.

Contents

Chapter 1

Introduction

"It is not easy to wreck a nice beach"
"It is not easy to recognize speech"

Homophonous sentences:
two sentences that have an almost identical
pronunciation and are hard to differentiate
between acoustically

This chapter gives a brief introduction into the field of ASR and some relevant knowledge on the Dutch language. It also includes the problem statement and goals of this research.

In the legendary movie "2001: A Space Odyssey" from the year 1968, we saw an intelligent computer named Hal that, on top of being highly intelligent and cunning, was able to understand human speech perfectly, even better than a human could. This level of speech understanding by computers has ever since been the dream of automated speech recognition (ASR) researchers. Would it not be far more convenient to be able to tell a computer what you want it to do instead of using a more artificial input device such as a keyboard or number pad? Especially for people who are not very comfortable with new technological products and computers, it could be far easier if the interface were voice-oriented. There also are situations where it is even dangerous to interface with a computer through a keyboard, for example when using a navigation system while driving a car. Unfortunately, 40 years later we still have not managed to accomplish ASR at a human level. As it turns out it is in fact not easy to recognize speech.

1.1 Automated Speech Recognition

Why is ASR hard to accomplish? There are various reasons why this is a complicated task. Speech, captured in speech signals, has a rich variation between different speakers. Physiological aspects such as length of the vocal cords, shape of the mouth cavity, position of the teeth all influence the speech signal that a speaker produces. Other factors that differentiate speech include social aspects. Native speakers can hear differences between standard Dutch pronounced by speakers from different provinces in the Netherlands and even between speakers

from certain cities. An acquaintance from the city of Zoetermeer went to the US and surprisingly was recognized there as originating from Zoetermeer simply by his specific pronunciation of the letter *r*. Humans can also distinguish between male and female speech and even between speech from different age categories [Schötz, 2007]. For example, when we answer the phone, we can tell whether it is a child or an elderly person calling. All these aspects influence the speech signal and complicate matters for a computer. An ASR system should be able to recognize all these different pronunciations of the same sounds and then conclude that they are in fact the same sounds.

But even when one speaker pronounces the same word twice, there will be differences in the speech signal. If a speaker has a cold, he sounds different. His emotions also influence the pronunciation. Speech from a certain speaker is even different at different times of the day [A.C.M. Rietveld, V.J. van Heuven, 1997], although it is not mentioned why. The duration of the different sounds that constitute the word in question, will also vary slightly. Because speaking involves a continuous motion of the articulators —tongue, teeth, lips— the sounds themselves even differ depending on which sounds precede and follow it. This effect is known as coarticulation.

Besides the influence of the speaker, there also are external factors that complicate matters. There will be different amounts of noise that obscure the speech signal and different input devices and encoding schemes that influence the speech signal. This is known as channel variability.

Apart from these more acoustical problems, there also are other complications. When looking at a transcript of spontaneous speech, it can readily be seen that there are many repetitions, filler words, mispronunciations, errors that are being corrected and other mistakes that complicate matters. Very often, spontaneous sentences will not be grammatically correct. An example sentence from a Dutch speech collection: "ja [incomprehensible] je hebt uh je heb hier in de tuinkamer heb je d'r eentje..", which in a translated form would resemble "yes [incomprehensible] there's uh there is here in the garden room there's one". When we look at the example of a dictation task, human transcribers are able to filter out most of the previously named errors. With software, it is a different story. A tester from a newspaper gave the leading voice detection software package a try and came to the following conclusion [Rob Pegoraro, 2006]:

> With a program such as Dragon, you must reprogram yourself to think in complete sentences, avoiding "ers" and "ums" and maintaining a steady volume and cadence. You have to let the software train you before it will work to its fullest.

Another difficulty is the vocabulary. The vocabulary of a modern language changes continuously, with new words entering, old ones becoming obsolete and fashion words occur and disappear sometimes so fast that they do not even become an entry in the dictionary. And current speech recognizers rely on dictionaries to determine which sound sequences form a word in a specific language and which do not. Certain languages, such as Dutch, also allow the formation of a new word by combining two existing words, following certain rules. This makes it hard to capture a vocabulary of a language even without the constant changes.

Many of these problems can be categorized under the header of 'too much variation', on very different areas. During this research, we will not try to tackle

all these problems simultaneously but focus on the acoustic variation between speakers. Currently, the acoustic variation between between speakers of different characteristics is mostly ignored. Speech recognizers are trained on corpora of speech from a balanced group of speakers, using statistical averages calculated over the group as a whole to tune the acoustic speech models to. These models can then be used to recognize speech from all speakers. Limiting this variation by creating specialized acoustic speech models for different speaker groups could improve the performance of ASR systems. Most current ASR systems already include provisions for different genders. However, it is indicated in the literature of phonetics that age also influences the speech production organs and thus leads to differences in speech acoustics [A.C.M. Rietveld, V.J. van Heuven, 1997], [Schötz, 2007]. Indeed, human listeners are able to differentiate between speech from speakers with different genders and even ages [Harry Hollien, 1987], [Schötz, 2007]. A special ASR system was also created for Swedish children [Daniel Elenius and Mats Blomberg, 2004]. However, the effects of these biological factors have not yet been researched for Dutch speakers. We will thus look at the acoustical differences between different age categories and genders specifically for the Dutch language, using the speech data from the recently finished Corpus Gesproken Nederlands (CGN, the spoken Dutch corpus). This will allow us to conclude whether different models for different age groups could be helpful for ASR of the Dutch language.

1.2 The Dutch Language

Before starting ASR research on the Dutch language, some basic knowledge of this language is indispensable. Dutch is not one homogeneous language: there are two distinct main dialects. In the Netherlands people speak Northern Dutch and in Flanders, the Dutch-speaking part of Belgium, people speak Southern Dutch, also known as Flemish. The most important differences between these two versions of the language can be found in the pronunciation.

There are some differences in the pronunciation of consonants, vowels and diphthongs [Véronique Hoste, Walter Daelemans, Steven Gillis, 2004]. For example, Dutch speakers sometimes use unvoiced consonants where Flemish speakers use voiced consonants. This occurs in the word 'gelijkaardig' (similar), which starts with an initial voiceless fricative (x) in Dutch and with a voiced velar fricative (ɣ) in Flemish.

The differences between the two vocabularies are small: native Dutch speakers use the same dictionaries in which some words have the annotation 'Northern Dutch' or 'Southern Dutch'. In fact the well-known dictionary 'Van Dale' mentions only 2332 Flemish words in a collection of 311245 Dutch words [Prof. dr. Dirk Geeraerts, 2000]. However, the usage of some of these words differs: in Northern Dutch, the personal pronoun 'ge' (thou) is archaic and rarely used, mostly in a solemn context. In Southern Dutch however, it is used very often as the standard personal pronoun for the second person singular and lacks the formal connotation, similar to the usage of 'you' in English.

Unlike American and British English, there are no differences in grammar or spelling customs, but there are however some occasional differences in style [William Z Shetter, 2002]. Overall Dutch and Flemish speaking people are able to understand each other without too much added effort. However, when

television stations in Flanders and the Netherlands broadcast each other's fictional television series, subtitles are often added to bridge the differences in pronunciation.

These differences require special attention when constructing a speech recognizer. It has been shown that adaptation of acoustic speech models to a dialect or accent will greatly improve the recognition results. It was indicated that for English pronounced with a Spanish accent, retraining results in a far higher recognition rate than with normal English pronunciation models [Ayako Ikeno, Bryan Pellom, Dan Cer, Ashley Thornton, Jason Brenier, Dan Jurafsky, Wayne Ward, William Byrne, 2003]. Retraining should thus also prove to be very useful for the Northern and Southern Dutch models.

1.3 N-Best and TU Delft

At this moment, it remains unclear how well automated recognition of Dutch speech works. This would require a standardized task and standardized data to base the ASR system on. There has been research on Dutch ASR but using different ASR systems for solving different tasks, which prohibits comparison of the results. So it would be useful to have a benchmark test to compare with: agree on one specific task using the same speech data as a basis to compare results with. For this reason, a project is started named 'N-Best': a joint Dutch-Flemish project with the goal to "evaluate the performance of present-day large vocabulary continuous speech recognition (LVCSR) systems for the Dutch language". During this project, all participating universities and companies create their own automated speech recognition (ASR) system, after which a benchmark is conducted. This research is conducted as a part of the N-Best project, which is discussed in more detail in section chapter 3. During this specific research, a Dutch LVCSR system is created that can be used as a basic system for ASR research as well as during the N-Best project.

Delft University of Technology has conducted multiple research projects in the field of ASR, such as [Pascal Wiggers, Leon J.M. Rothkrantz, 2006] and [Leon J.M. Rothkrantz, Pascal Wiggers, Jan-Willem A. van Wees, and Robert J. van Vark, 2004]. However there is no standard ASR system available to use in these different projects at the TU Delft. An ASR system to be used as the basic system across different projects would eliminate the need to start each project by choosing an appropriate ASR system and training it. This would give more time to the researchers to actually spend on the research. It would make it easier to test out new ideas and would thus give an extra boost to the ASR research. It is also necessary for the N-Best benchmark test to have a LVCSR ready in time, one that can recognize broadcast news and one for the recognition of spontaneous speech, which is the hardest ASR task there is. Since TU Delft has not yet finished an ASR system of their own and has never built a separate speech recognizer for Flemish or for spontaneous speech, an existing speech recognizer is needed to enter the project with.

1.4 Research Objectives

There are two main objectives in this research project:

- to prepare a Dutch LVCSR system for the TU Delft to use as a basic system for ASR research and during the N-Best project, for the recognition of broadcast news as well as for spontaneous speech.

The TU Delft needs a Dutch LVCSR in time to apply to the test data from the N-Best project. A new LVCSR system is currently under development at the TU Delft but it might not be ready in time for this project. Out of two popular LVCSR systems, Sonic [SonicWeb] was chosen to be ported to the Dutch language and be used by Delft during this project [Clerx, 2007], if the new system from Delft is not ready in time.

- to research whether the acoustic differences between Dutch speakers from different age categories, for male and female speakers separately, are sufficiently large to potentially be useful in the field of ASR.

Therefore, we investigate the acoustic differences between different Dutch speakers; more exactly the differences between speakers of different age and/or education level. If the acoustic differences are large enough, these speaker categories could improve LVCSR systems' recognition rate by incorporating specialized acoustic models for each category.

1.5 Thesis Structure

This thesis is written as a partial fulfillment of the requirements for obtaining a Master of Science degree. After this introductory chapter, the thesis continues with a chapter on the theories behind this research, chapter 2. Next, the N-Best project is introduced in chapter 3. Chapter 4 presents the tools that are used during this project. The process of porting the Sonic LVCSR to the Dutch language, is described in chapter 5. Chapter 6 describes the research on the acoustical differences between different age categories. The last chapter, chapter 7, presents the conclusions from this research and gives recommendations for further research.

Chapter 2

Theory

A language is a dialect with an army and navy

Aphorism

This research uses concepts from the speech recognition field combined with knowledge from the phonetics domain. This chapter will introduce the key concepts from these domains in two separate sections.

2.1 Basic Concepts of Speech Recognition

This introduction into the field of ASR is based on the book by [Jurafsky and Martin, 2000]. Currently, speech recognition is commonly done by calculating which phrase W has the highest probability of having been uttered, given the recorded utterance O. The probability of each phrase is calculated separately and the phrase with the highest probability is chosen to be the correct one. So we want to calculate the following probability:

$$\widehat{W} = \max_{W} P(W|O). \tag{2.1}$$

Unfortunately, it is not clear how this probability distribution, over two variables that both have a potentially infinite number of states, should be defined. Fortunately, this formula can be rewritten as follows, using the well known theory of Bayes.

$$\widehat{W} = \max_{W} P(W|O) = \max_{W} \frac{P(O|W)P(W)}{P(O)}. \tag{2.2}$$

Since we are only interested in finding the phrase W that has the highest probability and the term $P(O)$ is the same for all phrases, we can leave this term $P(O)$ out of the equation. This is very convenient since we do not know how to calculate the probability of a specific 'utterance' or 'observation sequence' $P(O)$. We now only have to solve

$$\widehat{W} = \max_{W} P(W|O) \approx \max_{W} P(O|W)P(W). \tag{2.3}$$

On first sight, not much has changed: we still have to define two distributions over an infinite number of states, but now compact models can be defined that

calculate these probabilities on the fly. The probability that each word has of having been uttered, is determined by multiplying two factors: the prior probability

$$P(W) \tag{2.4}$$

and the observation likelihood

$$P(W|O). \tag{2.5}$$

The prior probability is the probability that a specific phrase has been uttered, regardless of what was heard. More commonly used phrases thus have a higher prior probability then rarely used phrases. The other term, the observation likelihood, represents the probability that a pronouncing a certain phrase was said, sounds like the observations. The prior probability is better known as the language model while the observation likelihood is called the acoustic model. The speech recognition process is represented in figure 2.1 and will be explained in more detail in the following sections.

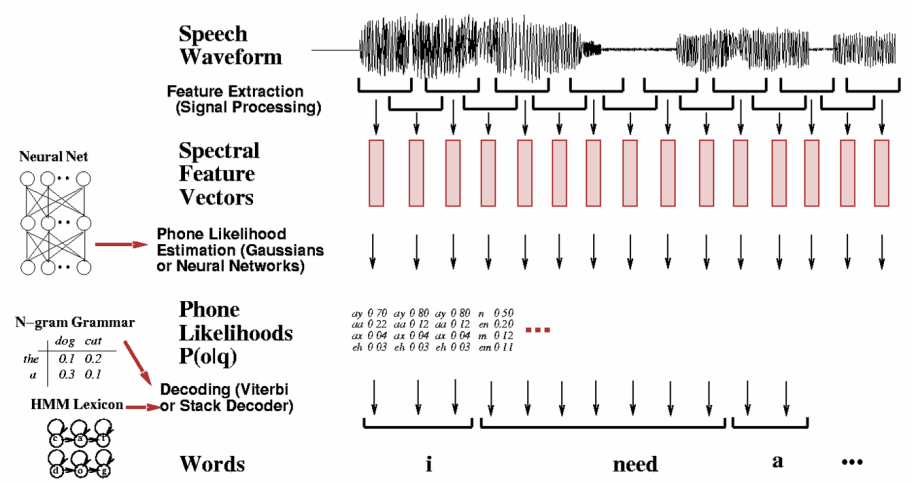

Figure 2.1: Schematic architecture for a simplified speech recognizer. The speech waveform first undergoes a signal processing step which produces a representation in spectral feature vectors. Phone likelihoods are subsequently estimated, after which a decoding step can finish the recognition process.

2.1.1 Feature Extraction and Phonemes

When performing speech recognition, the first step is to analyze the given sound wave and extract relevant data from it [Jurafsky and Martin, 2000]. This signal processing starts by dividing the sound wave in time slices from which spectral features are extracted. Spectral features indicate how much energy is present in the signal at different frequencies. Later on, these features are used to determine which sounds were uttered. Instead of the general word 'sound' we talk about a 'phoneme'. A phoneme is, according to [Huang, 2001], defined as:

> any of the minimal units of speech sound in a language that can serve to distinguish one word from another.

For example, /p/ is a phoneme in the English language, distinguishing *tap* from *tag*. These spectral features are used in combination with hidden Markov models(HMMs) to determine the probabilities for the different phonemes of having been uttered, while taking the speech waveform into consideration. This probability was introduced before as the observation likelihood $P(W|O)$. These calculations are explained in more detail in the next section.

2.1.2 Hidden Markov Models

Now we can introduce the application of HMMs in the field of ASR. The HMMs are used for phone likelihood estimation, as indicated in figure 2.1. It was mentioned above that the first step in the speech recognition process is the extraction of spectral features for each time frame. Now, for each two consecutive time frames, t and $t+1$, the HMM-based recognizer is assumed to transition from state i to state j with probability a_{ij}, or stay in state i with probability a_{ii}, and emit an observation symbol o_t with probability density $b_j(o_t)$. The observation symbol o_t is a 39-dimensional feature vector with real values. Each phoneme is typically modeled using anywhere between 3 to 12 HMM states, depending on the recognizer implementation. We will give an example of a very simple HMM in figure 2.2 from [van Dalen, 2006].

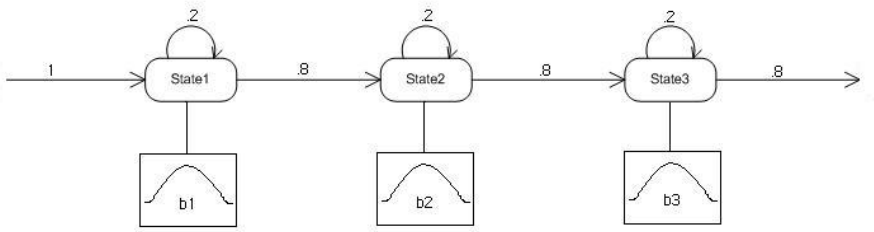

Figure 2.2: An HMM example state diagram. Three states with corresponding output distributions are interconnected by transitions with transitional probabilities.

This particular HMM has 3 states. This HMM starts in state 1 with a probability of 1, meaning that the corresponding phoneme always starts in state 1. From here the HMM can continue, after one time slice, to state 2 with a transitional probability of 0.8 or stay in state 1 with a transitional probability of 0.2. After the last state of this phoneme, state 3, the next HMM is entered. A specific output distribution b_1 corresponds with state 1. This output distribution is Gaussian and is thus defined by its mean μ and standard deviation σ.

The feature vectors that were extracted from the audio time slices, are used as input for the output probability distributions b_1, b_2 and b_3 to calculate the observation likelihood $P(O|W)$. This is done as follows: suppose that we know that either only the phoneme /p/ or the phoneme /b/ has been pronounced in our input speech waveform. For each of these phonemes, we have a HMM with three states, representing the acoustics of three different stages in the pronunciation of the phoneme in question (the onglide, the pure state and the offglide). We now want to determine the probability of each of these phonemes

matching with the speech signal. Then we can simply pick the phoneme with the highest probability. This probability is calculated as follows. We use the well-known Gaussian probability density function:

$$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{2.6}$$

As the feature vectors are 39-dimensional, a multivariate Gaussian density function has to be used. In practice however, the individual dimensions of the vectors are considered independent, i.e. the covariance matrix of the distribution is diagonal. As a consequence, each phoneme state can be represented by a set of Gaussians, one for each feature vector element. In other words, every state of every HMM has its own Gaussian distribution. The Gaussian density function is then applied to every feature dimension of every feature vector in combination with the corresponding HMM state. Multiplying these probabilities results in the observation probability for the current phoneme state given the current feature vector (i.e. the probability that pronouncing this phoneme state would lead to the feature vector under consideration). When we have a hypothetical phoneme state sequence, we can calculate a list of individual phoneme state observation likelihoods. To obtain the actual phoneme sequence probability, we first multiply the probability of starting in the starting state of the sequence with the transition probabilities corresponding with the state transitions producing the state sequence. Now we can multiply these individual phoneme state observation likelihoods with the transition probabilities corresponding with the consecutive phoneme states to obtain the sequence observation likelihood.

Not all phoneme sequences are allowed: only phoneme sequences constituting actual words are allowed. A phonetic lexicon is used by the ASR system to look up which phoneme sequences correspond with actual words. The corresponding phoneme HMMs can then be combined into one bigger word HMM by simply linking the HMMs together with new transitions.

But how do we know for which sequences the probability should be calculated? To try out all legitimate word sequences from scratch would be inefficient to say the least. This is where the Viterbi algorithm proves its worth. This algorithm will be explained in section 2.1.4, but first we need to introduce the notion of the language model.

2.1.3 The Language Model

The language model from equation 2.4, tells the ASR system what the likelihood is of a certain word being spoken, regardless of the audio signal. It basically tells us which word sequences are more likely than others to occur in a specific language. The probability of a word sequence is defined to be equation 2.7:

$$p(W) = p(w_1 w_2 \dots w_n). \tag{2.7}$$

The probability of this word sequence can be calculated by multiplying a sequence of probabilities for shorter word sequences, as can be seen in equation 2.8:

$$p(W) = p(w_1)p(w_2|w_1)p(w_3|w_1 w_2)\dots p(w_n|w_{n-1}w_{n-2}\dots w_1). \tag{2.8}$$

To calculate the probability for the last word in equation 2.8, the complete word history has to be taken into account. For longer word sequences, it is practically impossible to find good estimates for these probabilities. Reliably estimating them would require larger amounts of training texts than can be collected in practice. This is why we have to limit the amount of preceding words we take into consideration and assume that only the N preceding words influence the probability of the next word. When N equals 2, this results in the following formula:

$$p(W) = p(w_1)p(w_2|w_1)p(w_3|w_1w_2)\dots p(w_n|w_{n-1}w_{n-2}). \tag{2.9}$$

This independence assumption results in N-grams, more specifically unigrams when $N = 0$, bigrams when $N = 1$ and so on. Unigrams contain probabilities for each separate word, indicating how probable it is that this word occurs, regardless of the rest of the phrase. Bigrams consider the probability of a certain word occurring while taking the previous word under consideration, as was done in equation 2.9. In order to use unigrams, we need to know the probabilities for the occurrences of each word in the dictionary. These unigram counts are calculated by taking a large text to use as training data, and then simply counting how often each word in the vocabulary occurs in the training text. Dividing these totals by the total number of words in the training text, gives us estimates for the probabilities we need. Using these relative frequencies to estimate probabilities is an application of maximum likelihood estimation since the likelihood of the training set is maximized given the model. This approach can be expressed as equation 2.10:

$$P(w_i) = \frac{C(w_i)}{\sum_n C(w_n)}, \tag{2.10}$$

where $C(w_i)$ is the number of times word w_i occurs in the corpus.

When we decide to use bigrams —and unigrams, for the first word of a sentence— we look at how probable a certain word is to occur following the previous word. For this we count how often each two-word sequence occurs in the training text and divide this by the total number of two-word sequences available, see equation 2.11. This provides us with a more accurate estimate of the prior probabilities than unigrams do. This approach can be expanded to trigrams, fourgrams and so on. This results in the general formula in equation 2.12. Theoretically, it would be best to look back at as many words as possible and to take the whole of the preceding text into account. It would however be impossible to collect sufficient training data to make a reliable estimate of these probabilities. So a trade-off must be made and currently, trigrams are most commonly used.

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}. \tag{2.11}$$

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}. \tag{2.12}$$

When using a training text for estimating these probabilities as described above, the words in the vocabulary that do not occur in the training text get

a probability of 0. With them, all sentences in which these words occur also have a probability of 0. With a probability of 0, these words would thus never be recognized by the ASR system; they would in practice be removed from the vocabulary. But since these words are in fact existing words, they should be viable to be recognized by an ASR system and should have a small probability higher than 0. The techniques that are used to assign them a small, non-zero probability are called smoothing. This involves slightly reducing the probabilities of the words that do occur and augmenting the probabilities of the other words slightly so now all words in the vocabulary qualify for recognition. As a result, a portion of the probability density has been redistributed to these words. As an estimate, these words often receive a probability of about the probability words get that only occur once in the training text.

The remarkable thing is that these N-grams can also be seen as a Markov model, as illustrated in figure 2.3. The structure is as follows: after a dummy starting state, we have a set of nodes, one for each word in the dictionary. The probability of transitioning to such a word, is its unigram probability. After these first words, a second layer of nodes follows, once again one for each word in the dictionary. Each node w_1 in the first layer can have a connection with each node w_2 in the second layer; the corresponding probabilities are the corresponding bigram probabilities $p(w_2|w_1)$. This structure can easily be extracted to trigram or fourgram structures. A sentence now consists of a sequence of words while each word consists of a sequence of phonemes, which themselves consist of multiple states. In this light, a sentence can be regarded as one big HMM, containing smaller word HMMs built up from phoneme state HMMs. Now we use the Viterbi algorithm to find the most likely sequence through this giant HMM.

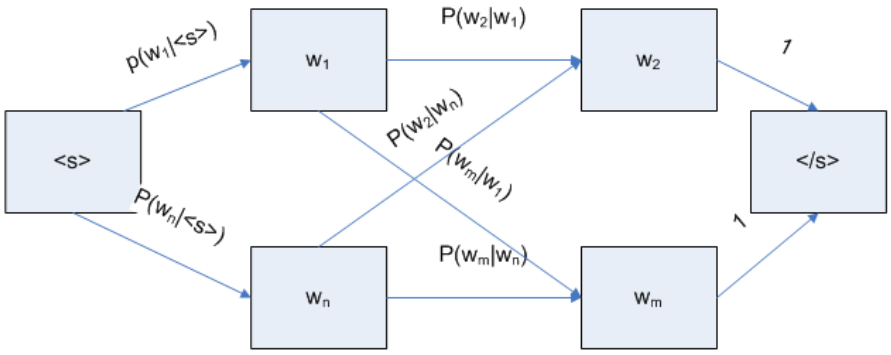

Figure 2.3: The language lodel can be regarded as a Markov model, using N-gram probabilities as the transitional probabilities.

2.1.4 The Viterbi Algorithm

The Viterbi algorithm calculates the most likely sequence through a HMM, given a sequence of observations. It can be found in figure 2.1 as the decoding stage. This algorithm needs multiple inputs:

1. a sequence of feature vectors,

2. a set of acoustic HMMs, along with state transitions and the corresponding transitional probabilities,

3. a language model, for the prior probabilities and the transitional probabilities from one word to the next,

4. a pronunciation dictionary, containing the possible words and their pronunciation as a sequence of phonemes.

The algorithm then fills a matrix with the likelihoods of all possible subsequences. They are calculated using the transitional probabilities, from the HMMs, and the language model, and observation likelihoods of single phoneme state and feature vector combinations, using equation 2.6. The algorithm also keeps track of back-pointers, which are an efficient way to store sequences. When it is finished, the Viterbi-matrix contains the probabilities of all possible sequences. Now retrieving the most probable sequence is simply a matter of looking up the highest probability a sequence has and reconstructing the sequence via the back-pointers. We have now finished the process of calculating most probable sequence as far as the observation likelihood from equation 2.5 is concerned.

Using the most likely sequence is in fact an approximation since different phoneme state sequences correspond with the same sentence. If one pronounces the same sentence two times, it is next to impossible to pronounce it in exactly the same way. Some phonemes will most likely be pronounced a bit shorter and others slightly longer. A better estimate of the sentence's probability would be achieved by adding up these sequence probabilities that correspond with the same sentence, which is done by the forward algorithm. However, the forward algorithm has to be run separately on each different sentence, which makes it very inefficient. Furthermore, the Viterbi algorithm typically comes up with the same answer.

But how do we obtain the correct HMM values? All the HMM parameters —transition probabilities, Gaussians— need to be trained to make the HMMs accurately model the speech training data. This is done with an annotated data set through a procedure similar to the language model training procedure. However, unlike the language model, the acoustic models are *hidden* Markov models, which prohibits the use of a simple maximum likelihood approach. With HMM training, a form of expectation maximization (EM) is needed. This technique iteratively applies the forward algorithm to adjust the different HMM parameters.

The reader who is interested in a more detailed introduction in this subject is advised to consult [Jurafsky and Martin, 2000] or, more briefly, [Spaans, 2004].

2.1.5 Language Porting of Acoustic Models

When porting an HMM-based ASR system to another language, a few changes have to be made. Firstly, the acoustic models need to be adapted to the phonemes in the new language. Therefore, a mapping is made defining which phonemes in the original language resemble which phonemes in the new language. Then the HMMs are retrained using speech data from the new language to re-estimate the HMM values. Of course a vocabulary has to provided for the new language and also a new language model matching this vocabulary. After the retraining process, extra training steps can be performed to make the

HMMs even more accurate for the new language. After each retraining step, a test should be conducted to see how well the HMMs perform and the retraining should continue until the models no longer improve.

2.2 Basic Concepts of Phonetics

Let's look at the definition of phonetics according to [PhoneticsWiki]:

> Phonetics is the study of the sounds of human speech. It is concerned with the actual properties of speech sounds (phones), and their production, audition and perception, as opposed to phonology, which is the study of sound systems and abstract sound units, such as phonemes and distinctive features. Phonetics deals with the sounds themselves rather than the contexts in which they are used in languages. Discussions of meaning (semantics) do not enter at this level of linguistic analysis [A.C.M. Rietveld, V.J. van Heuven, 1997].
>
> Phonetics has three main branches:
>
> - articulatory phonetics, concerned with the positions and movements of the lips, tongue, vocal tract and folds and other speech organs in producing speech;
> - acoustic phonetics, concerned with the properties of the sound waves and how they are received by the inner ear; and
> - auditory phonetics, concerned with speech perception, principally how the brain forms perceptual representations of the input it receives.

This research belongs to the domain of acoustic phonetics and is limited to the research of the sound waves. It is noted in [A.C.M. Rietveld, V.J. van Heuven, 1997] that phonetics is an experimental science and is conducted through experiments on actual speech samples. This can also be seen in this research, which owes its weight to the big amount of speech data available to be studied. We take a closer look at the different phonemes of the Dutch language and the influence of gender and age on their realizations, which are called 'allophones'. This allows us to draw conclusions on the potential usefulness of these criteria for ASR systems for the Dutch language. The common notation for phonemes is: $/r/$, while allophones have the following notation: [r]. These phonemes may seem discrete, but they are not pronounced in isolation. Consecutive phonemes influence each other since our speech organs do not instantly jump from sound to sound. Speakers pronounce text in fluent motions, allowing the consecutive phonemes to influence each other.

Speech is produced by the speech organs. In general, speech is the result from air being exhaled, influenced by the vocal cords or changes in position of the tongue and the opening of the jaw or lips. Here we encounter some physiological differences between different genders and ages: the vocal cord length depends on these factors. Women have shorter vocal cords than men. There are also differences in size between the mouth-throat channel in men, women

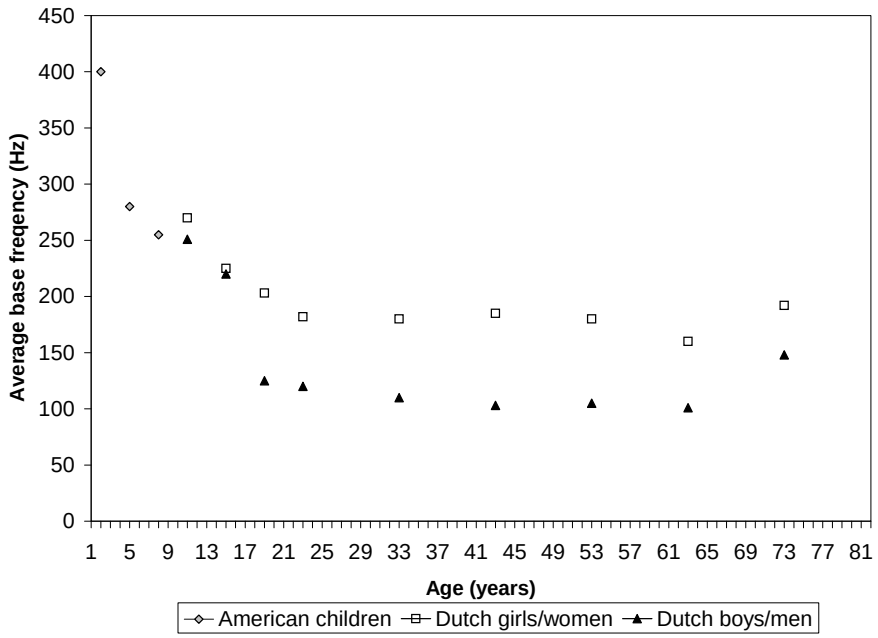

Figure 2.4: Relation between base frequency (in Hz) and age (in years) [A.C.M. Rietveld, V.J. van Heuven, 1997]

and children. It has been known from physics that the vocal cord length influences the base frequency. This results in differences in fundamental frequency for different genders and ages, as can be seen in figure 2.4.

Vibrating vocal cords produce a voiced phoneme while non-vibrating vocal cords produce a voiceless phoneme. But the physiology is not the only factor influencing frequency: every vowel also has its own intrinsic frequency.

The speech organs also change with age. The differences are most apparent in the fundamental frequency, especially below 18 years and above 62 years. Young children from both genders have a very high fundamental frequency, but only for boys there is big increase in vocal cord length and volume during puberty. Another point where age influences speech is the articulation speed. This difference is currently covered for the most part by the use of HMMs.

Besides gender and age, other speaker characteristics also influence the speech. There are regional differences [Verhoeven and Van Bael, 2002] and speech from people speaking or being raised with different dialects differs acoustically. An example of this is the devoicing of the phoneme /z/ around the city of Amsterdam in the Netherlands. There also are influences on word usage: dialects often use words that do not occur in the standard language. Sometimes there even are subtle differences at the syntactic level: in Dutch the order of the verbs in a relative clause is swapped in the Northern part of the Netherlands [Wiggers, To appear in 2008].

Another influence is the social group speakers belong to. Lower social classes use more dialect than the higher social classes do. Influences can, once again, most readily be seen in the pronunciation. Non-native speakers often keep influences from their native language, even for a long time after migrating [Ikeno et al., 2003]. On the grammatical level, higher classes do not use longer sentences; they do however vary their language more, known as 'lexical richness' [Wiggers, To appear in 2008]. There are also clear differences in word usage for different age categories and genders.

Other influences on speaker groups include speaking style, which can be formal or informal, and interaction type: people tend to speak differently to humans than they do to machines. Sometimes people over-articulate when talking to a speech recognizer.

Another peculiar acoustical phenomenon is the development of the so-called 'polder-Dutch', in which the pronunciation of diphtongs changes. An example is the change of pronunciation of /ei/ to 'ai'. This development seems to have been initiated by young, highly educated women; it is thus a pronunciation difference related to a combination of gender and social class [Wiggers, To appear in 2008].

In this research, the acoustical differences between different age categories for men and women will be investigated. This research is done in the context of the N-Best project, which will be introduced in detail in the next chapter.

2.3 Related Work

Only this year, Susanne Schötz summarized the related work in her paper on the Acoustic Analysis of Speaker Age [Schötz, 2007]. She mentions that "features related to speech rate, sound pressure level (SPL) and fundamental frequency (F_0) have been studied extensively, and appear to be important correlates of speaker age". She also mentions the substantial differences between the genders when

looking at the aging process: males experience big changes to their speech during puberty while female speech changes more around the menopause. Moreover, "age-related changes in adults are generally greater in men than in women". It was found that human listeners are able to differentiate between speech from speakers with different genders and even ages [Harry Hollien, 1987], [Schötz, 2007].

In [Daniel Elenius and Mats Blomberg, 2004], the recognition of children's speech was investigated using models for different age classes and a strong correlation between age and accuracy was found.

When it comes to the differences between Northern and Southern Dutch, the author of [Ayako Ikeno, Bryan Pellom, Dan Cer, Ashley Thornton, Jason Brenier, Dan Jurafsky, Wayne Ward, William Byrne, 2003] came to the important conclusion that even for English pronounced with a Spanish accent, acoustical retraining of the models to better fit the Spanish accented pronunciation results in much higher recognition rates. This indicates that a similar approach would benefit our Northern and Southern Dutch ASR tasks.

The Sonic speech recognition system was ported to Dutch before, as was noted in [Huijbregts, M.A.H., Ordelman, R.J.F., de Jong, F.M.G., 2005]. The BN task that was performed here, performed at a recognition rate of 70%.

The N-Best project itself has been described most recently in [Judith Kessens and David van Leeuwen, 2007]. Additional speech data for the Dutch language is being collected, according to [Catia Cucchiarini, Hugo Van hamme, Felix Smits, 2006].

Chapter 3

The N-Best Project

> Most conversations are simply monologues delivered in the presence of a witness
>
> *Margaret Millar*

In this chapter the N-Best project, of which this research is a part, will be discussed in detail. First the project will be presented and the remainder of the chapter explains the details of the project. These details provide the basis for the comparison of the two speech recognizers under consideration.

3.1 Introduction

The N-Best project was started with the goal to "evaluate the performance of present-day large vocabulary continuous speech recognition (LVCSR) systems for the Dutch language" [David van Leeuwen, Judith Kessens, 2006]. In order to achieve this, a common evaluation method is required. During this project a special evaluation framework will be developed that will help compare current LVCSR systems. It will also help track the progress made in this field by allowing future LVCSR systems to be compared using the same framework. The full name of the N-Best project is "Evaluation plan for the North-and South-Dutch Benchmark Evaluation of Speech recognition Technology (N-Best 2008)". The evaluation method chosen is inspired by past NIST evaluations in the US [Fiscus, 2005], [Fiscus, 2006] and the ESTER evaluation in France [ESTER].

This project is a cooperation of research groups in Flanders and the Netherlands. They have all committed themselves to participating in the evaluation performed by TNO Human Factors in the Netherlands in 2008. The evaluation will commence with an optional 'dry-run', during which TNO will send data in the evaluation format to the participants who can then rehearse the evaluation. Their results will then be sent back to TNO in order to allow TNO to rehearse their side of the evaluation. This dry-run will then be evaluated so that some changes can be made to the evaluation process, if necessary.

In 2008 the actual run will take place following the same procedure as the dry-run. TNO will evaluate the results it received from each participant and score them. An adjudication period will follow during which the participants

can comment on the evaluation results. The evaluation will be concluded with a workshop during which TNO and the participants will present their findings and exchange papers describing their systems in more detail.

3.2 Task and Materials

The task is to recognize all spoken words in a set of given excerpts of audio. This task is commonly known as *automatic speech recognition* or *speech-to-text* (STT). The audio material will be limited to specific domains and languages:

- Speech in the audio excerpts will be in the Dutch language.
- The global accent of the speakers will be Northern Dutch or Southern Dutch (also known as Dutch and Flemish).
- The audio excerpts will be obtained from radio and television news shows (Broadcast News, BN) and telephone conversations (conversational telephone speech, CTS).

There is no limit on the processing time used for the ASR but the results should be delivered to TNO within a month. Participants are obligated to provide a system description clarifying the technologies, training databases and training details used, allowing others to learn from their approach.

3.2.1 Conditions

Boundaries will be added to several aspects of this ASR task. For some aspects primary tasks will be stated that each participant must fulfill. For other aspects there will be a primary condition, which is compulsory, and contrastive conditions, which may optionally be performed in addition to the primary condition. Participants are allowed to add their own contrastive conditions of choice, to highlight their specific strengths and specialties.

Accent and Speech Domain

The two accents (Northern and Southern Dutch) combined with two speech domains result in four different primary tasks, see Table 3.1. This information may be used to condition parts of the ASR system on. The results will be analyzed separately for the four tasks so a participant can decide to focus on some of the tasks. However all tasks must be performed in order to make a valid submission.

Table 3.1: Tasks evaluated in N-Best 2008 with approximate durations of the evaluation data

	BN	CTS
Northern Dutch	2 hours	2 hours
Southern Dutch	2 hours	2 hours

Audio Material

The audio material will be recorded starting from 1 January 2007. The participants may not use training data collected in the same period in order to avoid that their ASR systems are trained with the same data they will be tested with. This would lead to better results than the ASR systems can achieve on normal, unseen data and would thus be misleading. The BN will be recorded from public or commercial television and radio broadcast in the Netherlands and in Flanders. For the CTS data will be used that is similar to the data in the well-known CGN (Corpus Gesproken Nederlands). The BN domain will primarily cover news and affairs shows. Each 2 hour BN material will consist of about 10 excerpts from a single show, where 'broad segments', the periods under evaluation, are indicated by an index file, totaling to about 12 minutes of speech per show. Each 2 hour CTS recording will originate from 12 dialogs of about 10 minutes, where each conversation side will speak roughly 5 minutes.

The CTS evaluation material will roughly be balanced for important speaker characteristics such as sex and geographical location. Similarly, the BN material will be chosen from a wide range of news shows and broadcast channels, some of which will not be included in the training material.

For the BN tasks, broad segments will be chosen that will contain mainly speech. Hence, segmentation of the BN audio file in speech, music, jingles and other audio types is not an item under evaluation. However, the broad segments will be much larger in size than the typical 'chunk size' found in CGN, and may include some non-lexical audible events, such as coughs, hesitations, slams, etc.

The CTS task will consist of 2-channel files, for with both sides of the conversation have to be processed. The broad segments will include the 'silent' parts of conversation sides.

Processing Time

It was decided that there should not be constrictions on run time during the first edition of this project. However participants are allowed to admit contrastive results that were obtained under run time restrictions. The run time restrictions are expressed in RTs, which stands for 'times real time' and is defined as follows:
RT = processing time/duration of the audio material
An overview of the primary and contrastive conditions on run-time can be seen in Table 3.2.

Table 3.2: Primary and contrastive processing time conditions defined in N-Best

Name	Real-time factor	Condition
unlimited	$< \infty$	primary
ten-times RT	< 10	contrastive
real-time	< 1	contrastive

Training

The participants will receive the training material from TNO, consisting of acoustical and language modeling training data. The acoustical training data will be a selection of CGN. The text resources will be a selection from the Twente News

Corpus (TwNC) for Northern Dutch and from Mediargus for Southern Dutch. In the primary condition only this training data can be used but as a contrastive condition other training data may also be used, as long as it originates from before 1 January 2007.

3.2.2 Evaluation Measure

Primary Evalution Measure

The primary evaluation measure will be the Word Error Rate (WER), as calculated by NIST sclite tools. ASR systems are often compared using this measure which corresponds to the percentage of incorrect words, as defined in equation 3.1 from [Jurafsky and Martin, 2000]. Alignment between the reference transcription and ASR word hypothesis will be carried out in a way that minimizes the WER. The WER values will be reported for each of the four primary acoustic/domain conditions separately.

$$WordErrorRate = 100 \times \frac{Insertions + Substitutions + Deletions}{TotalWordsinCorrectTranscript} \quad (3.1)$$

For the calculation of the word error rate only lexical entries in the reference and hypothesis files are accounted. Non-lexical events (coughs, hesitations, filled pauses) will not be included in the primary WER scores but may be analyzed separately.

In Dutch, proper names should be capitalized (e.g. "de Tweede Kamer" vs. "een blauwe kamer"), this is different from evaluation in English. The first word of a sentence should not be capitalized, unless it would have to be capitalized for being a proper name or similar reasons.

Words should be accented in cases where leaving the accents out could lead to (written) ambiguities in Dutch (e.g., een blauwe loge vs. niet één logé). Punctuation will not be considered.

Alternative Evaluation Measure

The WER will also be calculated in a time-mediated way, and reported separately. For sites that include word-based confidence measures, this confidence measure will be evaluated along similar lines as NIST Normalized Cross Entropy [Entropy].

3.2.3 Training Data

The training data in the evaluation will be standardized as much as possible, in order to make the results of the evaluation meaningful to future researchers, and concentrate on technological and training approaches for this evaluation.

The acoustic training material will be obtained entirely from CGN (Corpus Gesproken Nederlands, Spoken Dutch Corpus), version 2.0. The corpus is separated in Dutch and Flemish parts, and consists of several components. For the speech domains BN and CTS we have specified the components that can be used for training in Table 3.3. Note that the durations indicated are calculated from the raw audio files, and may include a significant amount of silence.

Table 3.3: Specification of the acoustic training components of CGN

Speech domain	Component	Duration (hours) Dutch	Flemish
Broadcast News	f: broadcast interviews	42.9	20.9
	i: live commentaries	30.7	12.9
	j: news/reports	8.3	9.1
	k: broadcast news	27.5	8.2
	l: broadcast commentaries	7.9	6.8
	Total	99.4	52.9
Conversational Telephone Speech	c: switchboard	55.3	36.5
	d: local minidisc	36.7	27.5
	Total	92.0	64.0

The training data for the language model is obtained from two resources: the Dutch publisher PCM (via the Twente News Corpus) and the Flemish Mediargus (via ESAT Leuven). From both sources a collection of newspapers over a fixed period will be chosen to form the training material for the language model.

Other text resources available within the N-Best evaluation can also be used in the primary condition, such as the transcripts of the acoustic training material and transcripts of other CGN components. In the contrastive training condition other text resources than specified here can be used, as long as the original creation date pre-dates 1 January 2007.

Vocabulary, Dictionary, and Questions

The pronunciation dictionary that may be necessary for building an LVCSR system for Dutch will not be distributed or specified, even though it can be considered a data resource that strongly influences the recognition performance. 'Derived' consequences of the dictionary, such as the phone set and the vocabulary, are considered design choices and are part of the difference between the various systems under evaluation. However, note that the CGN is delivered with a pronunciation dictionary that covers most of the words in the CGN. Similarly, if the LVCSR technology needs phonological questions for decision trees used in a state-clustering process, sites must provide these themselves.

3.2.4 Development Test Data

A part of the acoustic training data will be split off as development test data. It will be formatted in the same way as the evaluation material will be distributed. The splitting will be performed based on recording date, such that the development test data will originate from a more recent period than the training data.

3.3 Formats

The data formats will be similar to other ASR evaluations, mostly performed by NIST, as to minimize the effort for the ASR sites to run the evaluation.

3.3.1 Audio Format

The audio format of development test and evaluation data will be distributed in NIST sphere format. The technical encoding parameters are tabulated in table. Other audio resources, such as the acoustic training database, will be distributed in the format of the original supplier.

Each BN excerpt and each telephone conversation will be distributed as a single audio file, even though not all parts of the audio will be part of the evaluation.

3.3.2 Evaluation Control Files

Along with all audio files an index file will be distributed indicating the segments within the audio files that are part of the evaluation. The format of these files will be NIST Unpartitioned Evaluation Map (UEM) files [Fiscus, 2006]. These ASCII text files consist of 4 fields, the semantics are in table 3.4. Each line in the UEM indicates a segment in an audio file that must be processed by the speech recognition system.

Table 3.4: Sphere encoding parameters for evaluation and development test

Parameter	BN	CTS
Encoding	PCM	A-law
Resolution	16-bit	8-bit
Sample frequency	16 kHz	8 kHz
No. Channels	1	2

Several segments within the same audio file can be referenced, as well as one or two channels for CTS data. Note that UEM segments will be generally larger in duration than the typical 'chunk' size in the CGN training material. There will be one UEM file for each of the four primary tasks in this evaluation.

3.3.3 Result Format

The ASR-sites are required to produce the word hypothesis in time-marked conversation (CTM) format [Fiscus, 2006]. This text file consists of records each having 8 fields, for which the meaning is indicated in Table 3.5. The CTM format can also be used for other kinds of information, such as speaker identity or non-lexical acoustical events. This can appreciated from fields 7 and 8, where for field 7 the alternative token types frag (word fragment), fp (filled pause) are among the types that may occur in the reference transcription [Fiscus, 2005].

However, for the purpose of this evaluation, only tokens of the type lex (i.e., words) will be scored for calculating the WER. Similarly, field 6, the words confidence, is optional and the value NA may be given, in which case no confidence scoring is performed.

For every UEM file one CTM result file should be generated for the primary system and primary condition. For contrastive systems and contrastive conditions, a subset of UEM files can be processed.

Table 3.5: Fields in the CTM format

Field	Meaning	Example
1	source ID	cn01345
2	channel (1 or 2)	1
3	start time (sec)	17.354
4	duration (sec)	0.786
5	token (i.e., word)	minister
6	confidence (<0-1>)	0.9983
7	token type	lex
8	speaker	unknown

3.3.4 Training Files

Acoustical and language modeling training files will be distributed in the formats that the original supplier uses.

3.4 Evaluation Rules

There are a number of rules in order to make the outcome of the evaluation more meaningful and to make it possible to evaluate the merits of different approaches the different participants have taken. In this section we will tabulate the rules indicated elsewhere in this document, as well as introduce some new ones.

- Any form of human interaction with the evaluation data is not allowed. In particular, listening to the evaluation data, or inspection of the ASR results, either though manual browsing or statistical reports of the results, is not allowed until the primary system results have been submitted to TNO.
- All audio files should be processed independently. It is not allowed to adapt models to one audio file and used the adapted models for processing other files. Within one audio file, all available information may be used, and in an off-line way, so that for instance speaker clustering/adaptation within one BN news show is possible.
- For the primary condition, the acoustical material for training Dutch acoustic models is limited to the CGN database (any component). The acoustical models may be 'seeded' from acoustical models trained in a different language, or from a different Dutch database. Here we mean by seeding that acoustic models trained using other databases can be used for the initial segmentation of new acoustic models.
- For the primary condition, the text resources for training the language models is limited to the databases indicated in section 3.2.3, including the transcripts of CGN.
- Any form of training data obtained from sources after 1 January 2007 is not allowed, for any evaluation condition. This applies in particular to acoustical and language modeling or vocabulary choice. Even in contrastive submissions, data originating from sources produced after this

date is not allowed. If a site wishes to use acoustical or textual resources from the web in an purely automatic fashion, care should be taken that the date of origin stems from before 1 January 2007.

- All evaluation data must be processed and results file should be generated for all four primary tasks.
- Knowledge of the primary task (domain and accent) is allowed. Any form of other meta-data that can be useful to the ASR system must be obtained by automatic means. This means, for instance, that for the BN tasks it is allowed to try to recognize the speaker and used special models for this speaker. However, there is no guarantee that the BN material will contain the same speakers as in the primary training material.
- The CTM files should be sent to the evaluator before the end of the evaluation deadline. Later results will be marked as such. Results received after the reference transcriptions have been released will not be accounted as primary results.

The time table of this project is added as Appendix A.

Chapter 4

Tools and Resources

> The tongue is but three inches long, yet it can kill a man six feet high
>
> *Japanese Proverb*

This chapter will describe the tools and resources that were used during this project. The most important resource is the Spoken Dutch Corpus, containing the speech data needed to train Sonic. These large speech data files were divided into more convenient segments using a tool named 'shntool'. These speech fragments were then converted to the appropriate file format with the 'sox'-tool.

4.1 The Spoken Dutch Corpus

The resource that will be used for the audio data is the Corpus Gesproken Nederlands, or Spoken Dutch Corpus. The Corpus Gesproken Nederlands (CGN) was collected through a joint Flemish-Dutch project, carried out between 1998 and 2003, and can be ordered from this website [The Spoken Dutch Corpus Homepage]. The intended result of this project was 'a database of contemporary standard Dutch as spoken by adults in the Netherlands and Flanders' [CorpusGesprokenNederlandsDocumentation]. In the end, a total of 9 million words was collected. This was divided into two parts: over 5.6 million words were collected in the Netherlands and 3.3 million words in Flanders. The CGN consists of a collection of audio data along with several kinds of transcriptions and annotations. This includes an orthographic transcription, a phonetic transcription, part-of-speech tagging and word segmentation. Background noises are not represented in the transcript. The CGN also includes frequency lists for the data in the corpus. Furthermore, the data is divided into several categories, as can be seen in table 4.1.

4.1.1 Preprocessing of the Data

During transcription, the audio files were divided into small chunks of about 2 or 3 seconds. The chunks are separated by time markers which serve as anchor

Table 4.1: Categories in CGN

	Category	Flemish words	Dutch words
a.	Spontaneous conversations ('face-to-face')	878,383	1,747,789
b.	Interviews with teachers of Dutch	315,554	249,879
c.	Spontaneous telephone dialogs (recorded via a switchboard)	465,096	743,537
d.	Spontaneous telephone dialogs (recorded on MD with local interface)	343,167	510,204
e.	Simulated business negotiations	0	136,461
f.	Interviews/discussions/debates (broadcast)	250,708	539,561
g.	(political) Discussions/debates/meetings (non-broadcast)	138,819	221,509
h.	Lessons recorded in the classroom	105,436	299,973
i.	Live (e.g. sports) commentaries (broadcast)	78,022	130,377
j.	News reports/documentaries (broadcast)	95,206	90,866
k.	News (broadcast)	82,855	285,298
l.	Commentaries/columns/reviews (broadcast)	65,386	80,167
m.	Ceremonious speeches/sermons	12,510	5,565
n.	Lectures/seminars	79,067	61,834
o.	Read out speeches	351,419	551,624
	Total	**3,261,628**	**5,564,644**

points for the alignment of the speech file and the transcript. The CGN also contains a lexicon.

The Sonic speech recognition system requires the orthographic transcriptions accompanying the sound fragments for the acoustic training process. These transcriptions are added to the CGN in the following ShortTextGrid format:

> Below non-literal text is indicated by means of curly brackets: [...]. The numbering of the lines has been used for the purpose of reference and is not part of the format.
>
> The first three lines are always the same.
> 1. File type = "ooTextFile short"
> 2. "TextGrid"
> 3. empty line
>
> On lines 4 and 5 a description is given of the timespan involved. Time is expressed in terms of the number of seconds, using three decimals.
> 4. begin time
> 5. end time
>
> Lines 6 and 7 describe the number of tiers that occur in the file.
> 6. <exists>
> 7. number of tiers

Lines 8 up to and including 12 contain information about the first tier.
8. "IntervalTier"
9. "speaker name"
10. begin time
11. end time
12. number of intervals in this tier

Lines 13 up to and including 15 describe the very first interval.
13. begin time
14. end time
15. "orthographic transcript"

Then all further intervals of the first tier occur in chronological order as in lines 13 up to and including 15. Every next tier after that follows all intervals of the preceding tier. The structure is identical to that of the first tier from line 8 onwards.

4.2 The Wav-Splitter: Shntool

During this graduation project, it was necessary to split many audio files in the wav-format into smaller audio files. These audio files should be split at the anchor points from the CGN annotation data in order to comply with the ASR system's requirements. For this task, a tool named 'shntool' was used, version 3.0.2 [Jordan, 2007b].

The shntool is a 'a multi-purpose WAVE data processing and reporting utility' [Jordan, 2007a]. It allows the user to get information on and perform different operations on sound files in various formats. Shntool does not come with a graphical user interface but works from the command line, which makes it easy to call this tool from DOS batch-files, for example.

The shntool is an open source tool and has a modular build, which makes it easier to adapt specific parts of the source code. This tool is a combination of three different parts:

1. the core program,
2. the mode modules,
3. the format modules.

The core program is a mere wrapper around the different mode modules, which implement the actual program functions. All these different mode modules each add different functionality to the tool. There are modules to display properties of the wav-file, to compute a hash-fingerprint, to join multiple wav-files, to generate a cue-file, to trim the silence from the end of wav-files and many more. Calling these different modules is done by typing 'shntool' followed by the name of the desired module. For example, the split tool is invoked using the command `shntool split ...`. This module is the relevant part of the shntool for this project. The format modules make it possible to use audio file types other than wav. For each format supported by shntool, a different format module exists.

Once again, it is possible for users to add their own format modules and thus expand the range of supported audio format types. This functionality is not required for this graduation project.

Each module has several usage options. Since only the split module was used, the options for the other modules will not be discussed here. The split program splits audio files at specific points. There are various formats in which these split points can be offered. They can be input via a file or via the terminal. The split points can be specified in bytes, cue sheets or in notation involving minutes, seconds, milliseconds and/or frames. The splitting program also allows the user to specify how the new and smaller audio files should be named.

While using this tool to split large audio files, it became clear that there was a limited number of parts a single audio file can be split in. Since the shntool is open source software, it was possible to alter the code and recompile the program so that this limit no longer posed a problem.

4.3 The Wav-to-Raw Converter: Sox

During the project it was necessary to convert wav-files to raw-files, i.e. 16-bit linear PCM format. This was done using a tool named 'sox': 'Sound eXchange: universal sound sample translator'. This is also a command line tool and it can, among other things, convert audio files from one format to another, including wav and raw. It can even add effects to the sound files, but that was not necessary in the context of this project. To quote the 'README' file:

> SoX is intended as the Swiss Army knife of sound processing tools. It doesn't do anything very well, but sooner or later it comes in very handy.

It was mentioned in the Sonic manual [Pellom, 2001] that:

> If you are working with Microsoft wav-files, you can strip the header from the file using the "sox" program for Windows or Unix:
> `sox infile.wav -w -s infile.raw`

So that was what was done.

4.4 The CMU-Cambridge Statistical Language Modeling Toolkit

During the creation of the Dutch ASR system, a language model needed to be created. The widely used CMU-CSL language modeling toolkit was used, see also [CMU]. This toolkit takes a large training text as its input and then allows the users to train a language model based on this training text. It is possible to use context cues to indicate the beginning and ending of sentences and other things that the user finds relevant. There are also multiple ways to let the tool back-off from certain things; such as sentence boundaries. The language model can then, for example, disregard the words from the end of the previous sentence when predicting the probability of the first word of a new sentence. There are several options for handling out-of-vocabulary words and discounting

strategies, used for smoothing. The toolkit can output models in the ARPA-format in readable ASCII-format, which is the way Sonic needs its language models. It is also possible to calculate the new language model's perplexity, which is an information measure indicating how well a language model predicts a certain test text.

4.5 The Sonic Speech Recognition System

In my literature study [Clerx, 2007], I compared two popular state of the art ASR systems named Sonic [SonicWeb] and Julius [Julius Homepage]. They are in fact basic systems providing core speech recognition functionality. These systems are not complete speech recognizers but they provide a large collection of functions that are currently used during research projects. These systems need to be adapted and trained for new ASR tasks using specific speech and language data. They offer a large collection of specific features and functions that researchers may or may not use for their task. Using such an ASR system is similar to solving a puzzle, with the ASR system's collection of functions and the different kinds of training data as the pieces of the puzzle. And it was decided that Sonic provides the most convenient pieces for this specific puzzle.

Sonic is a toolkit for enabling research and development of new algorithms for continuous speech recognition. Although it is not a general HMM modeling toolkit, it enables researchers to conduct speech recognition experiments or to prototype live voice-enabled applications. The system originates from the Center for Spoken Language Research at the University of Colorado, Boulder. The recognizer can run in batch-mode to process many audio files sequentially, or in live-mode to speak into the microphone and see output in real-time. Provided with the software are tools for retraining the speech recognizer on new data, tools for integrating new language models, both statistical N-gram and grammars, and port the recognizer to a new language. Speaker and environment adaptation routines are also provided in addition to example applications to run experiments in either batch or live-mode.

Sonic has already been ported to French, German, Italian, and Spanish (Mexican and Chilean) during a workshop [Pellom and Cole, 2003]. It has also been ported to Turkish [Ö. Salör, B. Pellom, T. Ciloglu, K. Hacioglu, M. Demirekler, 2002] and Croatian, Arabic, Russian, Portuguese, Korean, and Japanese [SonicWeb]. This indicates that porting it to Dutch should be feasible.

Sonic is specifically designed for speech recognition research with careful attention applied for speed and efficiency needed for real-time use in live applications. The system can work in real-time for modern PCs (Intel Pentium 4, 2.2 GHz) for vocabularies up to approximately 40k words. Performance gains can be further obtained using speaker adaptation, which can provide improved hypothesis pruning. The long-term goal is to provide real-time speech recognition up to 64k words to the research community.

4.5.1 Performance

Sonic has already been benchmarked on tasks that are similar to the tasks that have to be performed during the N-Best project. The Wall Street Journal task is comparable with the BN task in the sense that both contain news content. The Wall Street Journal task however consists of read speech while the BN task also contains sport commentaries, which is spontaneous speech. So it can be expected that the results for the BN task will be less good than those for the Wall Street Journal task. The Switchboard task is very similar to the CTS task in the N-Best project: spontaneous large vocabulary speech. The results are listed in table 4.2.

Table 4.2: Sonic benchmark [Sonic Homepage]

Speech Recognition Task Description	Vocabulary Size	Word Error Rate without adaptation	Word Error Rate with adaptation
Wall Street Journal	5k	3.9%	3.0%
Wall Street Journal	20k	10.0%	8.6%
Switchboard	40k	41.9%	31.0%

In comparison, with SPHINX-II, a word error rate of 6.7% is attained on the speaker-independent 5000-word Wall Street Journal continuous-speech recognition task [Mei Hwang, 2001]. And with the Microsoft Whisper recognizer, a WER of 4.87% was achieved, unfortunately without mentioning the vocabulary size. [Hagai Attias, Li Deng, Alex Acero, John C. Platt, 2001]. Sonic's performance on this task is thus comparable to these other two well-known systems. In [A. Morgan, N. Chen, B.Y. Zhu, Q. Stolcke, 2004] from 2004 it is mentioned that

> "...current systems giving incorrect output for 20-40% of the words, depending on the system complexity and test set."

This indicates that Sonic's performance on these tasks is average.

4.5.2 Features

We will now list some features indicating Sonic's capabilities. For a detailed explanation of these features, we refer to [Clerx, 2007].

Phonetic Aligner

- Provides word, phone, and HMM state-level boundaries for acoustic training.
- Decision-tree based trainable letter-to-sound prediction module.
- Multilingual lexicon support.

Phonetic Decision Tree Acoustic Trainer

- Estimates parameters of state-clustered continuous density Hidden Markov Models.

- Incorporates phonetic position and context questions.
- Distributed/parallel acoustic trainer (multiple machines, operating systems, CPUs).

Core Recognizer

- Token-passing based recognition using a lexical-prefix tree search.
- Cross-word triphone models & up to 4-gram language model in first-pass.
- HMM state durations modeled using Gamma distributions.
- N-best list output; Lattice dumping, and second-pass rescoring functionality.
- Word confidence computed from word-posteriors of word-graph.
- Class-based, word-based, and concept-based N-gram language models.
- Dynamically switched statistical language models (dialog state-conditioned language models).
- Keyword & regular expression based grammar spotting with confidence.
- Phonetic fast-match constrained ASR search for improved decoding speed.
- PMVDR and MFCC feature representation.

Speaker Adaptation

- (Confidence Weighted) Maximum Likelihood Linear Regression (MLLR).
- Constrained Maximum Likelihood Linear Regression (CMLLR).
- Lattice-based MLLR (Lattice-MLLR).
- Maximum a Posterior Linear Regression (MAPLR).
- Vocal Tract Length Normalization (VTLN).
- Cepstral mean and variance normalization.

Language Portability

- Aligner, trainer, recognizer designed to incorporate new phone sets and foreign vocabularies.

4.5.3 Core Technology

In this section, we will explain some of these terms that are considered part of the main technology. Explaining all possible options would lead us to far but they are all explained in [Clerx, 2007].

Feature Extraction

When performing speech recognition, the first step is to analyze the given sound wave and extract relevant data from it. This signal processing starts by dividing the the sound wave in time slices from which spectral features are extracted. Spectral features give information about how much energy is present in the signal at different frequencies. Later on, these features are used to determine which sounds were uttered. Sonic offers the choice between the most widely used feature representation named Mel-frequency cepstral coefficients (MFCC) and the newer Minimum Variance Distortionless Response approach named PMVDR. MFCC has proven to be one of the most effective sets of features for speech recognition. It consists of a transform on the signal spectrum followed by a specific filtering procedure. This approach however is not perfect: it has a limited ability to remove undesired harmonic structures, especially for high pitched speech. Furthermore for high pitched voiced speech the bandwidths are mis-estimated. Moreover MFCC is expected to carry a good deal of speaker dependent information. The evidence of this is that the same feature representation is commonly used in speaker recognition systems. It is also widely accepted that MFCC is quite fragile in noise and additional compensation such as feature enhancement. As a result, model adaptation is needed for acceptable performance in realistic environments. The MVDR methodology, on the other hand, can effectively model medium and high-pitch speech and excellently smooth undesired excitation information. This yields a performance gain in noisy conditions:

> "The WER is shown to decrease by 27.3% with respect to the MFCCs and 18.8% with respect to recently proposed PMCCs on an extended digit recognition task in real car environments."

MVDR also better suppresses speaker dependent information yielding more accurate recognition and faster decoding in both clean and noisy conditions. This approach is however not yet widely adopted among ASR researchers. It was shown in Padmanabhan and Dharanipragada [2005] that another implementation of MVDR outperformed MFCC and another recent feature extraction technique for noisy conditions named PLP in terms of the WER. However in the same paper a new technique was proposed that can be combined with MFCC, MVDR and PLP. It is called penalized Mutual Information Projection and was shown to result in a relative improved WER of 0.9% for MFCC, 16.5% for PLP and 10.5% for PMCC.

Acoustic Models

Many speech recognizers use acoustic models based on HMMs. The acoustic models used in Sonic are a special form of these HMMs called decision-tree state-clustered continuous density HMMs with associated gamma probability density functions to model state durations. Each state is modeled by a weighted set of M Gaussian distributions, which represent physical aspects of the sound that is being analyzed. Acoustic modeling in Sonic makes the assumption that the feature elements are independent and can therefore be modeled using a diagonal covariance matrix rather than a full-covariance matrix. The emission probability, $b_j(o_t)$, therefore becomes,

$$b_j(o_t) = \sum_{m=1}^{M} \frac{w_m}{(2\pi)^{D/2}\sqrt{\prod_{d=1}^{D}\sigma_m^2[d]}} \exp\left\{-\frac{1}{2}\sum_{d=1}^{D}\frac{(o_t[d]-\mu_m[d])^2}{\sigma_m^2[d]}\right\}$$

where $o_t[d]$ represents the d^{th} feature dimension (D=39) for the t^{th} frame of data. Each Gaussian is weighted by a factor, w_m, such that the weights sum to unity. The state durations are modeled using another type of distributions: gamma distributions. The discrete parametric gamma distribution is defined as

$$p(\tau) = K \exp\{-\alpha\tau\}\,\tau^{p-1}, \tau = 0, 1, 2, \ldots$$

where a>0 and p>0. K is a normalizing term. The discrete parametric Gaussian distribution is denoted as

$$p(\tau) = K \exp -\frac{(\tau-\mu)^2}{2\sigma^2}$$

In Burshtein [1996] the gamma distribution was found to produce a high-quality fit to the empirical state and word duration distributions. On the other hand, the discrete parametric Gaussian distribution produces high-quality approximations for word durations, but is inferior in its ability to describe certain state durations. In addition, unlike the Gaussian distribution, the gamma distribution is one-sided: it assigns zero probability to negative τ's, which is appropriate for duration distributions. Finally, the slower decay of the gamma distribution compared to the Gaussian distribution is more appropriate for duration modeling. Careful examination showed that the gamma fit is almost always closer to the empirical distribution than the other parametric approximations examined, although the difference from the Gaussian distribution is small for word durations. Gamma distributions are thus an appropriate choice for modeling state durations.

In order to model speech accurately, HMM states are clustered in various ways depending on their triphone context i.e. the immediate phoneme to the left and immediate phoneme to the right of the current phonetic unit. The reason behind this is that subsequent phonemes are not spoken in isolation but overlap in continuous speech. Phonemes are thus realized differently depending upon the neighboring sounds. This effect is known as co-articulation. For a language such as English there are 45^3 possible triphone units. Many of these contexts never appear in the training data and therefore most modern recognition systems use some sort of unsupervised clustering method to obtain a primitive set of about 5000-6000 clustered states. Each clustered state models a unique phonetic quality in the speech data. Each HMM state can be modeled with a variable number of multivariate mixture Gaussian distributions. This means that a mixture of different Gaussian distributions is used to model each HMM state. These different Gaussian distributions are weighted in order to better approach the actual distribution of this state. The triphone models are also shared between all words to avoid excessive memory requirements. Special cross-word triphone models are added to specifically model the co-articulation effects between neighboring words [G. Boulianne, J. Brousseau, P. Ouellet, P. Dumouchel, 2000]. The decision trees are used to determine contextually equivalent sets of HMM states. A phonetic decision tree is a binary tree in which a question is attached to each node. One tree is constructed for each state of each phone to cluster all of the corresponding states of all of the associated triphones.

The states in each subset are tied to form a single state. They are built using splitting questions which are automatically derived to maximize the likelihood of the training data. In the case of a phonetic decision tree, these questions are related to the phonetic context as illustrated in figure 4.1 from [Young et al., 1994].

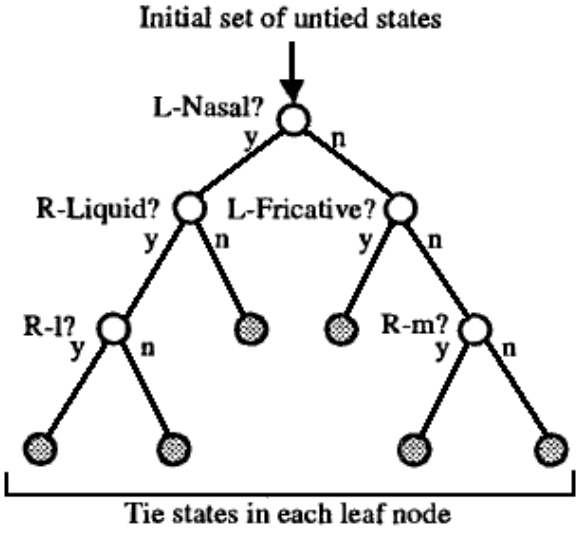

Figure 4.1: Phonetic decision tree example

When building a speech recognizer based on HMMs, a procedure is needed to train the HMMs. This is often a time consuming process that requires complex algorithms known as forward-backward algorithms. Sonic's acoustic model trainer instead uses the Viterbi algorithm for model estimation. This is an approximation of forward-backward algorithms which substantially reduces the amount of CPU effort needed to train acoustic models compared with forward-backward training methods [Bimbot, 1995]. The forward-backward training methods are well-known and can be reviewed in most text books, including [Jurafsky and Martin, 2000]. Viterbi training makes the simplifying assumption that each observation has resulted from the single most likely state sequence that might have caused it [Jia Li, Amir Najmi and Robert M. Gray, 2000]. Another way to view the Viterbi training is that the state sequence with the maximum a posteriori probability is assumed to be the real state sequence. While more efficient computationally, Viterbi training does not in general result in maximum likelihood estimates. However the training process is described in [Pellom, 2001] as follows:

> "The training process consists of first performing state-based alignment of the training audio followed by an expectation-maximization step in which decision tree state-clustered HMMs are estimated. Acoustic model parameters, i.e. the means, covariances, and mixture weights, are estimated in the maximum likelihood sense. The training process can be iterated between alignment of data and model es-

timation to gradually achieve adequate parameter estimation. The final model parameters are quantized for acoustic model storage."

However it was stated in [Jurafsky and Martin, 2000] that the forward-backward algorithm is actually a special case of the expectation-maximization algorithm. Unfortunately it remains unclear how Sonic can make good on the claim that there is an expectation-maximization step in the training process, using only the Viterbi algorithm and no type of forward-backward algorithm. As far as we can see, the training approach using Viterbi is much faster than traditional forward-backward algorithms but also much less accurate. Nevertheless the benchmark results in table 4.2 indicate a high recognition rate so the training process does not seem to degrade the performance.

Language Models

Sonic can work with class-based, word-based and concept-based models. We will only need the "standard" word-based models: each word in the vocabulary his its own N-grams associated with it, regardless of its class or concept. The other models are explained in [Clerx, 2007]. uses 'backoff' N-grams. Backoff is a tactic that can be used to fill in certain blanks in the statistical N-gram data. When there are no examples available of a particular trigram $w_{n-2}w_{n-1}w_n$ to compute $P(w_n|w_{n-1}w_{n-2})$, we can estimate its probability by using the bigram probability $P(w_n|w_{n-1})$. Similarly the unigram $P(w_n)$ can be used to compute $P(w_n|w_{n-1})$.

Two Pass Recognition

Sonic is executed in two steps: the first pass and the second pass. Unigram, bigram, trigram, and fourgram models can be applied during the first pass of recognition. Using a unigram model is the fastest choice but leads to inferior estimations in the first pass compared with the other options. Fourgram models make the best estimations but are slow. During the second pass of recognition rough results can be rescored using a longer span language model. This rescoring selects the best result(s) out of all the possibilities that were found during the first pass.

In more detail, the first pass recognizer performs a quick and simple analysis to determine the N possible phrases with the highest probability. This N-best list can also be output by Sonic. Word confidences can now be calculated for each of these possible phrases. These confidence scores indicate how much confidence the algorithm has in this result. If the confidence in the most probable phrase after the first pass is low, these possibilities are then passed on to the second pass recognizer. This recognizer is more complex than the first pass recognizer and it uses better, i.e. longer span, language models to rescore the possibilities to provide a more accurate decision. During the second pass it is also possible to compute word-posterior probabilities to provide word-level confidence scores. Using this two pass strategy reduces the average complexity compared to a single pass recognizer incorporating both the first and second pass recognizer: the search space for the complex recognizer has been reduced by adding a low complexity first pass recognizer [Naveen Srinivasamurthy, Antonio Ortega and Shrikanth Narayanan, 2003].

The possibilities that are found during the first pass can be passed on in different formats. The first option is to use the N-best hypothesis: a simple list summing up the N possible phrases with the highest probability. This N-best list can contain sentences that only differ in one word. This means that a lot of redundancy is present in this notation. To avoid this redundancy, the second possible notation can be used: a word lattice. In this notation the separate possible sentences are combined into one lattice[1] structure, sharing words that appear in multiple sentences. This is a more efficient representation but slightly more difficult to read for humans. If arcs between the words are added to this lattice structure, we can turn this format into a word graph. Temporal constraints are implicitly embedded in this graph [Huang, 2001]. The difference between a word graph and a word lattice is illustrated in figure 4.2 from [Helzerman and Harper, 1994]. The posterior probabilities in the word graph are also used to compute the word confidence [Woodland, 2000]. Another possible format in Sonic is the progressive search lattice. This technique is useful for developing and implementing speech recognition systems with high computational requirements. The scheme iteratively uses more and more complex recognition schemes, where each iteration constrains the search space of the next [Murveit et al., 1993].

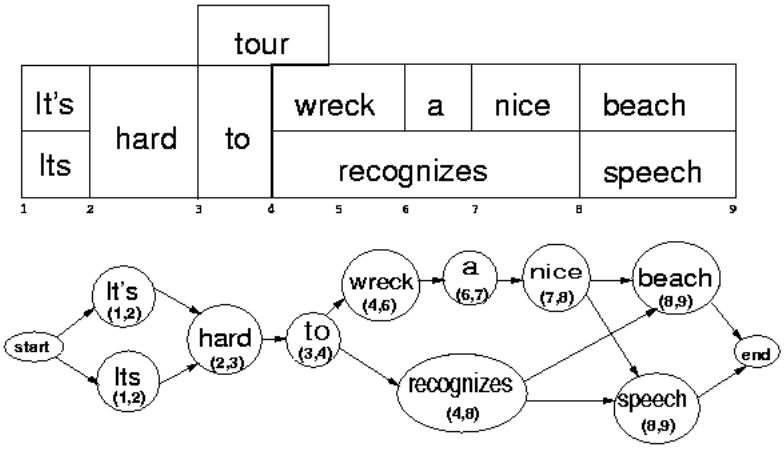

Figure 4.2: Word lattice and word graph with time stamps

Token-Pass Search Algorithm

Search within Sonic is based on the token passing model for speech recognition. Statistical N-gram decoding is accomplished using a tree-structured lexicon. Many two pass decoding algorithms require the use of a fast match for quickly finding when words in the lexicon are likely candidates for matching some portion of the acoustic input. Many fast algorithms are based on the use of a tree-structured lexicon, which stores the pronunciation of all the words in such a way that the computation of the forward probability can be shared for words which start with the same sequence of phones.

[1] A regular, periodic configuration of points, particles, or objects throughout an area or a space

In Sonic's implementation, tokens are propagated through a static lexical tree. As a result Sonic does not perform tree copying. The tokens themselves contain links to word history information so that long-span N-gram models can be used. To improve search efficiency, Sonic merges hypotheses that share the same two-word histories. When a token enters the root node of a lexical tree, Sonic predicts the best-case value for the trigram probability, by considering the previous two words and all possible follower words that are possible given the tree's root node. Upon exiting a leaf in the tree, Sonic inserts the word hypothesis into a lattice and recovers the true trigram probability by subtracting the logprobability estimate, from entrance at the root node, with the corrected trigram probability, considering the exact 3 or 4 word sequence of the propagated token. The search utilizes cross-word acoustic models in the first pass. Efficiency is further improved by applying beam pruning. Low-probability paths are then pruned at each step and are no longer extended. For each time step, the algorithm maintains a short list of high-probability words whose path probabilities are within some percentage, called beam width, of the most probable word path. Only transitions from these words are extended when moving to the next time step. Since the words are ranked by the probability of the path so far, which words are within the beam will change from time step to time step. This approach allows for a significant speed-up at the cost of less accurate results. Specialized search beams are associated with the entrance into the root nodes of the tree, nodes located within the tree itself, as well as for states near the leaf of the tree.

Histogram pruning is also applied to limit the maximum number of tokens propagated at word-ends, associated with the leaf nodes, or globally during search. The idea behind this is to apply vector quantization to the feature vectors. Now the feature vectors are divided into similar feature vectors, all matched to the same codebook vector. By counting the feature vectors matched to each codebook vector, we obtain a histogram. Now we know that similar feature vectors can be found only within the same bin and maybe its neighboring bins. The other bins can be pruned to increase the speed of the search, for example [Kunio Kashino, Takayuki Kurozumi and Hiroshi Murase, 2003].

The result of the first-pass of N-gram decoding is a word lattice. The word lattice can be converted into a word graph, illustrated in figure 4.2, and rescored using alternative, more sophisticated and time-consuming language models.

Phonetic Aligner

At the heart of the acoustic model training process is the Viterbi-based phonetic aligner called 'align'. This program takes raw audio files with associated text transcriptions as input and produces a time-aligned representation of the data: an association between frames of input audio to words, phonemes, and HMM states.

Accurate alignment of the audio data is key to training high-quality acoustic models. The phonetic aligner is user configurable and extensible while providing support for multiple languages. Phonetic alignment requires two knowledge sources: an acoustic model and a pronunciation lexicon. The aligner is designed to be user definable and configurable.

4.5.4 Usage

In order to use the batch recognizer the following items are required:

- One or more audio files to process
- A phoneme configuration file which defines the phoneme units in the desired language
- A language & acoustic model
- A lexicon containing word pronunciations for words contained in the language model

A single configuration file is used to setup the recognition job. The Sonic configuration file provides the listings of the decoder settings, phoneme set, language model, acoustic model, and pronunciation lexicon. They are used jointly as illustrated in figure 4.3.

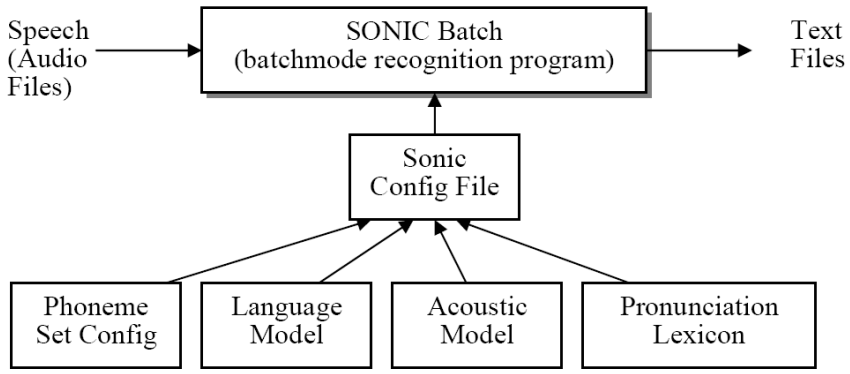

Figure 4.3: Inputs required to perform batch-mode recognition in Sonic

4.5.5 Why Sonic?

In the preceding literature study [Clerx, 2007], a detailed comparison was made between Sonic and Julius. Eventually Sonic was chosen for this project for a number of reasons, explained in this section.

Sonic offers the possibility to use MFCC or MVDR feature parameters, without the need for an external feature extraction tool. Furthermore it has better state duration modeling and an efficient acoustic model with decision tree clustering. It also offers faster training using an internal training algorithm and the possibility of lattice dumping. Sonic comes with a porting manual to facilitate the porting process and is explicitly built to be ported. Finally Sonic offers extra speed-up as a result of more pruning methods and increased precision through a large variety of speaker adaptation techniques, which are absent in Julius.

Considering the limited time available in the N-Best project, the faster training algorithm of Sonic is a big plus. On top of that the many speaker adaptation techniques that are implemented in Sonic and absent in Julius, should result in a higher recognition rate. Therefore, we recommend to use Sonic for the N-Best project.

Chapter 5

Porting Sonic to the Dutch Language

> It's not what you tell them . . . , it's what they hear
>
> *Red Auerbach*

The first step toward using Sonic as a speech recognizer for the N-Best project, is to port this speech recognizer to the Dutch language. This chapter will describe in detail how Sonic was ported from American English to Dutch.

5.1 Introduction

During the preceding literature study [Clerx, 2007], it was concluded that Sonic was a better choice for this project since it offers multiple speaker adaptation techniques and a faster training algorithm. As a result, we will port it to the Dutch language in this chapter.

In order to use Sonic as a speech recognizer for the Dutch language, some adaptations have to be made. Originally Sonic was built to recognize American English [Pellom, 2001] but it can be retrained to recognize other languages. This has already been done successfully with French, German, Italian, and Spanish (Mexican and Chilean) during a workshop [Pellom and Cole, 2003]. Sonic has also been ported to Turkish [Ö. Salör, B. Pellom, T. Ciloglu, K. Hacioglu, M. Demirekler, 2002], Croatian, Arabic, Russian, Portuguese, Korean, and Japanese [SonicWeb]. In order to help users with the porting process, a manual is available on porting Sonic to another language [Pellom, 2004]. The porting process involves retraining the Hidden Markov Models(HMMs). This retraining requires the use of different kinds of training data. This will be described in more detail in the next section.

5.2 Training Data

Porting Sonic requires several types of training data [Pellom, 2004]. Firstly, we need a mapping of phonemes in the target language to the most similar

phonemes in the source language, American English. Secondly, a knowledge base of the target language is needed, containing audio data along with word-level transcriptions and a pronunciation lexicon.

5.2.1 Dutch Phonemes

In order to port Sonic to Dutch, we need a phone set and phone mapping from Dutch phones to US English phones. Since we will use the audio data and phonetic transcriptions from the Corpus Gesproken Nederlands[1] (CGN), it is necessary to map the phoneme set from CGN to the English phonemes used in Sonic. The Dutch vowels from the CGN are displayed in table 5.1 and the Dutch consonants can be found in table 5.3. The Dutch language also uses some foreign vowels in loan words. They are much less frequent and the ones that are used in CGN can be seen in table 5.2.

Table 5.1: Dutch vowels with example words

IPA **symbol**	CGN **symbol**	IPA **example word**	**word orthography**	**English translation**
ɪ	I	bɪt	bit	bit
ɛ	E	bɛt	bed	bed
ɑ	A	bɑt	bad	bath
ɔ	O	bɔt	bot	bone
ʏ	Y	hʏt	hut	cabin
i	i	bit	biet	beetroot
y	y	fyt	fuut	grebe
eː	e	beːt	beet	bite
øː	2	nøːs	neus	nose
aː	a	zaːt	zaad	seed
oː	o	boːt	boot	boat
u	u	hut	hoed	hat
ɛi	E+	ɛi, vɛin	ei, wijn	egg, wine
œy	Y+	œy	ui	onion
ʌu	A+	zʌut, fʌun	zout, faun	salt, faun
ə	@	də	de	the

The English phonemes that are used by Sonic, are added in table 5.4. The manual on porting Sonic to other languages specifically states that this phoneme mapping does not have to be perfect. The reason for this is that the HMMs will be retrained, which will correct mismatches in the initial phoneme mapping. The porting manual describes this step as follows: "Just pick the closest one". The closest one was thus selected for each Dutch phoneme by the author, by pronouncing each Dutch and English phoneme and by evaluating the resemblance between them based on pronunciation and audio differences. The result of this process can be found in table 5.5.

[1] Spoken Dutch Corpus

Table 5.2: Foreign vowels with example words

IPA **symbol**	CGN **symbol**	**word orthography**	**English translation**
ɛː	E:	scène	stage
œ	Y:	freule	milady
ɔː	O:	zone	zone
iː	i	analyse	analysis
yː	y	centrifuge	centrifuge
uː	u	rouge	rouge
ɛ˜	E˜	vaccin	vaccine
ɔ˜	O˜	congé	leave
ɑ˜	A˜	croissant	croissant
ʏ˜	Y˜	parfum	perfume

Table 5.3: Dutch consonants with example words

IPA **symbol**	CGN **symbol**	IPA **example word**	**word orthography**	**English translation**
p	p	pɛn	pen	pen
b	b	bit	biet	beetroot
t	t	tɑk	tak	branch
d	d	dɑk	dak	roof
k	k	kɑt	kat	cat
ɡ	g	goːl	goal	goal (in sports)
f	f	fits	fiets	bicycle
v	v	oːvən	oven	oven
s	s	sɔk	sok	sock
z	z	zep	zeep	soap
ʃ	S	ʃɛf	chef	boss, chief
ʒ	Z	ʒyʁi	jury	jury
x	x	ɑxt	acht	eight
ɣ	G	ɣa ːn	gaan	to go
ɦ	h	ɦut	hoed	hat
m	m	mɛns	mens	human being
n	n	nɛk	nek	neck
ŋ	N	ɛŋ	eng	scary
ɲ	J	oraɲə	oranje	orange
l	l	lɑnt	land	land/country
ʁ	r	ʁɑt	rat	rat
j	j	jɑs	jas	coat
ʋ	w	ʋɑŋ	wang	cheek

Table 5.4: Sonic phoneme set

Phone	**Example**	**Phone**	**Example**	**Phone**	**Example**	**Phone**	**Example**
AA	father	DX	butter	KD	talk	GD	mug
AE	mad	DH	them	JH	Jerry	SH	show
AH	but	EH	bed	K	kitten	T	tot
AO	for	ER	bird	L	listen	TH	thread
AW	frown	EY	state	M	manager	UH	hood
AX	alone	F	friend	N	nancy	UW	moon
AXR	butter	G	grown	NG	fishing	V	very
AY	hire	HH	had	OW	cone	W	weather
B	bob	IH	bitter	OY	boy	Y	yellow
CH	church	IX	roses	P	pop	Z	bees
D	don't	IY	beat	R	red	ZH	measure
PD	top	BD	tab	S	sonic	SIL	silence
TD	lot	DD	had	TS	bits	br	*breathe*
ls	*lipsmack*	lg	*laughter*	ga	*garbage*		

Table 5.5: Dutch-American English phone mapping

Dutch	**English**	**Dutch**	**English**	**Dutch**	**English**	**Dutch**	**English**
I	IH	Y+	ER	d	D	n	N
E	EH	A+	OW	k	K	N	NG
A	AA	@	AX	g	G	J	NG
O	OY	E:	AE	f	F	l	L
Y	AH	Y:	ER	v	V	r	R
i	IY	O:	AO	s	S	j	JH
y	UH	E~	AY	z	Z	w	W
e	EH	O~	OW	S	SH	SIL	SIL
2	ER	A~	OW	Z	ZH	br	br
a	AY	Y~	AH	x	HH	ga	ga
o	OW	p	P	G	G	ls	ls
u	UW	b	B	h	HH	lg	lg
E+	EY	t	T	m	M		

5.2.2 Dutch Knowledge Base

The Dutch knowledge base is a subset of the CGN. This corpus contains all the data needed to retrain the HMMs in Sonic:

1. Audio Data
2. Word-level Transcriptions
3. Pronunciation Lexicon

We will explain these resources in more detail in separate sections.

Audio Data

We need to train Sonic to recognize telephone speech and broadcast news, see also chapter 3. Training HMMs is an intensive process which often suffers from a lack of training data. The more audio data is used, the better the acoustic models can be trained. So it is best to use all available data from the CGN for the HMM training process. In the N-Best project description [David van Leeuwen, Judith Kessens, 2006], an indication was given of the categories that are exemplary for the two tasks. In order to obtain enough training data for the HMMs we will use the other categories as well. The CGN categories which were not mentioned in the N-Best project description are assigned to the task or tasks they resemble, as indicated in table 4.1. We use all the categories because a well-known aphorism in speech recognition states that "there is no data like more data". Out of all the data, 70% is used as training data and 3 times 10% of the data is used for testing. This allows for multiple tests.

Table 5.6: Categories in CGN

	Category	**Broadcast news**	**Telephone speech**
a.	Spontaneous conversations ('face-to-face')	X	X
b.	Interviews with teachers of Dutch	X	X
c.	Spontaneous telephone dialogs (recorded via a switchboard)		X
d.	Spontaneous telephone dialogs (recorded on MD with local interface)		X
e.	Simulated business negotiations		X
f.	Interviews/discussions/debates (broadcast)	X	
g.	(political) Discussions/debates/meetings (non-broadcast)	X	
h.	Lessons recorded in the classroom	X	
i.	Live (e.g. sports) commentaries (broadcast)	X	
j.	News reports/documentaries (broadcast)	X	
k.	News (broadcast)	X	
l.	Commentaries/columns/reviews (broadcast)	X	
m.	Ceremonious speeches/sermons	X	
n.	Lectures/seminars	X	
o.	Read out speeches	X	

We decided to train the HMMs for Dutch and Flemish separately since there are significant differences in pronunciation between these two standards. The CGN pronunciation lexicon even contains different pronunciations for words in Dutch and Flemish. There are also two different pronunciations for 'formal' and 'informal' Flemish. Furthermore CGN category *e* does not contain any Flemish data: it is only available for Dutch.

We start out with the standard version of Sonic which is built to recognize American English. By retraining the HMMs using the Dutch audio data for the BN task, Sonic will be able to recognize the Dutch data from this category far better. This will constitute the basic BN speech recognizer for Dutch. Afterwards, these new, retrained HMMs will be retrained once again for the CTS task, using the CTS data categories from table 4.1. This will result in the basic CTS recognizer for Dutch. The same process should then be repeated for Flemish. In order to speed up the training process for Flemish, it is possible to start out with the Dutch recognizer and not with the original US English recognizer.

Sonic expects the training data to be put in a specific directory structure:

- speaker1/
  - file1.raw (headerless PCM audio)
  - file1.txt (text transcription)
  - ...
  - fileN.raw
  - fileN.txt
  - gender (MALE — FEMALE)
- speaker2/
- speakerN/

In order to achieve this with the CGN data, I created several Java programs:

1. collectSpeakerSexes
2. createSpeakerDirs
3. prepareForShntool

The collectSpeakerSexes Java program extracts the names and genders of each speaker from the CGN text file describing the project's speakers. The result is a text file consisting of rows with a single speaker name and the corresponding gender. The createSpeakerDirs program takes the text file that was built by collectSpeakerSexes as its input and creates a batch-file. This batch-file creates a directory for each speaker with the speaker's name as the directory name, and also adds a text file named 'gender', containing the word 'MALE' or 'FEMALE'. The third and last Java program, prepareForShntool, is the largest and most important one of the three. It has multiple tasks:

1. Convert the speech file transcriptions to the appropriate format.
2. Create a file with split points, to be used by the wav-splitter.

3. Create a batch file which:
   (a) invokes the wav-splitter 'shntool',
   (b) deletes the file with split points that was used by the wav-splitter,
   (c) invokes the 'sox'-program which converts the wav-files into raw-files.
   (d) deletes the wav-files.
4. Create a batch file which deletes the audio files and text files which do not contain any speech.

Transcriptions

The Sonic speech recognition system requires orthographic transcriptions of all the chunks of sound data. The CGN contains different kinds of transcriptions for the audio data, including the orthographic and phonetic transcription, part-of-speech tagging and word segmentation. The orthographic transcriptions required by Sonic, can be found in the 'ort'-category. Sonic requires orthographic transcriptions without punctuation marks so the punctuation marks are stripped by the prepareForShntool-program when they are converted into smaller text files. The CGN documentation [CorpusGesprokenNederlandsDocumentation] describes the transcriptions as follows:

> All the recorded material was transcribed orthographically. The orthographic transcription is a verbatim record of what was actually said. In the transcription process repetitions, hesitations, false starts and such were transcribed. Background noise, on the other hand, was seldom represented in the transcriptions. [...] the transcription has been checked manually. [...] While the transcription was being produced anchor points were introduced to mark off brief stretches of speech (approx. 3 seconds). Thus it became possible to identify words or phrases in the speech signal.

Pronunciation lexicon

To create an appropriate lexicon for Dutch, we import the elaborate database of the CGN and use only the two tables with the orthography of the words in the lexicon and their pronunciation in phonemes in Dutch. First, we only take the pronunciation from the Netherlands and eliminate the words that do not have such a pronunciation. These 330 records include some typically Flemish words such as "d'rhene" (meaning: "to there").

Sonic requires a pronunciation lexicon in the following format:

```
ACCIDENTAL      AE K S AX D EH N AX L
ACCIDENTAL(2)   AE K S AX D EH N T AX L
```

It is thus possible to include alternative pronunciations for a certain word, but a sequence number in brackets must be added to the orthographic word form. It is also possible to add normalized log probabilities to these alternative pronunciations. The most likely pronunciation has a log probability normalized to 0 while the less likely alternatives have a negative log probability. For example:

```
[0.00000] ACCIDENTAL      AE K S AX D EH N AX L
[-0.22185] ACCIDENTAL(2)  AE K S AX D EH N T AX L
```

The CGN offers an elaborate lexicon containing information on the token (word form), part of speech, lemma, syntax orthographic status, pronunciation, morphology and nature (continuous/discontinuous) of a multi-word expression. This is an example lines from the lexicon for the word 'dotatie':

> 75488\dotatie\N(soort,ev,basis,zijd,stan)\dotatie\24120\[DET:<de>]
> [HD:<dotatie>]\\dotatsi\dotasi\dotatsi\do-'ta-
> tsi\((doteer)[V],(atie)[N—V.])[N]\V\\

The only information we need from this is the word form and its three pronunciations. Firstly, the \-symbol is replaced automatically with tabs, using a basic text editor. Secondly, the lexicon text file is imported in a database and queries are performed on this database to filter out duplicate entries, i.e. entries which differ only in fields that are not needed for our lexicon. We now obtain a text file with comma-separated values in two columns: one with a word form and a second one with the word's pronunciation, without spaces. In order to add the spaces, a php-program is constructed. Spaces need to be inserted between two different phonemes, which include combined symbols such as A, A+ and A~. A regular expression was used in the php-program to determine where to insert the spaces. Now the lexicon is ready.

Alternatively, one could also use the frequency counts in the CGN to calculate an estimate of the normalized log probabilities for the alternative pronunciations but it would be better to use a larger text source for this goal. The text resources where however not available in time for this and taking into account that acoustic training costs lots of calculation time, we decide to go ahead with the construction of a lexicon without these probabilities.

5.3 Language Model

This section describes how the language model needed for the retraining of Sonic, was created. For the N-Best project, it is necessary to create a Dutch language model. To achieve the best results with the different acoustic models, we decide to train different language models for the different tasks. The base line Dutch ASR models are tested with a specific language model for this collection of data. This language model is trained using the transcripts from this speech data. For this we use the CMU-Cambridge Statistical Language Modeling Toolkit, which was introduced in section 4.4. Sonic accepts language model in the ARPA-format, which this toolkit can create.

A large training text is needed for this process. We thus create a Java program to collect 90% of the transcripts from the CGN in one text file, with phrase breaks inserted between phrases. The remaining 10% was used for a test of the language model. The phrase breaks are indicated by two tags: <s> to indicate the start of a sentence and </s> to indicate the end. The language model for a CGN recognition task would best be trained using actual CGN transcripts since this is the most representative data available. For the N-Best tasks, TNO was supposed to provide a language model training text in the form of a news corpus but did not deliver before this research was finished. This would have been especially useful for the BN recognition task. The language model training task is performed according to the guidelines from the web site [CMU].

The training process is done in a number of steps. First we let the tool count the frequencies with which the different words occur in the training text.

```
cat lm4_train.text | text2wfreq > lm4.wfreq
```

Next we generate a vocabulary containing the most frequent 65535 words, which is the maximum number of words allowed by the tool. This forces us to leave out ten thousands of words that are less frequent but just because they are less frequent, this is not a big problem. Using too big a vocabulary can even deteriorate the performance of an ASR system since this increases the chance of mistaking one word for another.

```
cat lm4.wfreq | wfreq2vocab -top 65535 > lm4.vocab
```

Now we can use this vocabulary and the training text together to create the N-grams.

```
cat lm4_train.text | text2idngram -vocab lm4.vocab > lm4bin.idngram
```

In the last step, we construct a language model from these N-grams. Many options are available for this step. We supply the file with N-grams, the vocabulary, an output file in the ARPA-format, a file listing the context cues ¡s¿ and ¡/s¿, an option to let the tool find out for itself how much memory to use. We also choose to use an open vocabulary model, which allows for out-of-vocabulary words to occur in the test text, since we are not sure that every word from the training text occurs in the vocabulary. These OOVs are treated as any other word in the vocabulary, following the example language model that comes with Sonic. This example also uses a Witten-Bell discounting strategy, which we also select. This discounting strategy is described in [Ian H. Witten and Timothy C. Bell, 1991]. The manual describes this discounting strategy as follows:

> The discounting ratio is not dependent on the event's count, but on t, the number of types which followed the particular context. It defines d(r,t) = n/(n + t), where n is the size of the training set in words. This is equivalent to setting P(w — h) = c / (n + t) (where w is a word, h is the history and c is the number of occurrences of w in the context h), for events that have been seen, and P(w — h) = t / (n + t) for unseen events.

```
idngram2lm -idngram lm4bin.idngram -vocab lm4.vocab
-arpa lm4.arpa -context context_cues2.ccs -calc_mem
-vocab_type 1 -witten_bell -bin_input
```

Now we can even evaluate the performance of this new language model on a training text. This is done with the following commands:

```
evallm -arpa lm4.arpa -context context_cues2.ccs
perplexity -text lm4_test.text -backoff_from_ccs_inc
```

We choose to back off from the context cues inclusively. This means that we use the sentence boundaries as boundaries for the trigrams. For the probability of the first word of a new sentence, we do not take the previous words into

consideration, since they are a part of the previous sentence. The same strategy can also applied to the OOVs but that did not influence the test results much. This test resulted in a perplexity of 272.70, which corresponds with an entropy of 8.09 bits. Now what does this mean? The perplexity is a unit from the information theory indicating how well the language model predicts the test text. The language model is in fact a probability distribution over the training text. It is described at as follows [Perplexity]:

> The perplexity for a probability distribution p is defined by the following formula:
>
> $$2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)} \tag{5.1}$$
>
> where $H(p)$ is the entropy of the distribution and x ranges over events. [...] Often one tries to model an unknown probability distribution p, based on a training sample that was drawn from p. Given a proposed probability model q, one may evaluate q by asking how well it predicts a separate test sample x_1, x_2, ..., x_N also drawn from p. The perplexity of the model q is defined as
>
> $$2^{-\sum_{i=1}^{N} 1/N \log_2 q(x_i)} \tag{5.2}$$
>
> Better models q of the unknown distribution p will tend to assign higher probabilities $q(x_i)$ to the test events. Thus, they have lower perplexity: they are less surprised by the test sample. [...]
>
> The lowest perplexity that has been published on the Brown Corpus (1 million words of American English of varying topics and genres) is indeed about 247 per word, corresponding to a cross-entropy of $log_2 247 = 7.95$ bits per word or 1.75 bits per letter.

When the N-Best organization provides us with the news corpus, we will be able to train a new language model for the BN task and can then compare the perplexity of the new language model with the current perplexity, to conclude which language model is better suited for this task.

5.4 Acoustic Retraining of Sonic

Now all resources are available, the acoustic retraining process can commence. In section 5.4.1 we will explain how this process is executed and in section 5.4.2 the practice will be discussed.

5.4.1 Approach

This process's structure is illustrated in figure 5.2. This process can be automatically executed via a collection of C-shell scripts. The port-sonic.csh script is the main script that calls the other scripts and thus guides the porting process. In order to let Sonic know where all the training data is located, the path-setup.csh script needs to be adjusted by the user. This script contains references to the lexicon, the phone list, the phone map, the sample rate, the letter-to-sound rules and a file containing decision tree questions. Below we will

look at the different scripts, named step0 to step4, in more detail. The steps are executed in the order depicted in figure 5.1:

port-sonic.csh	main script
path-setup.csh	setup of input/output
step0-kb.csh	lexicon compression / letter-to-sound
step1-align.csh	initial alignment (English)
step2-mlf.csh	generate master label files
step3-train.csh	gender-dependent HMM training
step4-realign.csh	realign files

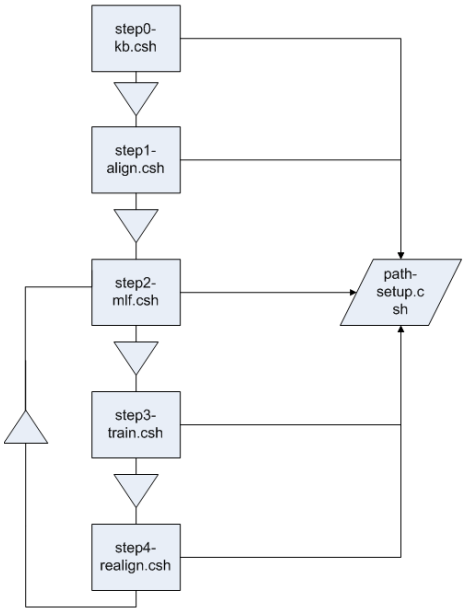

Figure 5.1: Sonic's acoustic retraining process. The path setup file defines where the input files are located and is used by the other scripts. The process is directed by port-sonic.csh (not depicted to improve clarity).

Steps 2 to 4 are repeated to improve the training results.

The C-shell scripts that come with Sonic, can not cope with speaker directories containing many audio files. The scripts thus were adapted by the author to circumvent this problem.

Step0 The first step in Sonic's acoustic retraining process prepares two required inputs: the pronunciation lexicon and the letter-to-sound rules. It all commences with a utility to construct a binary version of the lexicon: lex_encode. The binary version is a compressed form of the lexicon which improves the recognizer access speed. The binary version of the lexicon is also used throughout the training process. Consecutively we train the letter-to-sound module using the command line tools t2p_fea and t2p_train. The letter-to-sound module is used to generate the pronunciation of words that do no not appear in the lexicon. This is done with decision trees: trees with acoustic splitting questions. These decision trees split on the basis of letter context. This will be described in more detail in section 5.4.1. The tool t2p_fea automatically aligns letters and

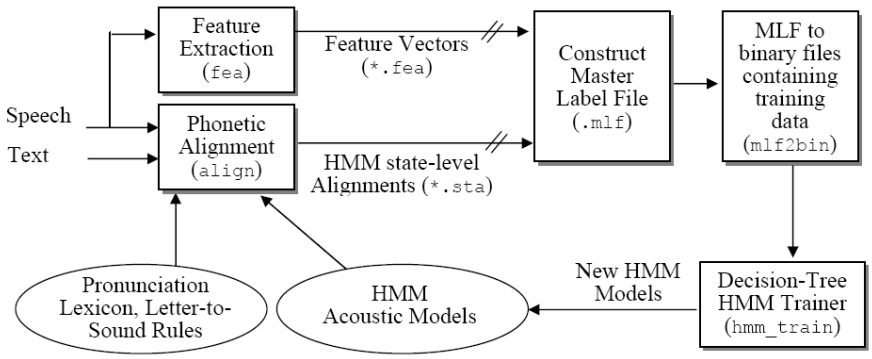

Figure 5.2: Sonic's acoustic model training process

phonemes on the basis of the input lexicon. This alignment is performed as shown in this example:

Letter Context							Desired Phoneme
_	_	_	*R*	O	A	D	R
_	_	R	*O*	A	D	W	OW
_	R	O	*A*	D	W	A	_
R	O	A	*D*	W	A	Y	D
O	A	D	*W*	A	Y	_	W
A	D	W	*A*	Y	_	_	EY
D	W	A	*Y*	_	_	_	_

Step1 During this step, the initial alignment of the data takes place. The data is initially aligned using the American English models and is realigned using the new acoustic models during step4. But before the alignment can take place, features have to be extracted from the audio data with the Sonic-tool named fea. This tool extracts MFCC features by default, in the form of 39-dimensional floating point vectors containing the 12 MFCCs and the normalized frame energy, along with their first and second derivatives. For every second of audio, 100 feature vectors are extracted.

Step2 The second step creates the master label files for each speaker. They are simple files containing the training feature file name followed by the state-level alignments of the phonemes. Whether the phoneme is at the beginning, middle or end of a word is also indicated. To give an example:

```
/home/rec/segment/com/cu/read_speech/20010621/000/sls-20010621-000-003.fea
0  14 15 15 16 16 SIL b
17 22 23 23 24 28 F   b
29 29 30 32 33 35 R   m
36 43 44 47 48 51 AE  m
52 53 54 54 55 58 N   m
59 64 65 73 74 76 S   e
77 98 99 123 124 128 SIL b
```

Step3 The training of the gender-dependent HMMs starts by calling the mlf2bin-tool from Sonic. This tool extracts the relevant data from the master label files. The state-based alignments of the training data are first converted into binary feature files for training. One temp file is created for each state of each base phone. These temp files are then used to train the acoustic models. The training process is described in the manual:

> During the training process, single-mixture triphones are estimated for each triphone occurrence in the training data. The data is then placed at the root node of the decision tree and splitting questions are evaluated. The question that results in the largest increase in likelihood for the training data is used to split the node. The splitting process continues until the likelihood change falls below a threshold or the number of frames assigned to the clustered state becomes too small. Finally, the data assigned to each leaf node in the tree is then used to estimate the mixture-Gaussian distributions. The HMM parameters (mean vectors, variances, mixture weights, and clustering information) are then written to disk. The decision tree rules can be specified in an ASCII-file. Rules are denoted by a rule-name (a '$' is put before each rule name) followed by a list of phonemes or rules that are included. For example,
>
> $nasal M N NG
> $voiced_stop B D G
> $unvoiced_stop P T K
> $stop $voiced_stop $unvoiced_stop
>
> An example comprehensive set of decision tree questions for US English are found in, sonic/2.0-beta5/doc/examples/dt.rules. Note that every phoneme specified in the phoneme set configuration file should appear at least once in the decision tree question set.

A splitting tree question set was derived from the English and German question set, added as appendix B and appendix C. The required information on the Dutch phonemes was derived from [für Deutsche und Niederländische Philologie FU Berlin, 2004b] and [für Deutsche und Niederländische Philologie FU Berlin, 2004a]. First we added the questions from the German and English question set that are also applicable to Dutch, with the appropriate phonemes from the CGN phoneme set. Some rules that are not present in the English and German set were added, for example the rule $foreign-vowel, derived from the CGN phoneme set. This resulted in the following rule set:

```
$silence SIL br ls lg ga
$aspiration x h
$schwa @
$s_z s z S Z
$nasal m n N J
$liquid l r
$glide w j
$diphtong E+ Y+ A+
```

```
$dental_plosive t d
$dental t d s z n l r
$labial_plosive p b
$labial p f b v m w
$palatal S J j
$velar k x G N r
$voiced_plosive b d
$voiceless_plosive p t k
$plosive p b t d k
$voiced_fric v z G Z
$voiceless_fric f s S x h
$fricative f v s z S x G h Z
$foreign_vowel E: Y: O: E  O  A  Y
$short_vowel I E A O Y
$long_vowel i y e 2 a o u
$nasal_vowel E  O  A  Y
$high-vowel i y u
$mid-vowel e I 2 Y: @ o O
$low-vowel E A a
$front-vowel i y e I 2 Y: E Y E: E+
$central-vowel $schwa
$back-vowel u o O A a O: A+
$rounded-vowel y 2 Y: u o O Y
$unrounded-vowel i e I E @ a A E: E+
$vowel I E A O Y i y e 2 a o u E+ Y+ A+ @ E: Y: O: E  O  A  Y
$consonant p b t d k g f v s z S Z x G h m n N J l r j w
$obstruent p f b v t s d z S k x G h
$sonorant m w n l r J j N $vowel
$voiced b v m w d z n l r J j G N $vowel
$voiceless p f t s S k x h
```

Step4 The last step in the porting process is the realignment of the audio files using the newly trained models.

The approach chosen to create the models for the BN and the CTS tasks, is the following: first we train basic Dutch speech models for Sonic using a large amount of training data from all available categories from the CGN and, when a reasonable recognition rate is reached, retrain these models specifically for the BN and CTS task. The same process could then be undertaken for the Flemish data.

5.4.2 Results

The training process of the basic Dutch speech models results in the following recognition rates for complete audio files shown in table 5.7.

Sentence Level Input

I now try to improve this recognition rate. Firstly, we decided to input only sentences to the ASR system instead of whole audio files. Recognizing only one

Table 5.7: Test Results: Standard Recognition After Three Training Steps

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	1101	358594	**9.3**	22.7	68.0	7.5	98.2	100.0
Mean	3.7	1211.5	13.1	38.0	48.9	95.2	182.1	100.0
Standard Deviation	11.0	2212.2	10.9	26.7	32.1	312.0	307.9	0.0
Median	1.0	348.5	10.1	31.8	52.5	1.0	95.9	100.0

sentence at a time significantly simplifies the ASR task: it gives important information to the ASR system. When supplying complete audio files with multiple sentences to the ASR system, it has to determine by itself where the sentence boundaries are located and with an average of 3.7 sentences for 1211.5 words, we see that Sonic inserts very few sentence boundaries. Too few sentence boundaries hinder correct usage of the language model since probabilities of words at the beginning of sentences are then wrongfully based on words from the previous sentence. The manual however mentions nothing about sentence boundaries being inserted but it is often the case that ASR systems expect the input files to contain only one sentence. A preprocessing step is usually done to split the audio files at long silences, since people tend to pause longer between sentences than between words. For the training data, we can insert sentence boundaries, and split the audio and text files, based on the sentence boundaries in the transcripts. This requires adapted Java programs to enable the splitting at different points and to create new transcript files. When I finish these adaptations, it results in the improved recognition rate in table 5.8:

Table 5.8: Test results: standard recognition for sentences

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	41657	358380	**18.9**	41.9	39.2	5.0	86.1	95.7
Mean	140.7	1210.7	20.2	39.6	40.2	3.5	83.3	97.0
Standard Deviation	329.3	2211.4	12.6	11.3	15.3	10.4	16.8	4.8
Median	39.5	348.5	18.8	39.2	38.8	1.6	83.5	98.7

Grammar Scale Factor

Secondly, I experiment with the grammar scale factor. This scaling factor adjusts the weight the language model has in the decoding process, at the expense of the acoustic model's weight. There is a separate scaling factor for the first pass, which typically ranges from 20 to 30, and one for the second pass, typically between 10 and 30. This adaptation improves the recognition rate some more, see table 5.9. The standard grammar scale factors are not mentioned in the manual so we do not know whether they have been increased or decreased.

Other grammar scale factors could be tried and could improve the recognition rate with a few extra percentages. However, since this is just a base system that will be retrained and not used as is, and every test takes two days to complete and the time available is limited, we decide not to do this at this time. More elaborate experimentation with the grammar scale factor will be done with the final models for the BN task in section 5.5.3. We now continue by retraining the models.

Table 5.9: Test results: recognition for sentences with a different scaling factor

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	41637	358262	**24.3**	47.1	28.5	8.2	83.9	95.6
Mean	140.7	1210.3	26.6	44.9	28.5	6.1	79.5	96.6
Standard Deviation	329.3	2211.6	16.2	12.7	15.6	16.0	23.2	5.6
Median	39.5	348.5	25.5	44.0	25.7	3.1	77.8	98.0

Iterative Retraining

Finally, we look at the effects of extra training steps. After one extra training round, the recognition rate improves again to the results in table 5.10.

Table 5.10: Test results: recognition for sentences after retraining

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	41657	358380	**24.9**	47.1	28.1	8.7	83.9	95.7
Mean	140.7	1210.7	27.5	44.2	28.4	6.8	79.4	96.8
Standard Deviation	329.3	2211.4	16.6	13.4	16.0	20.0	26.3	4.3
Median	39.5	348.5	25.9	43.2	25.7	3.2	77.3	98.0

However, another extra training round results in a recognition rate of 24.8% instead of the 24.9% we already had achieved and another training step again makes the recognition rate a bit worse: 24.5%. The recognition rate no longer improves with extra training steps so this apparently is the highest recognition rate that can be achieved with the bare Sonic ASR system for this very broad task, leaving aside potential, probably small, gains from other grammar scale factors.

Sonic's Distributed Training Sonic also features distributed training using multiple servers, to speed up the training process. One central client application with the training data then sends jobs for separate phonemes to the servers. However, these server applications often crashed and then it was necessary to restart the whole training process or manually select the erroneous phonemes

and restart the training process specifically for them. This took more time then using only one server for the calculations so we stopped using this feature.

Discussion

We should keep in mind that all categories except for the broadcast news and telephone conversations are being used for this test, for the training of the acoustic models as well as for the tests. These categories together form a broad collection of speech with different subjects and various acoustic circumstances. These categories must share one general language model and acoustic model when they are combined in one recognition task. This makes it a very hard ASR task to fulfill. This is also the reason that the N-Best project only focuses on two more limited tasks: the broadcast news and conversational telephone speech task. But at this moment, we are only training a Dutch base ASR system, which does not have to be perfect since the acoustic models will still be retrained for the specific tasks. These tasks are narrower and thus easier to create specialized acoustic models for. Therefore, we can expect a much better recognition rate for specialized models for these tasks. At this point, we thus decide to leave these models as they are and move forward to retrain them for the BN task, which is easier than the CTS task for numerous reasons. The BN task contains read speech as opposed to the very spontaneous speech in the CTS task. Read speech is always much easier to recognize since it contains mainly grammatically correct sentences, for which it is easier to create a language model. Only for an occasional misread, the sentences contain some replacements and maybe some repeated word groups. Spontaneous speech, on the other hand, contains many filler words and sounds which may differ from speaker to speaker. Spontaneous speech also contains many ill-formed phrases, since speakers do not write and correct the sentences before they start speaking but simply begin talking and finish the sentence as they go along. This results in errors, restarts and interjections, which all complicate the ASR process. This is described in section 5.5.

5.5 The Broadcast News Task

Now we have a baseline system to recognize the Dutch language, we can specialize it for a more specific task: the BN task. This is done by retraining the baseline system using the CGN data from the corresponding categories and creating a new language model.

5.5.1 Retraining

For the BN task, we use acoustic training data from the CGN categories f, i, j, k and l, as specified by the N-Best project in table 3.3. Firstly, this data is split up in small chunks. Then the acoustic retraining process of Sonic can be repeated, using this data as the training data. The first retraining step then results in the statistics in table 5.11, when tested with the CGN language model. We see the total number of sentences and words and the percentage of correctly recognized words and the types of errors: words that have been substituted by other words and words that have been deleted or inserted. This sums up to a percentage of errors and the last column gives the percentage of sentences that

contain an error. The mean, standard deviation and median are also calculated. The most important figure in these results is the percentage of errors, which we want to minimize. This is the main indicator of the ASR system's performance.

Table 5.11: Test results: recognition of BN sentences after one retraining step

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**46.6**	34.2	19.2	10.1	63.5	91.8
Mean	59.6	665.3	44.1	36.8	19.1	7.6	63.5	93.9
Standard Deviation	75.8	898.8	14.2	10.7	14.4	15.4	21.4	8.1
Median	29.5	309.0	44.3	37.0	16.3	5.4	61.2	96.1

To see whether this recognition rate can be approved upon, we try a second retraining step. The extracted features are realigned with the transcriptions, using the newly trained models as the acoustic model. The acoustic model is then retrained. The results are shown in table 5.12.

Table 5.12: Test results: recognition of BN sentences after two retraining steps

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**47.0**	34.1	18.8	10.4	63.4	91.9
Mean	59.6	665.3	44.4	37.0	18.5	8.0	63.6	93.9
Standard Deviation	75.8	898.8	14.7	11.3	14.6	14.6	21.2	8.1
Median	29.5	309.0	45.1	37.5	15.3	6.2	61.2	95.8

Since the recognition rate is still improving, we perform a third retraining step, resulting in table 5.13. The models are no longer improving so we stop retraining them and continue with other measures of improving the recognition rate.

Table 5.13: Test results: recognition of BN sentences after two retraining steps

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**47.0**	34.1	18.8	10.4	63.4	91.9
Mean	59.6	665.3	44.4	37.0	18.5	8.0	63.6	93.9
Standard Deviation	75.8	898.8	14.7	11.3	14.6	14.6	21.2	8.1
Median	29.5	309.0	45.1	37.5	15.3	6.2	61.2	95.8

5.5.2 The Language Model

For the BN task, we first use the same language model as we did for the complete CGN task. We would have preferred to use a new language model, trained on the training data that would be provided by TNO. However, this data was not provided to us in time so instead, we create a new language model based on the transcripts from the CGN categories that are included in the BN task. This new language model is created according to the process described in section 5.3. It achieves a perplexity of 304.63 on the test set, which corresponds to an entropy of 8.25 bits. This measure is most useful for comparing two language models so we also calculate the perplexity for this task using the old language model, based on the transcripts from the complete CGN. This model, surprisingly, achieves a perplexity of 155.09, corresponding with an entropy of 7.28 bits. This seems to indicate that the old language model, based on all the CGN transcripts, better predicts the BN test data. This is highly unlikely and it was already noted by [Clarkson and Robinson, 1999] that "There are many examples of cases in which a language model has a much lower perplexity than the baseline model, but does not result in a reduction in WER, and often results in a degradation in recognition accuracy". So the best test is to use the new language model for the BN test data and this gives us the much better test results in table 5.14.

Table 5.14: Test results: recognition of BN sentences using a new BN language model

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**64.6**	19.0	16.4	7.5	42.9	76.7
Mean	59.6	665.3	62.0	21.7	16.3	5.5	43.6	81.0
Standard Deviation	75.8	898.8	18.0	10.8	15.3	14.1	24.2	15.5
Median	29.5	309.0	63.3	20.9	12.2	3.2	39.6	83.1

The new language model improved the recognition rate with an astonishing 17.6%! This approaches the recognition rate for the Dutch BN task using Sonic as achieved by TNO during the course of the N-Best project: 70% [Judith Kessens and David van Leeuwen, 2007]. Furthermore, the system that achieved this recognition rate of 70%, used a language model trained "a large Dutch news related text corpus of in total some 400M words" [Huijbregts, M.A.H., Ordelman, R.J.F., de Jong, F.M.G., 2005]. During the N-Best project, such a corpus will also be provided but this research was finished before this training data was provided by TNO. This could explain the small gap in recognition rates. But there is another factor that can still improve on the current score: the grammar scale factor.

5.5.3 The Grammar Scale Factor

We remind the reader that this scaling factor adjusts the weights the language model and the acoustic model have in the decoding process. Tuning this factor already improved the recognition rate of the baseline system to the results in

table 5.9 and we now investigate the effects this scale factor has on the BN system.

There are two grammar scale factors that can be tuned: one for the first pass and one for the second pass. The previous results for the BN task were achieved with a grammar scale factor of 20 for the first pass and 10 for the second pass. These are the lowest typical values, according to the Sonic manual. They typically can go up to 30.

First we adapt the first pass grammar scale factor and keep the other one stable at 10. With a grammar scale factor of 25, we achieve the results in table 5.15.

Table 5.15: Test results: recognition of BN sentences using grammar scale factors 25 and 10

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**64.6**	17.9	17.5	6.5	41.9	75.0
Mean	59.6	665.3	62.0	20.4	17.5	4.5	42.5	79.8
Standard Deviation	75.8	898.8	18.3	9.6	15.5	12.8	23.7	14.7
Median	29.5	309.0	65.7	20.0	14.3	2.4	37.3	81.6

This does not change much so we increase the first grammar scale factor to 30, the results are in table 5.16.

Table 5.16: Test results: recognition of BN sentences using grammar scale factors 30 and 10

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**62.2**	18.0	19.8	5.3	43.2	74.8
Mean	59.4	665.3	59.4	20.1	20.4	3.8	44.3	79.6
Standard Deviation	75.8	898.8	18.6	8.9	15.8	11.2	22.8	15.1
Median	29.5	309.0	63.5	19.6	16.9	2.0	38.9	80.2

This leads to less performance so we decide to keep the first grammar scale factor at 25. Now we experiment with the second grammar scale factor. Changing this to 20 results in table 5.17.

Again there is little effect. We decide to try a value right in between to see whether there is a peek in performance or the performance is steady for this range. The results for the test with grammar scale factors 25 and 15 are listed in table 5.18.

To cover all the bases, we will again try a higher grammar scale factor for the second pass: 30. We see in table 5.19 that this has no influence whatsoever on the test results.

The last test I did, used an even higher grammar scale factor for the second pass: 35. This again changes completely nothing for the test results. The

Table 5.17: Test results: recognition of BN sentences using grammar scale factors 25 and 20

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**64.6**	17.9	17.5	6.5	41.9	75.0
Mean	59.6	665.3	62.0	20.4	17.5	4.5	42.5	79.8
Standard Deviation	75.8	898.8	18.3	9.6	15.5	12.8	23.7	14.7
Median	29.5	309.0	65.7	20.0	14.3	2.4	37.3	81.6

Table 5.18: Test results: recognition of BN sentences using grammar scale factors 25 and 20

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**64.6**	17.9	17.5	6.5	41.9	75.0
Mean	59.6	665.3	62.0	20.4	17.5	4.5	42.5	79.8
Standard Deviation	75.8	898.8	18.3	9.6	15.5	12.8	23.7	14.7
Median	29.5	309.0	65.7	20.0	14.3	2.4	37.3	81.6

Table 5.19: Test results: recognition of BN sentences using grammar scale factors 25 and 30

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**64.6**	17.9	17.5	6.5	41.9	75.0
Mean	59.6	665.3	62.0	20.4	17.5	4.5	42.5	79.8
Standard Deviation	75.8	898.8	18.3	9.6	15.5	12.8	23.7	14.7
Median	29.5	309.0	65.7	20.0	14.3	2.4	37.3	81.6

grammar scale factor for the second pass seems to have no effect at all on the recognition results. We now stop experimenting, concluding with the recognition rate of 64.6%.

5.5.4 Calculation Time

The N-Best project allows participants to draw attention to their specific strong points by sending in supplementary test results achieved under a contrastive condition. The Sonic manual already mentioned that the system was built "with careful attention applied for speed and efficiency" and this clearly had effect: the final models for the BN task recognized the test data at a calculation time of 0.83 RTs. This means that this system can be used as is to enter under the strictest contrastive condition, namely real-time.

Table 5.20: Test results: recognition of BN sentences using grammar scale factors 25 and 30

	Sentences	Words	Correct	Substitutions	Deletions	Insertions	Errors	Sentence Errors
Sum/Avg	9777	109108	**64.6**	17.9	17.5	6.5	41.9	75.0
Mean	59.6	665.3	62.0	20.4	17.5	4.5	42.5	79.8
Standard Deviation	75.8	898.8	18.3	9.6	15.5	12.8	23.7	14.7
Median	29.5	309.0	65.7	20.0	14.3	2.4	37.3	81.6

5.5.5 Conclusions

Porting speech recognizers is a process of trial and error. It takes much preparatory work to mold all the data in the right format. When this is finished, a lot of experimenting needs to be done to optimize the performance of the ASR system. Using specialized language models is clearly crucial for the performance of any ASR task; the language model had the most influence on the test results. Acoustic retraining of the models to make them more specialized for the specific task, is also very important. Another aspect of the ASR system that are of influence and that needs to be experimented with, is the grammar scale factor, especially for the first pass, when using a multi-pass ASR system.

There are still some parameters left that one can experiment with and that could have a minor influence on the results. Especially for spontaneous speech, the usage of a file with filler words and tweaking the associated filler word penalty, can potentially be very useful. One could also employ on of the many speaker and environment adaptation algorithms that are included in the Sonic ASR system. I focused the available time on the factors that are commonly believed by experienced ASR researchers to be the most influential.

Chapter 6

The Influence of Gender and Age

> Everything becomes a little different as soon as it is spoken out loud
>
> *Hermann Hesse*

In this chapter, we investigate the influence of gender and age on acoustic features of speech from the CGN. The speakers are classified into different age groups and their properties are then compared to see whether there are significant differences. Then we can conclude whether ASR systems can benefit from taking speaker age into account, or not.

The first section explains the choice of the different age categories. Section 6.2 describes how the data for the different groups was collected and processed.

6.1 Age Category Selection

The speech data used for this research, is a portion of the CGN. This corpus already contains annotations for the different audio files with specific age categories. These are the different categories used in the CGN:

- age0 = under 18 years of age,
- age1 = 18-24 years of age,
- age2 = 25-34 years of age,
- age3 = 35-44 years of age,
- age4 = 45-55 years of age,
- age5 = over 55 years of age,
- ageX = age unknown.

Category age0 unfortunately does not contain enough data with a phonetic annotation for this research. It will thus be excluded from the results. An addition to the CGN is however being developed containing speech data specifically from younger and elderly people [Catia Cucchiarini, Hugo Van hamme, Felix Smits, 2006] as well as from non-native speakers and speech data collected in a human-machine interaction setting. This new corpus will allow for more detailed research on this specific groups. Categories age1 to age5 contain balanced amounts of speech data which is convenient for this research. So we go ahead and use these CGN age categories.

6.2 Data Collection

In order to research the speech data on acoustic differences between ages, we need phonetic transcriptions to determine which phonemes are pronounced in the speech data. The speech data can then be aligned with these transcriptions so we know which feature vectors correspond with which phonemes. So first we look at the phonetic transcriptions.

6.2.1 Phonetic Transcriptions and Wav-Splitting

The CGN contains broad phonetic transcriptions for a part of the audio data. These phonetic transcriptions are spread across most categories. They were created by first automatic generating a phonetic transcription and then checking and correcting this transcription manually, in two steps by two different transcribers. The term 'broad' indicates that there was no allophonic variation or diacritics[1] in the predefined phoneme set [CorpusGesprokenNederlandsDocumentation].

Table 6.1 lists the amounts of phonetically transcribed words available in the CGN.

The phonetic transcriptions are available in the CGN in text files with the extension .fon. They are split up in chunks of a few words, using the same time markers as the .ort-files. The .fon-files are in the ShortTextGrid-format, which was already described in section 4.1. Sonic requires the transcriptions to be in plain text-format containing words from the vocabulary or with actual phonemes. In order to use the phonetic transcripts with Sonic, we have to convert the ShortTextGrid-format from the CGN to a text files with sequences of phonemes, with spaces in between and each phoneme proceeded by a '!'-symbol. There also are some special symbols in the .fon-annotations that require a special treatment during the data processing:

[1] A mark, such as the cedilla of façade or the acute accent of resumé, added to a letter to indicate a special phonetic value or distinguish words that are otherwise graphically identical. - The American Heritage Dictionary of the English Language

Table 6.1: Categories in CGN

	Category	**Flemish words**	**Dutch words**
a.	Spontaneous conversations ('face-to-face')	70,945	106,182
b.	Interviews with teachers of Dutch	34,064	25,687
c.	Spontaneous telephone dialogs (recorded via a switchboard)	68,886	201,141
d.	Spontaneous telephone dialogs (recorded on MD with local interface)	6,257	0
e.	Simulated business negotiations	0	25,485
f.	Interviews/discussions/debates (broadcast)	25,144	75,106
g.	(political) Discussions/debates/meetings (non-broadcast)	9,009	25,117
h.	Lessons recorded in the classroom	10,103	25,961
i.	Live (e.g. sports) commentaries (broadcast)	10,130	24,986
j.	News reports/documentaries (broadcast)	7,679	25,065
k.	News (broadcast)	7,305	25,296
l.	Commentaries/columns/reviews (broadcast)	7,431	25,071
m.	Ceremonious speeches/sermons	1,893	5,184
n.	Lectures/seminars	8,143	14,913
o.	Read out speeches	64,848	70,223
	Total	**331,837**	**675,417**

`[]` Not transcribable (garbage). This chunk should be removed since it does not correspond with a phoneme.

`♯` Speaker sound(ggg). This chunk should be annotated with the symbol 'ggg' so Sonic understands it is a speaker sound.

`_` Shared phoneme(s) (dAn _nu /lE+st _stan). From a phonetic point of view, these double phonemes are redundant so we remove the _as well as the double phoneme(s).

`-` A phoneme placed in between of the normal phonemes (nu-w-Is). The fact that this sounds should not be there according to the orthography, is irrelevant for the phonetics so we remove the '-'-symbols.

A new Java-program, PrepareForShntool, is created to build phonetic transcription files for Sonic and prepare a batch-file for the splitting of wav-files in smaller wav-files at the time markers that also split the annotation files. This batch-file is then executed and calls the splitting tool, shntool, for the separate wav-files. PrepareForShntool also makes sure that the file parts from different age categories are saved in separate folders and afterwards are converted to the raw-format using the Sox-tool. A file with speaker-age combinations in the CGN was generated by Pascal Wiggers' software, to facilitate the spreading of data across folders according to speaker age. Afterwards, empty directories are removed by another Java program named AfterShntool.java. All the software created for the age-related research is contained in a separate package from the Java programs used for the porting of sonic in chapter 5.

6.2.2 Feature Extraction

Now we have folders containing small wav-files with corresponding phonetic transcriptions, we can extract feature vectors from them. For this goal, a csh-script is created that calls the fea-tool from Sonic for each of the files. This tool is used for this research with the following options:

```
-f 16000 Input audio sampling rate in Hz (default is 8000 Hz)
-a       Output feature vectors in ASCII format
-l       Batch-mode feature extraction. The file input argument
is an ASCII file containing an input/output pair on each line
separated by space or tabs. The feature extraction module will
read each audio file listed and output the corresponding
features to the output file. This option can be used with the
-b flag to simulate live-mode feature extraction. If this
option is used, the final fea arguments of an input and output
mfc file can be ignored.
```

Curiously the options -a and -l turn out to work just fine separately but result in an error when used jointly. So we create a csh-program to create another csh-program with on each line one call of the fea-tool for one audio file. Now we are ready to extract the features for the phonetically transcribed audio files. The feature vectors have a length of 39 and consist of:

```
Outputs 39 dimensional feature:
          - 12 static
          - 12 Delta
          - 12 Delta-Delta
          - 1  Normalized Power
          - 1  Normalized Delta Power
          - 1  Normalized Delta-Delta Power
```

The manual mentions that, for MFCC features, the power is normalized to lie between -0.5 and +1.0. The command line information indicates that the power of the PMVDR features is also normalized but it is not mentioned to what boundaries, not even in the paper on PMVDR to which the manual refers [Umit H. Yapanel, John H.L.Hansen, 2003]. This leaves us in the dark about which feature is the normalized power.

6.2.3 Alignment

Now we have the feature vectors needed for this research, we still need to align the audio files with the phonemes. Sonic also provides a tool for this means called 'align'. The manual describes many options for this tool:

```
Input Options
-bat <file> N
Batch-mode alignment. Batch-mode input file should contain
command-line options (one complete command-line per line of
the file). The aligner will read this file and process one
file at a time as specified in the command-line parameters.
```

```
-t File containing text-transcription to align. Can contain
embedded phoneme sequences using the "!" character followed
by the phoneme name.
User-defined Model Selection Options
-phone_config <file> File Specify a user defined phone set
configuration file.
-phone_map <file> File Specify a phoneme mapping file. This
file is used to map phonemes in a new language to phonemes
in an existing language.
-model <file> N User defined binary acoustic model as built
from the model training process.
-model_rate <float> N Set the sampling rate of the input
acoustic models (used in conjunction with the -mod option to
specify the acoustic models and the model's sampling rate).
This option allows the aligner to operate on 22kHz sampled
audio while utilizing acoustic models trained at a lower
sampling rate (e.g., 8kHz, 16kHz, etc.)
-feature_type N Sets the internal feature type. Current valid
feature types include MFCC, and MFCC_C0. Default is MFCC
feature type.
-lex_file N User specified lexicon
-lts_file N User specified letter-to-sound model
Output Options
-ow <file> N File for output word-level alignments. Contains
begin & end sample, word
-op <file> N File for output phone-level alignments. Contains
begin & sample, and phoneme
-os <file> N File for output state-level alignments. Contains
begin/end frame for each state followed by phoneme. This file
is used during acoustic model training.
```

An example of its usage is given in the manual:

```
align -mod comm-male.mod -model_rate 8000 -feature_type PMVDR
-con -t example.txt -ow example.wrd -op example.phn -os
example.sta -v -sil -f 8000 example.raw
```

As it turns out, the manual is incorrect when stating that the MFCC-features are the standard feature set. Segmentation faults occur when the newly trained Dutch models for Sonic are applied. The command-line help surprisingly states that the 'feature type (def. is PMVDR)'. It was the intention to do to the research on MFCC-features since they are still the most widely used feature type in the field of ASR but with this discovery at a late stage in the research, it was decided to continue with the PMVDR features. So all features in this research are of the PMVDR-type. Unfortunately, this type of acoustic features is claimed to contain less speaker-dependent information. We will be able to verify this statement.

Another difficulty is the use of the mod-option of the align-tool. Although it was not mentioned in the list of options, this option turns out to be necessary for the alignment using context-dependent models, which we trained in chapter 5. The align-tool accepts bath files as its input, to prevent having to call the tool for each file separately. This option turns out to work well so we use it.

After testing what specific combination of options is needed for the research and does not generate any segmentation faults, the alignment can commence. The alignment is done via two csh-scripts adapted from the scripts that came with Sonic, named step1-align.csh and step1b-align.csh. These scripts pass through all categories and align each speaker separately. For each speaker, the gender is read to choose the appropriate acoustic model and then all files are added to a batch-file, which was then given to the alignment tool for batch processing. The alignment was done with the following options, that were explained above:

```
-v
-model $model
-model_rate $samprate
-phone_config $phone_config
-lex_file  $lexicon
-lts_file  $let2snd
-sil
-con
-f  $samprate
-t  $txt
-ow $wrd
-op $phn
-os $sta
$rawfile
```

We use the best Dutch acoustic models we have previously trained for the alignment. This results in a set of state-based alignments. They have the extension .sta and look like this:

```
0  14 15 15 16 16 SIL b
17 22 23 23 24 28 F   b
29 29 30 32 33 35 R   m
36 43 44 47 48 51 AE  m
52 53 54 54 55 58 N   m
59 64 65 73 74 76 S   e
```

In this example, the first state of the phoneme /F/ lasts from frame 17 till frame 22. The second state of this phoneme is aligned with only one frame: number 23.

6.2.4 Feature Collection

Now we can collect the feature vectors belonging to each separate phoneme state. A new Java program is created for this task, called PrepareForMatlab.java. This program reads in every .sta-file which frame of the current audio file belongs to which phoneme state. Then it reads the feature vectors belonging to this phoneme state from the .fea-file and adds them to a text file specific to this age group and this phoneme state. Now we have a number of text-files containing feature vectors, one file for each phoneme state in one age category. Now we can read these text files with Matlab and research the data.

6.3 Data Analysis

When comparing this feature data, not all phonemes can be investigated: some of the phonemes simply occur not often enough in the audio data to base any conclusions on, for example the phoneme /A~/ as in "croissant". To obtain the most reliable results, we use the phonemes for which we have the most feature data. Since these phonemes occur the most, they should have the most influence on the recognition process anyway.

First, we write a Matlab-function ReadFeaFile.m to read in the feature data available for a specific phoneme, for speakers of one age category and one gender. This function is used by the other functions, that will plot the data.

6.3.1 Feature Averages

In the research on lexical stress [Rogier van Dalen, 2005], acoustic differences where measured between stressed and unstressed syllables. This resulted in differences in the speech signal's energy, including different average values for the energy factor in the feature vectors. To observe whether there are significant differences between the different age categories, we plot the average values per phoneme state for speech data from one gender and two different age categories. We then observe this plot to see which feature averages differ the most, if any. This gives us plots like in figure 6.1. The differences are not significant for this particular phoneme state and gender and these two age categories.

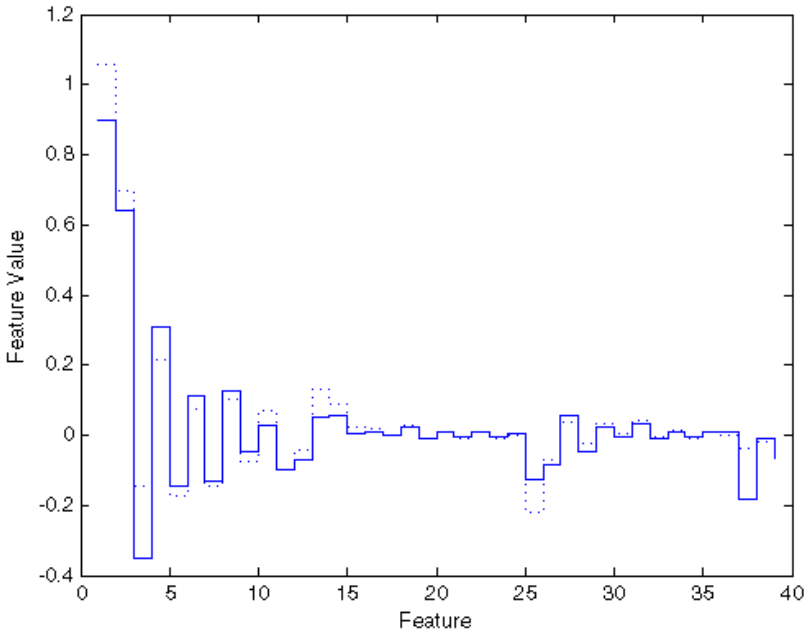

Figure 6.1: Averages for 39 features for the /@/, state 2 for female speakers from age category 1 (solid line) and category 2 (dashed line).

Other phonemes do have different average values for speakers from one gen-

der and two different age categories, for example the phoneme /d/ state 2 in figure 6.2.

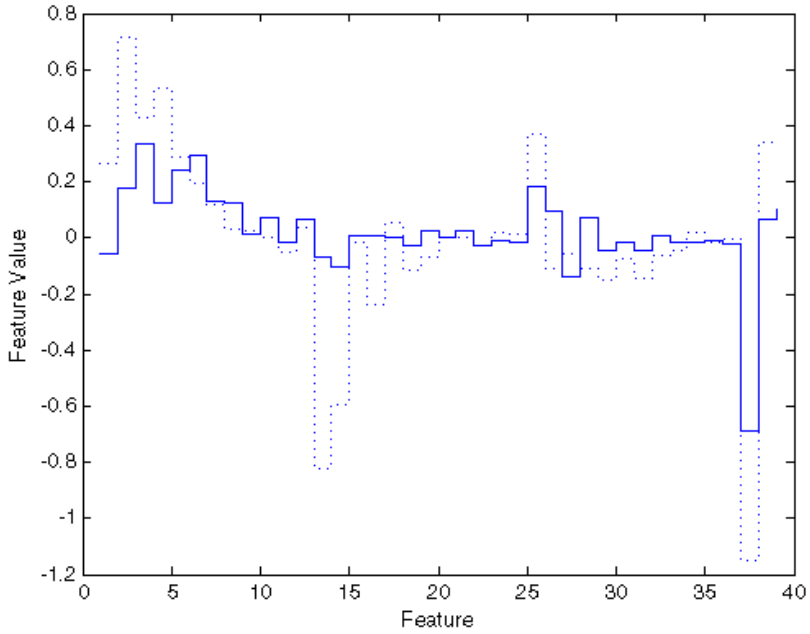

Figure 6.2: Averages for 39 features for the /d/, state 2 for female speakers from age category 1 (solid line) and category 3 (dashed line).

The average values range from roughly -6 to +2.5. In order to get a rough idea of where the important differences can be found, we decide to use a threshold for the average difference: 0.5 is sufficient to be noticed. Afterwards, we can take a closer look at these specific phoneme state's feature value distributions. Indexing which phoneme states have a difference above the threshold, results in table 6.2 for female speakers and table 6.4 for male speakers. Some observations can already be made from these graphs.

We see that for speech from female speakers, more phonemes differ and more features per phoneme differ. This is surprising since phonetics indicate that there are larger age differences between speech from male speakers. However, it is indicated that there is a difference in performance when applying PMVDR features for speech from female speakers, compared with male speakers. A much higher relative improvement was achieved for female speech than for male speech, compared with MFCC features: 35.5% compared with 19.0%, respectively [Umit H. Yapanel, John H.L.Hansen, 2003]. It is suggested that this is a result from the suitability of PMVDR features to high-pitch speech.

The first feature differs most often. For male as well as female speakers, the most differences can be found between speakers from age categories 1 and 3, and secondly between categories 2 and 3. This suggests that the changes occurring around the transition from category 2 to category 3, 35 years, already take a slow start in category 1 and then speed up during category 2. They

diminish before category 4. There are hardly any differences between female speakers from age categories 1 and 2, 3 and 4 (also between 2 and 4, 3 and 5). Differences between age categories 4 and 5 do occur, but rarely.

For one phoneme, we often see the age influences recurring for the same feature(s), for example feature 4 for /l/ and feature 2 for the /r/. These features probably stem from a range in the spectrum that is primordial for this phoneme. This relationship was also observed by [Rogier van Dalen, 2005].

Usually, ASR systems do incorporate provisions for different genders. Comparing the differences between the two genders, can provide us with a reference on these age differences. The differences between male and female speech, in these PMVDR features, are added in table 6.4. Surprisingly, there are almost no differences between the phonemes originating from male and female speakers. It was mentioned in the paper on Sonic's PMVDR features [Umit H. Yapanel, John H.L.Hansen, 2003] that these features remove much more speaker dependent information than MFCC features do. Apparently, the PMVDR features contain less gender-specific information than age-specific information. Clearly, not all speaker-dependent information is filtered out. Now we shall compare the feature vectors in more detail by looking at their distributions.

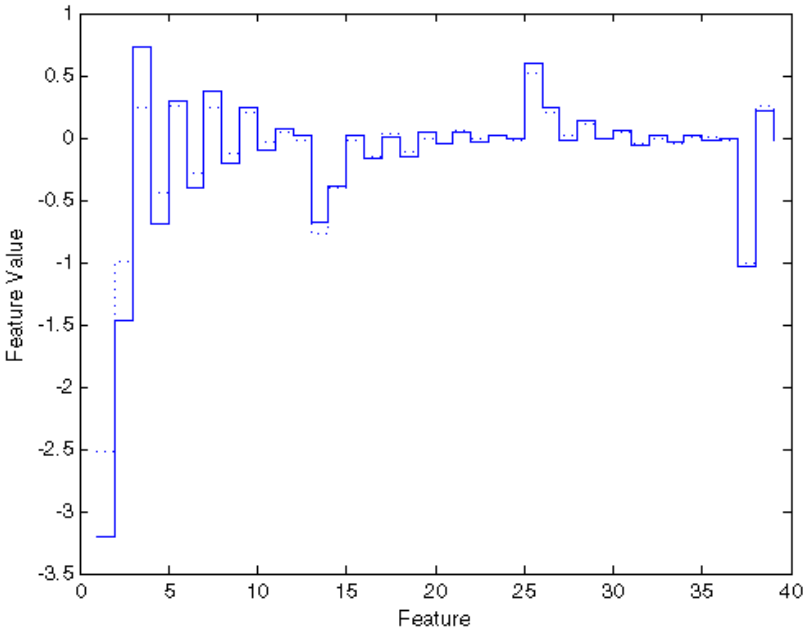

Figure 6.3: Averages for 39 features for the /t/, state 2 for female speakers (solid line) and male speakers (dashed line).

6.3.2 Feature Distributions

To get a more detailed view on the feature qualities, we will compare their distributions, following the approach used by [Rogier van Dalen, 2005]. We start with the gender-specific feature vectors. As said in the previous section,

Table 6.2: Female speakers: features with averages above the threshold

Phoneme	**Age1 Age2**	**Age2 Age3**	**Age3 Age4**	**Age4 Age5**	**Age1 Age3**	**Age2 Age4**	**Age3 Age5**
@1	-	-	-	-	13	-	-
@2	-	-	-	37	-	-	-
@3	-	-	-	-	1	-	-
n1	-	1	-	-	1	-	-
n2	-	1	1	37	1	1	-
n3	-	1,13	-	37	1,2,13	-	-
e1	-	3	-	-	3	-	-
e2	-	3	3	-	3,5	-	-
e3	-	3	-	-	3	-	-
t1	-	1,37	-	1,37	1,2,3,37	-	-
t2	-	2,3	-	2	2,3,13	-	-
t3	-	1	-	-	1,3,25,37	1	-
s1	1	1,2,13	-	1	1,2,13,37	1	-
s2	-	1,2	-	1	1,2,3	1	-
s3	1	1,2,25	-	1,2	1,25,37	1	-
r1	-	2	2	-	2	-	-
r2	-	1	2	-	2,3	-	-
r3	-	-	-	-	2,3	-	-
m1	-	1	-	37	1	1	-
m2	-	1	1	-	1	1	-
m3	-	1	1	-	1	1	-
d1	-	1,2,37	-	37	1,2,37	-	-
d2	-	13,37	-	-	2,13,14,37	-	-
d3	-	25	-	-	25	-	-
k1	-	37	37	37	4,37	-	1
k2	-	3	-	-	3,14,38	-	-
k3	-	1	-	-	1,2	-	-
l1	-	4	-	-	1,4	-	-
l2	-	-	-	-	1,4	-	-
l3	-	-	-	-	1,4	-	-

Table 6.3: Male speakers: features with averages above the threshold

Phoneme	**Age1 Age2**	**Age2 Age3**	**Age3 Age4**	**Age4 Age5**	**Age1 Age3**	**Age2 Age4**	**Age3 Age5**
@1	-	-	-	-	-	-	-
@2	-	-	-	-	2	-	-
@3	-	-	-	-	-	-	-
n1	-	-	-	-	1	-	-
n2	-	-	-	-	1	-	-
n3	-	-	-	-	-	-	-
e1	-	3	-	-	1,2	-	-
e2	-	3	-	-	2,3	2,3	-
e3	-	3	-	-	2,3	3	-
t1	-	1,37	-	37	1,2	-	-
t2	-	2	-	-	2,3	-	-
t3	-	1	-	-	1,3,25	1	-
s1	1	1,13	-	1	1,2,13,37	1,2,13	-
s2	-	1,2	-	1	1,2,25,37	1,2	1
s3	1	1	-	1	1,37	1	1
r1	-	1	-	-	2	2	-
r2	-	2	-	-	2	2	-
r3	-	-	-	-	2	-	-
m1	-	37	-	-	-	37	-
m2	-	1	-	-	1	1	-
m3	-	1	-	-	1	1	1
d1	-	37	-	1	-	-	-
d2	-	-	-	-	-	-	-
d3	-	-	-	-	-	-	-
k1	-	37	-	37	-	-	1
k2	-	-	-	-	-	-	-
k3	-	-	-	-	-	-	-
l1	-	4	-	-	4	-	-
l2	-	4	-	-	4	4	-
l3	-	4	-	-	4	-	-

Table 6.4: Features with different averages for male and female speakers

Phoneme	Features	Phoneme	Features
@1	-	r1	-
@2	-	r2	-
@3	-	r3	-
n1	-	m1	-
n2	-	m2	-
n3	-	m3	-
e1	2	d1	-
e2	2,3	d2	-
e3	-	d3	-
t1	-	k1	-
t2	1,2,3	k2	-
t3	-	k3	-
s1	-	l1	-
s2	-	l2	-
s3	-	l3	-

there are little features that differ significantly between male and female speech, when comparing their averages. One of the few phonemes is the /t/ state 2. The average phonemes for this phoneme were plot in figure 6.3. Now we compare the distributions of the phoneme /t/ values for male and female speakers for the most differing feature: feature 1. This is done by making a histogram of these values and normalizing them by dividing by the number of values. This distribution is plot in figure 6.4. We see that the two distributions overlap almost completely, so the differences are very small. For the features with a smaller difference in averages, the distributions are almost identical. This suggests that the models for male and female speech should be very similar. Now we look at the distributions for the different age groups.

We see in tables 6.2 and 6.3 that for both male and female speakers, there are differences in multiple feature averages for state 1 of the phoneme /s/. This is why we will look at their specific feature distributions, expecting to find the biggest differences here. Looking at feature 1, we get figures 6.5 for the female speakers and figure 6.6 for the male speakers. It can readily be seen that the difference between these distributions is much larger than the most differing feature state distribution between male and female speakers from figure 6.4. For the second feature, in figures 6.7 and 6.8, the distributions again are almost identical. Feature 13 differs more, as can be seen in figures 6.9 and 6.10. Lastly, we compare feature 37 in figures 6.11 and 6.12. For both male and female speakers, there is a clear difference between these distributions. Overall, there seems to be sufficient difference here to make a distinction between these two age categories.

To give a typical example of a feature distribution without much difference between the averages, we add figure 6.13. It is clearly impossible to make a distinction between these categories.

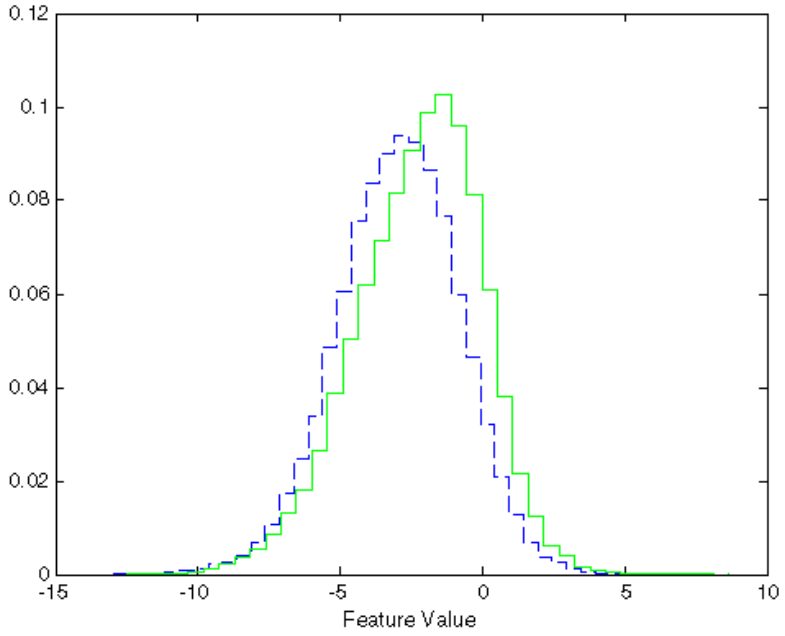

Figure 6.4: Distribution of feature 1 for /t/, state 2 for female speakers from age category 1 (solid line) and category 3 (dashed line).

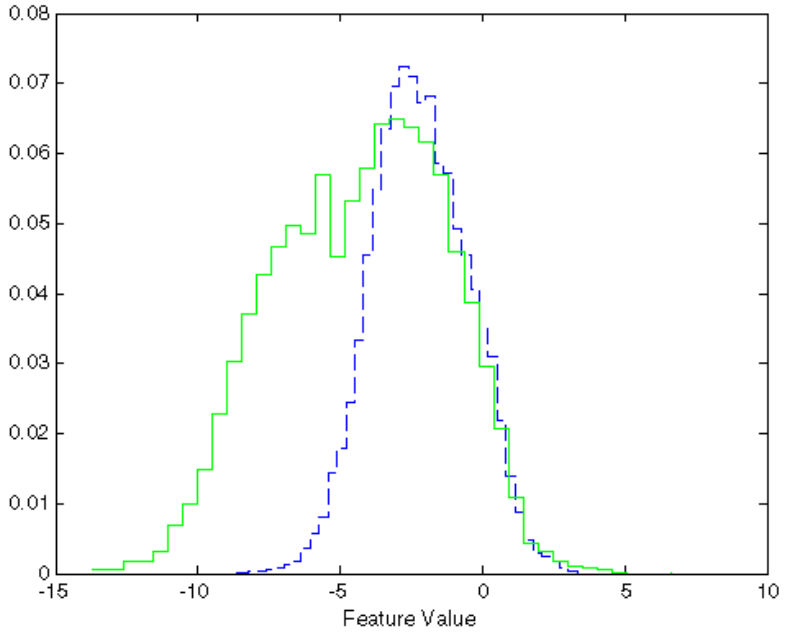

Figure 6.5: Distribution of feature 1 for /s/, state 1 for female speakers from age category 1 (solid line) and category 3 (dashed line).

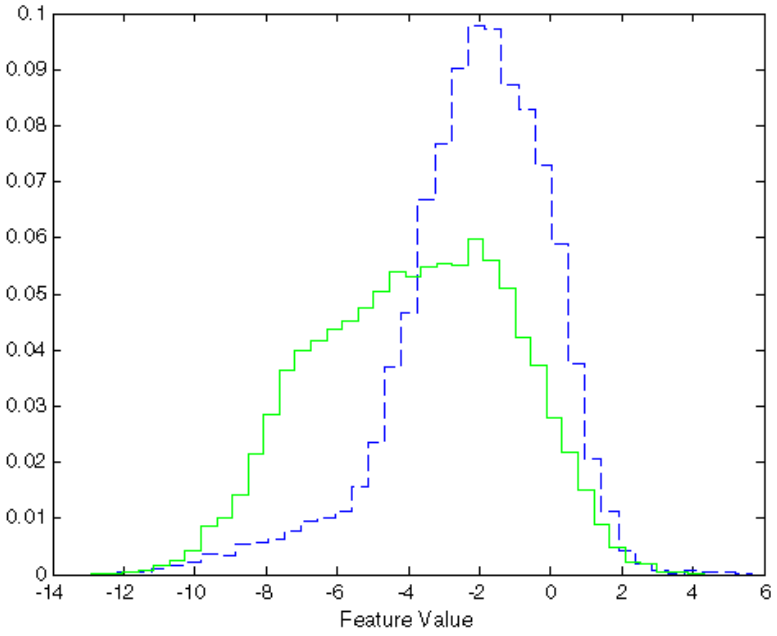

Figure 6.6: Distribution of feature 1 for /s/, state 1 for male speakers from age category 1 (solid line) and category 3 (dashed line).

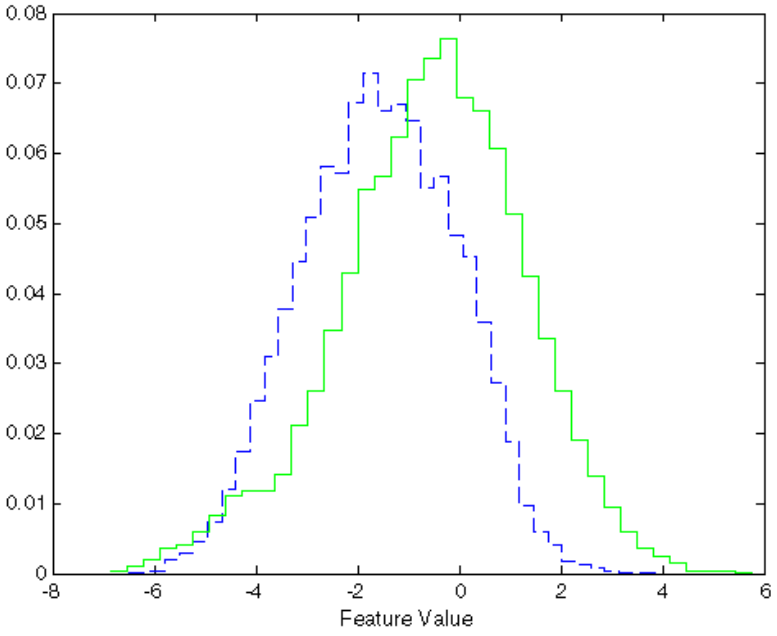

Figure 6.7: Distribution of feature 2 for /s/, state 1 for female speakers from age category 1 (solid line) and category 3 (dashed line).

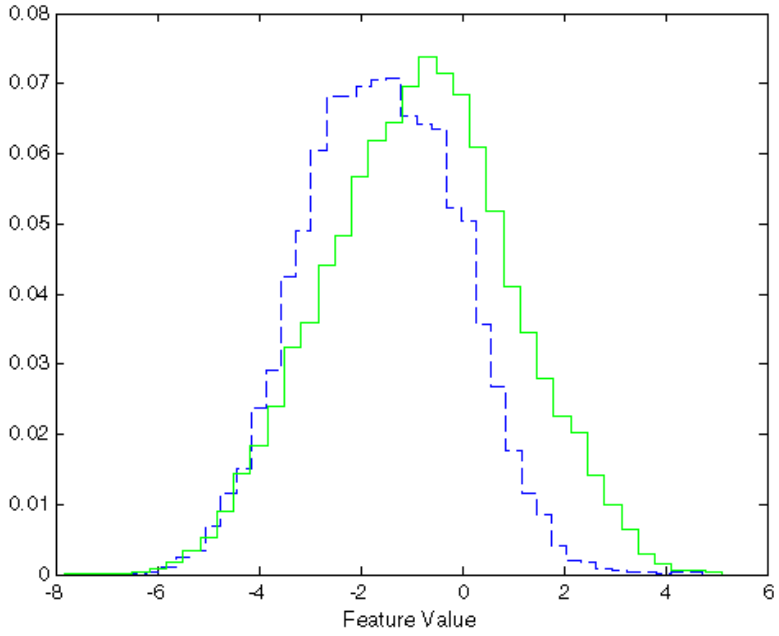

Figure 6.8: Distribution of feature 2 for /s/, state 1 for male speakers from age category 1 (solid line) and category 3 (dashed line).

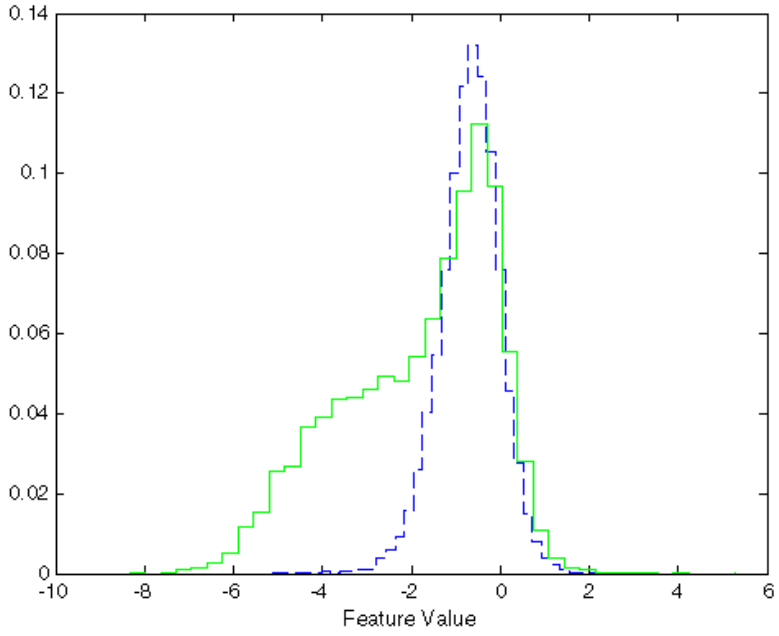

Figure 6.9: Distribution of feature 13 for /s/, state 1 for female speakers from age category 1 (solid line) and category 3 (dashed line).

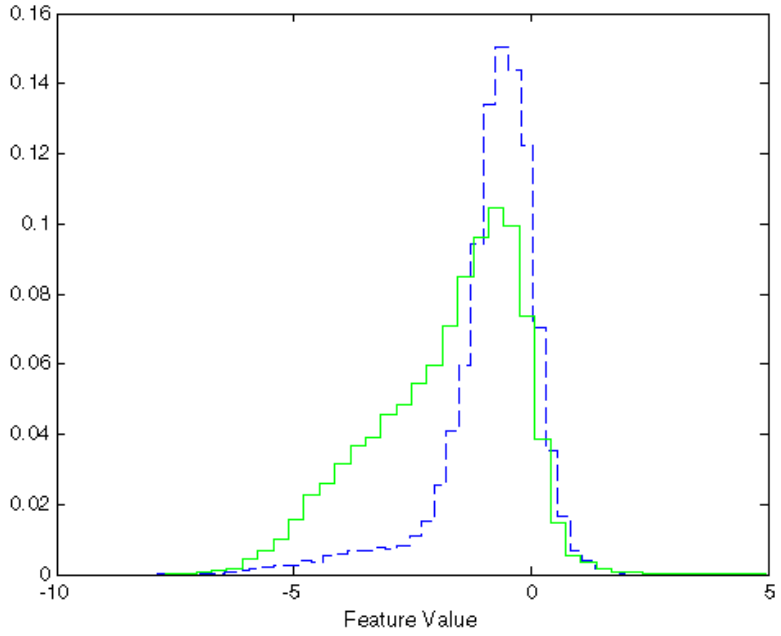

Figure 6.10: Distribution of feature 13 for /s/, state 1 for male speakers from age category 1 (solid line) and category 3 (dashed line).

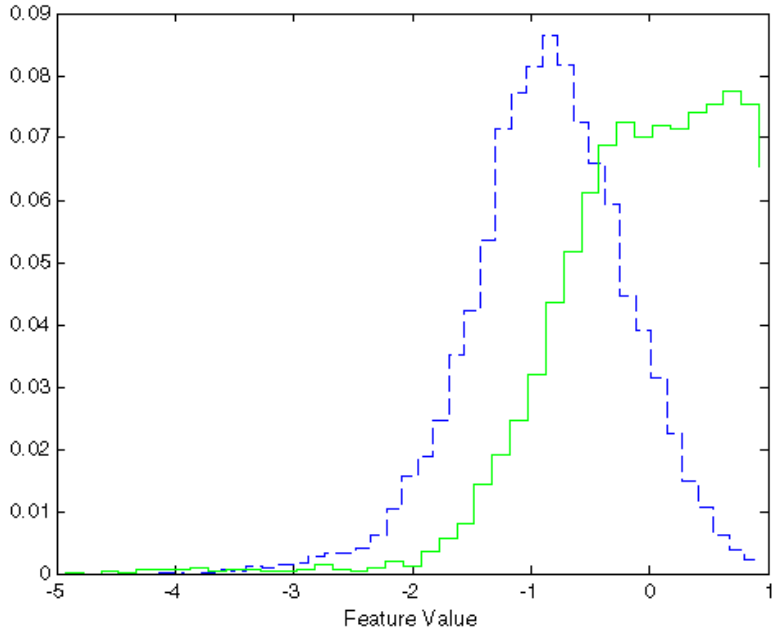

Figure 6.11: Distribution of feature 37 for /s/, state 1 for female speakers from age category 1 (solid line) and category 3 (dashed line).

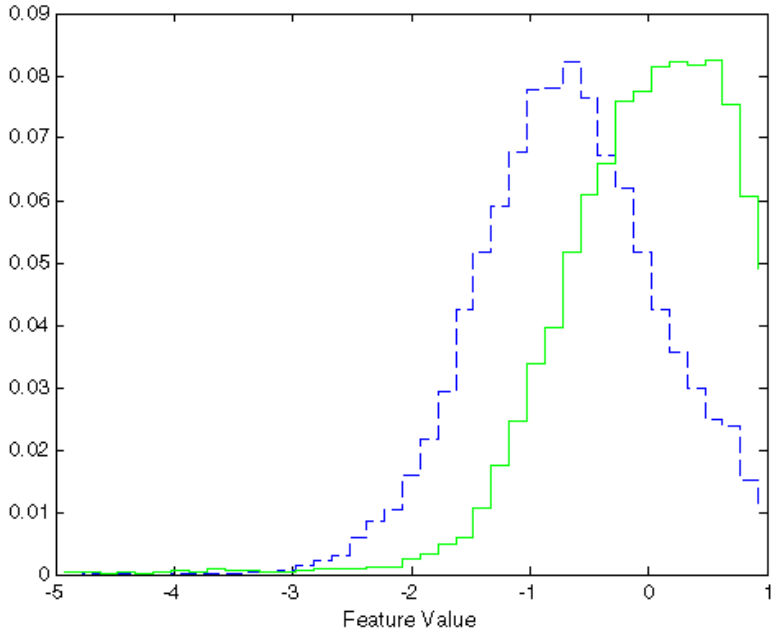

Figure 6.12: Distribution of feature 37 for /s/, state 1 for male speakers from age category 1 (solid line) and category 3 (dashed line).

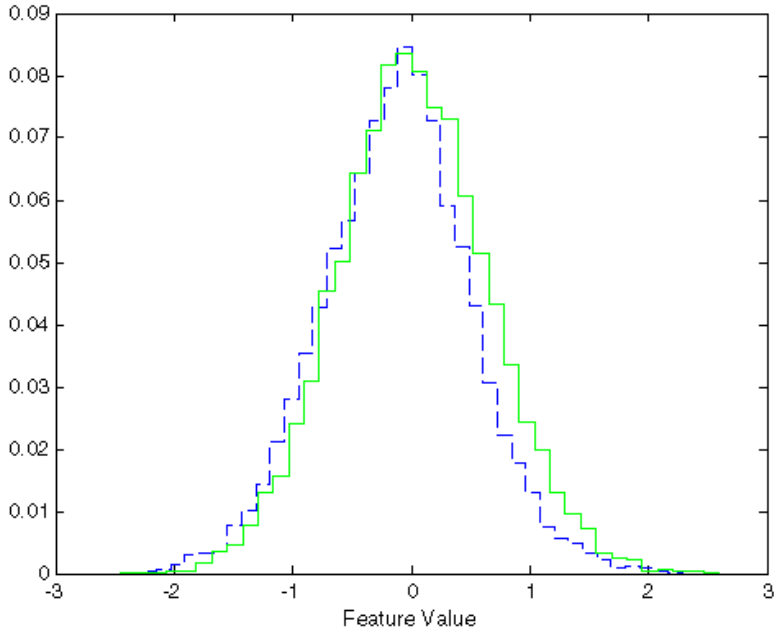

Figure 6.13: Distribution of feature 5 for /r/, state 3 for female speakers from age category 1 (solid line) and category 3 (dashed line).

6.4 Conclusions

At this moment, we would like to remind the reader that the original idea was to compare data in another feature format: MFCC features. This feature extraction process mimics the transformation that takes place in a human ear so these features should contain the same information people use to identify speaker age. Unfortunately, the Sonic manual and default features led us to using PMVDR features, developed to remove as much speaker-dependent information as possible. Since the process and time schedule was too far along to restart, we decided to continue with these features, although this reduced our chances of finding significant age differences in the speech data. All the conclusions that follow, only apply to this specific feature type.

However, analysis of the PMVDR feature data has led to some surprising conclusions. The PMVDR features succeeded quite well at removing differences between speech from different genders: the features for male and female speakers are almost identical. Relative to the differences between speech from different genders, age is a much bigger influence. It would thus make more sense to make provisions for different ages than for different genders, when using this type of acoustic feature. But not every age category can uniquely be identified: the differences are largest between age categories 1 and 3 and are almost completely absent between age categories 1 and 2, 3 and 4, 2 and 4, 3 and 5. There is a small difference between the genders here: there is some difference between women from age categories 3 and 4, 4 and 5 while the differences between speech from men of these ages has far less differences. Although there is less difference between men from age categories 2 and 3, 3 and 4, there are more differences between speech from age categories 2 and 4. This suggests there is a change in speech for female speakers in age category 3, that declines around the crossing to category 4. Differences between speech from male speakers across these categories adds up, on the other hand. This is not explained by literature from the phonetics field.

Surprisingly, the differences between the age categories are smaller for men than for women, although the literature on phonetics [Schötz, 2007] predicts otherwise. However, this statement refers also to the changes during puberty, which could not be researched here by a lack of data. Also, the PMVDR features are especially well suited for medium and high pitch speech, which could also be the cause of these differences.

Overall, it seems unlikely that with PMVDR features, using different age categories will improve the recognition rate for the Dutch language. Differences between phonemes are rare and where they occur, they are limited to a maximum of 4 features out of a set of size 39. Mostly, only one feature element differs. The only way to achieve certainty on this, is to try this out in practice, as is customary in this field of research. When this is done, provisions should be made for specific phonemes since the results of the age differences manifest themselves differently for different phonemes. It is also interesting to see whether the MFCC incorporate larger age differences for the Dutch language.

Chapter 7

Conclusions and Recommendations

> The most important thing in human communication is hearing what isn't being said
>
> *Anonymous*

In this final chapter, we reflect on the research goals and the choice for the Sonic ASR system in my literature study. Some final conclusions and recommendations for future work finish this master's thesis.

7.1 Goals

In this section, we recapitulate the goals of this research. There were two main objectives in this research project:

- to prepare a Dutch LVCSR system for the TU Delft to use as a basic system for ASR research and during the N-Best project, for the recognition of broadcast news as well as for spontaneous speech.
- to research whether the acoustic differences between Dutch speakers from different age categories, for male and female speakers separately, are sufficiently large to potentially be useful in the field of ASR.

Where the first goal is concerned, a Dutch LVCSR has been created for the Northern Dutch broadcast news task, described in chapter 5. The system performs well and achieves a good recognition rate. There was not enough time to repeat the steps in this creation using the Southern Dutch data but this can be done rather quickly using the programs I created on the Flemish data and waiting for the training steps to finish. The spontaneous speech task is also left to my successor in the N-Best project. This is because getting Sonic to work without in-house knowledge of this software and without support, took more time than expected.

The acoustic differences between Dutch speakers from different age categories, for male and female speakers separately, were investigated and it was

concluded that this does not seem promising when using PMVDR features. With MFCC features, this could still be beneficial.

7.2 Using Sonic

In my literature study [Clerx, 2007], a theoretical choice was made to use the Sonic ASR system for this project. At this point, we can reflect on how this choice has turned out.

The fast training algorithm promised Sonic's authors, delivered: only a few training passes were needed and the training passes were quick, when we obtained a more recent server. This has been very valuable during this project. The many speaker adaptation techniques have not yet been used but are still an interesting possibility that could increase the recognition rate even more.

It was the intention to use the classic and still standard MFCC features and then compare the results with the usage of PMVDR features, if there would be enough time left. Unfortunately we were lead astray on this point and started out with PMVDR features. Whether these features actually improved the recognition rate, can not be concluded without creating new speech models of the MFCC type. There was not enough time for this.

The many speech adaptation techniques in Sonic, still seem promising but have not yet been tried.

We used a trigram language model and that performed quite well. It achieved a good recognition rate and Sonic made good on the promise of speed: it performed its calculations on the BN task faster than real time, giving us the opportunity to enter the N-Best project with a contrastive condition.

7.3 Conclusions

The Sonic ASR system has been ported to the Dutch language and performs reasonable on the broadcast news task. The performance is lower than for American English but that is to be expected: the Dutch language incorporates many compounds and more verb conjugations, which makes ASR more complex. Sonic is a fast system with many interesting options but the manual really needs to be updated and improved. The learning curve is very steep and requires knowledge of programming and scripting languages.

This master's thesis also describes the practical side of speech recognition: where many researchers suffice by stating what recognition rate was obtained, this thesis also describes how that recognition was obtained in practice. This gives a good impression to new researchers of the practical side of this field.

When using PMVDR features, age differences are present but pretty limited. They are not expected to result in a much higher recognition rate. Differences between genders are even more limited though. Age differences could still improve ASR when using MFCC features.

7.4 Future Work

For the N-Best project, a Flemish version needs to be created, using the Northern Dutch ASR models as the basis, following the process described in section 5.5.

Separate models and approaches need to developed for the spontaneous speech task using conversational telephone speech. This can be done quite quickly using the programs I created on the Flemish data and waiting for the training steps to finish. Then some experimentation needs to take place, following my approach in section 5.5. The spontaneous speech task will involve adding some new pieces to the puzzle, starting with the usage of a filler file and filler penalty to compensate for the different filler words used in spontaneous speech. The basic Dutch speech system to be used as the basis, has already been prepared in sections 5.2 and 5.4. A new language model should also be created, as described in section 5.3. The language model is a crucial factor and should receive proper attention.

It would be very interesting to investigate whether age differences present themselves more in MFCC features, which incorporate more speaker-dependent information. This should also be tried in practice since that is the best test for ASR findings.

Appendix A

The N-Best Time Table

The evaluation is scheduled to take place in spring 2008. The exact dates of the evaluation are tabulated below.

1 January 2007 This is the start of the evaluation data collection process. No acoustic or text resource originating after this data may be used for training the ASR system.

6 August 2007 Start of distribution of dry-run material in evaluation-format.

31 August 2007 Deadline for submitting dry-run results.

31 March 2008 Deadline for signing up with the N-Best evaluation.

6 April 2008 Start of the evaluation. Evaluation material will be made available prior to this date, and running the evaluation can officially start after this date.

2 May 2008 Evaluation must be submitted to TNO.

9 May 2008 TNO releases the first evaluation results, reference transcripts will be distributed, adjudication period begins.

23 May 2008 End of adjudication period, TNO will fix scoring scripts and reference transcriptions.

4 August 2008 Deadline for workshop proceedings and electronic version of presentations.

11 August 2008 One-day workshop at TNO in Soesterberg for presentation of the N-Best evaluation results.

31 December 2008 End of research license of the training for sites that have not licensed the material directly from the supplier.

Appendix B

Sonic's English Decision Tree Rules

The Sonic ASR uses these splitting rules to create the decision trees.

$silence SIL br ls lg ga
$aspiration HH
$dental DH TH
$l_w L W
$s_sh S SH
$s_z_sh_zh S Z SH ZH
$affricate CH TS JH
$nasal M N NG
$schwa AX IX AXR
$voiced_fric DH Z ZH V
$voiceless_fric TH S SH F
$fricative DH TH S SH Z ZH V F
$liquid L R
$lqgl_back L R W
$liquid_glide L R W Y
$w_glide UW AW OW W
$y_glide IY AY EY OY Y
$diphthong UW AW AY EY IY OW OY
$round_vocalic UH AO UW OW OY W AXR ER
$labial W M B P V F
$palatal Y CH JH SH ZH
$alveolar N D T S Z DX TS
$alveolar_stop D T
$velar NG G K
$velar_stop G K
$labial_stop B P
$delete_stop PD TD KD BD DD GD

$oral-stop1 B D G P T K CH TS JH BD DD GD PD TD KD
$oral-stop2 P T K PD TD KD
$oral-stop3 B D G BD DD GD
$front-r AE EH IH IX IY EY AH AX Y AW
$back-r UH AO UW OW AA ER AXR OY L R W AY
$back-l UH AO UW OW AA ER AXR L R W AW
$front-l AE EH IH IX IY EY AH AX Y OY AY
$retro-flex R ER AXR
$retro-vowel ER AXR
$high-vowel IH IX IY UH UW Y
$lax-vowel EH IH IX UH AH AX
$low-vowel AE AA AO AW AY OY
$tense-vowel IY EY AE UW OW AA AO AY OY AW
$vowel AE EH IH IX IY UH AH AX AA AO UW AW AY EY OW OY ER AXR
$sonorant AE EH IH IX IY EY AH AX OY AY UH AO UW OW AA ER AXR AW L R W Y
$voiced AE EH IH IX IY UH AH AX AA AO UW AW AY EY OW OY L R W Y ER AXR M N NG JH B D DH G V Z ZH DX

Appendix C

Sonic's German Decision Tree Rules

The German rule set below is included in Sonic's porting example package. This rule set is used to build the decision tree.

```
$silence SIL br ls lg ga
$aspiration x h
$l_w l
$s_sh s S
$s_z_sh_zh s z S
$affricate ts tS Z j dZ
$nasal m n G
$schwa &
$voiced_fric z v V
$voiceless_fric s S f pf
$fricative s S z v V f pf
$liquid l r
$w_glide u U W w Y y w~ au o O O~
$y_glide i @ I ai E oi
$diphthong u U W w Y y w~ au ai E i @ I o O O~ oi
$round_vocalic u U W w Y y w~ o O O~ oi
$labial m b p v V f pf
$palatal Z j dZ S
$alveolar n d t s z ts tS
$alveolar_stop d t
$velar G g k
$velar_stop g k
$labial_stop b p
$oral-stop1 b d g p t k ts tS Z j dZ
$oral-stop2 p t k
$oral-stop3 b d g
```

$front-r e i @ I E & au
$back-r u U W w Y y w~ o O O~ a A $ a~ $ ~ oi l r ai
$back-l u U W w Y y w~ o O O~ a A $ a~ $ ~ l r au
$front-l e i @ I E & oi ai
$retro-flex r
$high-vowel i @ I u U W w Y y w~
$lax-vowel e &
$low-vowel a A $ a~ $ ~ au ai oi
$tense-vowel i @ I E u U W w Y y w~ o O O~ a A $ a~ $ ~ ai oi au
$vowel e i @ I & a A $ a~ $ ~ u U W w Y y w~ au ai E o O O~ oi
$sonorant e i @ I E & oi ai u U W w Y y w~ o O O~ a A $ a~ $ ~ au l r
$voiced e i @ I & a A $ a~ $ ~ u U W w Y y w~ au ai E o O O~ oi l r m n G Z j dZ b d g v V z

Appendix D

Paper

D.1 Keywords

Automatic Speech Recognition, Language Porting, Dutch Language, Flemish, Phonetics, Sonic, N-Best,Perceptual Minimum Variance Distortionless Response, PMVDR

D.2 Abstract

Humans are capable of estimating speaker ages by only hearing them speak. It is also well known from the field of phonetics that speaker age influences the speech signal. This has however not yet been researched for the Dutch language. In this research, the influence of age on speech is researched for both genders separately and compared with the gender differences, using Perceptual Minimum Variance Distortionless Response features. The influences of age are minimal for these features but greater than the differences between speech from different genders. Different spectral features are influenced for different phonemes. It seems unlikely that adapting speech recognizers using Perceptual Minimum Variance Distortionless Response features will lead to much improvement.

Furthermore, this thesis describes the process of creating a Dutch automated speech recognition system, using the Sonic large vocabulary continuous speech recognition system as a basis. The system achieves a recognition rate of 64.6% on the broadcast news task from the N-Best project. The porting process is described in detail and provides an accurate introduction to the practice of porting speech recognition systems.

References

Ö. Salör, B. Pellom, T. Ciloglu, K. Hacioglu, M. Demirekler. On Developing New Text and Audio Corpora and Speech Recognition Tools for the Turkish Language. In *ICSLP-2002:Inter. Conf. on Spoken Language Processing*, volume 1, pages 349–352, September 2002.

A. Morgan, N. Chen, B.Y. Zhu, Q. Stolcke. Trapping conversational speech: extending trap/tandem approaches to conversational telephone speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. (ICASSP '04).*, volume 1, pages I– 537–40, May 2004. doi: 10.1109/ICASSP.2004.1326041.

A.C.M. Rietveld, V.J. van Heuven. *algemene fonetiek*. Coutinho, 1997.

Ayako Ikeno, Bryan Pellom, Dan Cer, Ashley Thornton, Jason Brenier, Dan Jurafsky, Wayne Ward, William Byrne. Issues in Recognition of Spanish-Accented Spontaneous English. Technical report, Center for Spoken Language Research, University of Colorado at Boulder, Institute of Cognitive Science, University of Colorado at Boulder, Center for Language and Speech Research, The Johns Hopkins University, Tokyo, Japan, April 2003. in ISCA and IEEE Workshop on Spontaneous Speech Processing and Recognition.

Sabine Deligne & Frédéric Bimbot. Language modeling by variable length sequences: theoretical formulation and evaluation of multigrams. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 169–172, May 1995. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=479391`.

D. Burshtein. Robust parametric modeling of durations in hidden markov models. *IEEE Transactions on Speech and Audio Processing*, 4(3):240–242, May 1996. URL `http://ieeexplore.ieee.org/iel4/89/10806/00496221.pdf?isnumber=&arnumber=496221`.

Catia Cucchiarini, Hugo Van hamme, Felix Smits. JASMIN-CGN: Extension of the Spoken Dutch Corpus with speech of elderly people, children and non-natives in the human-machine interaction modality. In *LREC*, 2006.

P. Clarkson and T. Robinson. Towards improved language model evaluation measures. In *Proceedings of EUROSPEECH 99, 6th European Conference on Speech Communication and Technology*, volume 5, 1999. URL `citeseer.ist.psu.edu/clarkson99toward.html`.

W. Clerx. Comparison of two popular speech recognizers for use in the n-best project. Master's thesis, TU Delft, 2007.

CMU. The CMU-Cambridge Statistical Language Modeling Toolkit v2, October 2007. URL http://www.speech.cs.cmu.edu/SLM/toolkit_documentation.html.

CorpusGesprokenNederlandsDocumentation. Het Corpus Gesproken Nederlands, March 2004. Published on the annotation-DVD from the CGN.

Daniel Elenius and Mats Blomberg. Comparing speech recognition for adults and children. In *Fonetik*. Dept. of Linguistics, Stockholm University, 2004.

David van Leeuwen, Judith Kessens. Evaluation plan for the North- and South-Dutch Benchmark Evaluation of Speech recognition Technology (N-Best 2008). Technical report, TNO, Delft, Netherlands, 2006.

Entropy. A Tutorial introduction to the ideas behind Normalized cross-entropy and the information-theoretic idea of Entropy, 2004. URL http://www.nist.gov/speech/tests/rt/rt2004/fall/docs/NCE.pdf.

ESTER. Plan d'évaluation ESTER Phase I et II, 2003, 2005. URL http://www.afcp-parole.org/ester/docs.html.

Jonatan Fiscus. The 2001 nist evaluation plan for recognition of conversational speech over the telephone. Technical report, NIST, 2005. URL http://www.nist.gov/speech/tests/rt/rt2004/fall/docs/rt04f-eval-plan-v14.pdf.

Jonatan Fiscus. The rich transcription 2006 spring meeting recognition evaluation. Technical report, NIST, 2006. URL http://www.nist.gov/speech/tests/rt/rt2006/spring/docs/rt06s-meeting-eval-plan-V2.pdf.

Institut für Deutsche und Niederländische Philologie FU Berlin. De medeklinkers van het nederlands, 2004a. URL http://neon.niederlandistik.fu-berlin.de/phonology/consonants/.

Institut für Deutsche und Niederländische Philologie FU Berlin. De klinkers van het nederlands, 2004b. URL http://neon.niederlandistik.fu-berlin.de/phonology/vocals/.

G. Boulianne, J. Brousseau, P. Ouellet, P. Dumouchel. French large vocabulary recognition with cross-word phonology transducers. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings.*, 2000.

Hagai Attias, Li Deng, Alex Acero, John C. Platt. A new method for speech denoising and robust speech recognition using probabilistic models for clean speech and for noise. In *Eurospeech 2001*, 2001.

Harry Hollien. "Old voices": What do we really know about them? *Journal of Voice*, 1(1):2–17, 1987. doi: 10.1016/S0892-1997(87)80018-8.

Randall A. Helzerman and Mary P. Harper. An approach to multiply segmented constraint satisfaction problems. In *AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, pages 350–355, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence. ISBN 0-262-61102-3. URL `http://www.cs.washington.edu/research/jair/volume5/helzerman96a-html/node7.html`.

Xuedong Huang. *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, 2001.

Huijbregts, M.A.H., Ordelman, R.J.F., de Jong, F.M.G. A spoken document retrieval application in the oral history domain. In *Proceedings of the 10th international conference Speech and Computer*, pages 699–702. University of Patras, 2005.

Ian H. Witten and Timothy C. Bell. The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, July 1991.

Ayako Ikeno, Bryan Pellom, Dan Cer, Ashley Thornton, Jason M. Brenier, Dan Jurafsky, Wayne Ward, and William Byrne. Issues in recognition of spanish-accented spontaneous english. in ISCA and IEEE Workshop on Spontaneous Speech Processing and Recognition, April 2003.

Jia Li, Amir Najmi and Robert M. Gray. Image classification by a two-dimensional hidden markov model. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 48(2):517–533, February 2000.

Jason Jordan. Shntool Documentation, January 2007a.

Jason Jordan. Shntool Home Page, 2007b. URL `http://shnutils.freeshell.org/`.

Judith Kessens and David van Leeuwen. N-best: The Northern- and Southern-Dutch Benchmark Evaluation of Speech recognition Technology. In *Interspeech*, 2007.

Julius Homepage. Julius — an open-source large vocabulary csr engine, October 2006. URL `http://julius.sourceforge.jp/en/julius.html`.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, 2000.

Kunio Kashino, Takayuki Kurozumi and Hiroshi Murase. A quick search method for audio and video signals based on histogram pruning. *IEEE Transactions on Multimedia*, 5:47–58, September 2003. URL `http://ieeexplore.ieee.org/iel5/6046/27475/01223562.pdf?isnumber=&arnumber=1223562`.

Leon J.M. Rothkrantz, Pascal Wiggers, Jan-Willem A. van Wees, and Robert J. van Vark. Voice Stress Analysis. In *Text, Speech and Dialogue*, volume 3206 of *Lecture Notes in Computer Science*, pages 449–456. Delft University of Technology, Springer Berlin / Heidelberg, October 2004. doi: 10.1007/b10051110.1007/b100511.

Mei Hwang. Subphonetic acoustic modeling for speaker-independent continuous speech recognition. Technical report, Carnegie Mellon University, Pittsburgh, PA, USA, 2001.

Hy Murveit, John Butzberger, Vassilios Digalakis, and Mitch Weintraub. Progressive-search algorithms for large-vocabulary speech recognition. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pages 87–90, Morristown, NJ, USA, 1993. Association for Computational Linguistics. ISBN 1-55860-324-7.

Naveen Srinivasamurthy, Antonio Ortega and Shrikanth Narayanan. Efficient scalable encoding for distributed speech recognition, 2003. URL `http://biron.usc.edu/~snaveen/Publications.html`. Submitted to IEEE Transactions on Speech and Audio Processing.

Mukund Padmanabhan and Satya Dharanipragada. Maximizing information content in feature extraction. *IEEE Transactions on Speech and Audio Processing*, 13(4):512–519, July 2005. doi: 10.1109/TSA.2005.848876.

Pascal Wiggers, Leon J.M. Rothkrantz. Topic-based Language Modeling with Dynamic Bayesian Networks. In *Proceedings of the Ninth International Conference on Spoken Language Processing*, pages 1866–1869, September 2006.

B. Pellom. Sonic: The university of colorado continuous speech recognizer. Technical report, University of Colorado, March 2001. TR-CSLR-2001-01.

B.L. Pellom and R. Cole. The cslr international workshop, 2003. URL `http://cslr.colorado.edu/beginweb/2003summer_wkshp/`.

Bryan Pellom. Sonic: Language porting tutorial. Presentation provided with Sonic version 2.0 beta 3, 2004.

Perplexity. Perplexity, October 2007. URL `http://en.wikipedia.org/wiki/Perplexity`.

PhoneticsWiki. Phonetics. http://en.wikipedia.org/wiki/Phonetics, July 2007. URL `http://en.wikipedia.org/wiki/Phonetics`.

Prof. dr. Dirk Geeraerts, editor. *Van Dale Groot woordenboek der Nederlandse taal op cd-rom*. Van Dale Lexicografie bv Utrecht/Antwerpen, Plusversie 1.0 edition, 2000. Bevat de volledige inhoud van de driedelige Grote Van Dale(13e uitgave).

Rob Pegoraro. Speak and spell, slowly growing up. Washington Post, September 2006. URL `http://www.washingtonpost.com/wp-dyn/content/article/2006/09/09/AR2006090900091.html`.

Rogier van Dalen. Lexical Stress in Speech Recognition. Master's thesis, TU Delft, June 2005.

Susanne Schötz. *Speaker Classification I*, volume 4343, chapter Acoustic Analysis of Adult Speaker Age, pages 88–107. Springer Berlin / Heidelberg, 2007.

Sonic Homepage. SONIC: Large Vocabulary Continuous Speech Recognition System, Main project page, October 2006. URL http://cslr.colorado.edu/beginweb/speech_recognition/sonic_main.html.

SonicWeb. SONIC: Large Vocabulary Continuous Speech Recognition System, 2007. URL http://cslr.colorado.edu/beginweb/speech_recognition/sonic.html.

Mike A. Spaans. On developing acoustic models using HTK. Master's thesis, Knowledge Based Systems group at Delft University of Technology, December 2004. URL http://www.kbs.twi.tudelft.nl/Publications/MSc/2004-Spaans-MSc.html.

The Spoken Dutch Corpus Homepage. The Spoken Dutch Corpus, March 2007. URL http://www.tst.inl.nl/.

Umit H. Yapanel, John H.L.Hansen. A new perspective on feature extraction for robust in-vehicle speech recognition. In *Proceedings of Eurospeech'03*, September 2003. Geneva.

Rogier van Dalen. IN4012TU: "Real-time AI and Automated Speech Recognition": classroom example for chapter 7, May 2006. Hand-out during class at TU Delft.

J. Verhoeven and C. Van Bael. Akoestische kenmerken van de Nederlandse klinkers in drie Vlaamse regio's. *taal en tongval*, 54:1–23, 2002.

Véronique Hoste, Walter Daelemans, Steven Gillis. Using rule-induction techniques to model pronunciation variation in Dutch. *Computer Speech and Language*, 18(1):1–23, 2004.

Pascal Wiggers. *Modelling context in automatic speech recognition*. PhD thesis, TU Delft, To appear in 2008.

Inge Van Der Cruysse-Van Antwerpen William Z Shetter. *Dutch: An Essential Grammar*. Routledge, May 2002.

G. Evemnann & P.C. Woodland. Large vocabulary decoding and confidence estimation using word posterior probabilities. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, volume 3, pages 1655–1658, 2000. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=862067.

S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *HLT: Proceedings of the workshop on Human Language Technology*, pages 307–312, Morristown, NJ, USA, 1994. Association for Computational Linguistics. ISBN 1-55860-357-3.