UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

AN INVESTIGATION INTO THE FEASIBILITY OF MONITORING A CALL CENTRE USING AN EMOTION RECOGNITION SYSTEM

Werner Stoop
20178663
Submitted in partial fulfillment of the requirements for the degree
Master of Engineering (Computer-Engineering)
in the
Faculty of Engineering, the Built Environment and Information Technology
UNIVERSITY OF PRETORIA

May 30, 2010

Abstract

In this dissertation a method for the classification of emotion in speech recordings made in a customer service call centre of a large business is presented.

The problem addressed here is that customer service analysts at large businesses have to listen to large numbers of call centre recordings in order to discover customer service-related issues. Since recordings where the customer exhibits emotion are more likely to contain useful information for service improvement than "neutral" ones, being able to identify those recordings should save a lot of time for the customer service analyst.

MTN South Africa agreed to provide assistance for this project. The system that has been developed for this project can interface with MTN's call centre database, download recordings, classify them according to their emotional content, and provide feedback to the user.

The system faces the additional challenge that it is required to classify emotion notwithstanding the fact that the caller may have one of several South African accents. It should also be able to function with recordings made at telephone quality sample rates.

The project identifies several speech features that can be used to classify a speech recording according to its emotional content. The project uses these features to research the general methods by which the problem of emotion classification in speech can be approached. The project examines both a K-Nearest Neighbours Approach and an Artificial Neural Network-Based Approach to classify the emotion of the speaker.

Research is also done with regard to classifying a recording according to the gender of the speaker using a neural network approach. The reason for this classification is that the gender of a speaker may be useful input into an emotional classifier.

The project furthermore examines the problem of identifying smaller segments of speech in a recording. In the typical call centre conversation, a recording may start with the agent greeting the customer, the customer stating his or her problem, the agent performing an action, during which time no speech occurs, the agent reporting back to the user and the call being terminated. The approach taken by this project allows the program to isolate these different segments of speech in a recording and discard segments of the recording where no speech occurs.

This project suggests and implements a practical approach to the creation of a classifier in a commercial environment through its use of a scripting language interpreter that can train a classifier in one script and use the trained classifier in another script to classify unknown recordings.

The project also examines the practical issues involved in implementing an emotional classifier. It addresses the downloading of recordings from the call centre, classifying the recording and presenting the results to the customer service analyst.

Keywords: *Emotion Recognition; Call Centre Improvement; Customer Service; Gender Classification; Neural Networks; K-Nearest Neighbours; Scripting Languages; Speech Segment Isolation; Speech Feature Extraction;*

Opsomming

n Metode vir die klassifisering van emosie in spraakopnames in die oproepsentrum van 'n groot sake-onderneming word in hierdie verhandeling aangebied.

Die probleem wat hierdeur aangespreek word, is dat kliëntediens ontleders in ondernemings na groot hoeveelhede oproepsentrum opnames moet luister ten einde kliëntediens aangeleenthede te identifiseer. Aangesien opnames waarin die kliënt emosie toon, heel waarskynlik nuttige inligting bevat oor diensverbetering, behoort die vermoë om daardie opnames te identifiseer vir die analis baie tyd te spaar.

MTN Suid-Afrika het ingestem om bystand vir die projek te verleen. Die stelsel wat ontwikkel is kan opnames vanuit MTN se oproepsentrum databasis verkry, klassifiseer volgens emosionele inhoud en terugvoering aan die gebruiker verskaf.

Die stelsel moet die verdere uitdaging kan oorkom om emosie te kan klassifiseer nieteenstaande die feit dat die spreker een van verskeie Suid-Afrikaanse aksente het. Dit moet ook in staat wees om opnames wat gemaak is teen telefoon gehalte tempos te analiseer.

Die projek identifiseer verskeie spraak eienskappe wat gebruik kan word om 'n opname volgens emosionele inhoud te klassifiseer.

Die projek gebruik hierdie eienskappe om die algemene metodes waarmee die probleem van emosie klassifisering in spraak benader kan word, na te vors. Die projek gebruik 'n K-Naaste Bure en 'n Neurale Netwerk benadering om die emosie van die spreker te klassifiseer.

Navorsing is voorts gedoen met betrekking tot die klassifisering van die geslag van die spreker deur 'n neurale netwerk. Die rede vir hierdie klassifisering is dat die geslag van die spreker 'n nuttige inset vir 'n emosie klassifiseerder mag wees.

Die projek ondersoek ook die probleem van identifisering van spraakgedeeltes in 'n opname. In 'n tipiese oproepsentrum gesprek mag die opname begin met die agent wat die kliënt groet, die kliënt wat sy of haar probleem stel, die agent wat 'n aksie uitvoer sonder spraak, die agent wat terugrapporteer aan die gebruiker en die oproep wat beëindig word. Die benadering van hierdie projek laat die program toe om hierdie verskillende gedeeltes te isoleer uit die opname en om gedeeltes waar daar geen spraak plaasvind nie, uit te sny.

Die projek stel 'n praktiese benadering vir die ontwikkeling van 'n klassifiseerder in 'n kommersiële omgewing voor en implementeer dit deur gebruik te maak van 'n programeer taal interpreteerder wat 'n klassifiseerder kan oplei in een program en die opgeleide klassifiseerder gebruik om 'n onbekende opname te klassifiseer met behulp van 'n ander program.

Die projek ondersoek ook die praktiese aspekte van die implementering van 'n emosionele klassifiseerder. Dit spreek die aflaai van opnames uit die oproep sentrum, die klassifisering daarvan, en die aanbieding van die resultate aan die kliëntediens analis, aan.

Sleutel Woorde:*Emosie Herkenning; Oproep Sentrum Verbetering; Kliënte Diens; Geslag klassifisering; Neurale Netwerke; K-Naaste Bure; Programmeer Tale; Spraak Segment Isolasie; Spraak Kenmerk Ontrekking*

Glossary

AI Artificial Intelligence, 5
ANFIS Adaptive-Network-Based Fuzzy Inference System, 8
ANN Artificial Neural Network, 13
APD Augmented Prosodic Domain, 10
API Application Programming Interface, 35

CHMM Continuous Hidden Markov Model, 15
COM Component Object Model, 36
CSV Comma-Separated Values, 46

DLL Dynamic-Link Library, 36

FANN Fast Artificial Neural Network Library, 40
FCM Fuzzy C Means, 8
FIS Fuzzy Inference System, 8
FLDA Fisher's Linear Discriminant Analysis, 13

GMM Gaussian Mixture Model, 13
GPL GNU General Public License, 42
GUI Graphical User Interface, 34

HMM Hidden Markov Model, 14

IDL Interface Definition Language, 36

KNN k-Nearest Neighbours, 12

LDC Linear Discriminant Classifier, 12
LGPL Lesser GNU General Public License, 37
LPC Linear Predictive Coding, 10

MFCC Mel-frequency Cepstral Coefficient, 10
MLB Maximum Likelihood Bayes, 13
mRMR minimum Redundancy Maximum Relevance Feature Selection, 26
MSE Mean Square Error, 27
MTN Mobile Telephone Networks, 1

NN Neural Network, 13

PC Partition Coefficient, 14
PCA Principal Component Analysis, 12
PFS Promising First Selection, 12

RMS Root Mean Square, 51

SFS Sequential Forward Selector, 25
SPHMM Suprasegmental Hidden Markov Models, 15
SPiL Speech Processing in Lua, 34
SPTK Speech Signal Processing Toolkit, 40

TSK Takagi-Sugeno-Kang, 8

Contents

Chapter 1

Introduction

Large businesses invest a large amount of resources in customer service. In order to apply these resources effectively, it is imperative that they should listen to the needs of their customers. The easiest way to do this is to analyse the recordings made in their call centres, since these recordings reveal problematic aspects of the customer service of the particular business.

In this project a computer system will be developed to classify recordings made in the call centre of a business, based on the emotional content of such recordings. This will allow customer service analysts to focus on the recordings that are more likely to contain information on how the business can improve its service.

MTN South Africa agreed to provide assistance for this project. MTN is a major mobile telecommunication network operator in South Africa. The MTN Group operates one of the largest GSM networks in the world, and delivers service mainly in South Africa and other African countries.

The system that will be developed for this project will be able to interface with MTN's call centre database, download recordings, classify them according to their emotional content, and provide feedback to the user.

The system faces the additional challenge that it should be able to classify emotion notwithstanding the fact that the speaker has one of several South African accents. It should furthermore be able to work with recordings made at telephone quality sample rates. This is the topic of research done at the Meraka Institute, as described by Martirosian and Barnard [1], with which this project ties in.

1.1 Motivation

Customer service is an important function at large companies, such as MTN, which operate in competitive environments. The recordings made in the call centres of these companies provide an effective method of tracking customers' satisfaction and identifying customer service problems.

In order to leverage the information contained within these recordings, an analyst at the

company typically listens to a sampling of these recordings.

Unfortunately, only a fraction of the recordings made in these call centres provides information that can be utilised to enhance customer satisfaction. Initial investigation at MTN suggests that up to 90% of incoming calls are from customers making general queries, such as requesting the balance on their cell phone accounts. Only the remaining 10% are of interest to decision-makers within the company, since these calls provide insight into why customers are unhappy and which aspects of the services of the business can be improved.

A computerised system that can classify the recordings according to emotional content will be able to isolate this 10% of recordings in order to save time and resources and to enhance effectivity for the company involved.

1.2 Objective

The objective of this project is to develop a practical computer system capable of classifying recordings from a call centre according to their emotional content, which will enable the company to determine the efficiency of its service and identify problem areas.

The research will focus on two aspects: Finding the optimal combination of features and algorithms to achieve the best possible accuracy, and exploring ways of implementing such a system in a commercial production environment.

A successful system should have a classification accuracy comparable to that of a human, which is about 70%. This accuracy can be measured by keeping several recordings in a separate test set, classified by humans. A successful system should be able to classify the recordings in the same way as the human listeners did, for about 70% of the recordings.

The system also has the objective to classify recordings made at low sample rates, notwithstanding the fact that the speaker's first language is not English.

1.3 Research methodology

In order to develop such a system, a large sample of call centre recordings will be required that will comprise the speech corpus. MTN agreed to provide recordings for training and testing purposes, and in return a prototype of the system will be installed on their site. This will provide an opportunity to test the feasibility of the system.

Once the corpus has been obtained, it will be necessary for human listeners to classify these recordings according to emotional content. The classified corpus will be divided into a training and a test set.

Development of the system will involve extracting several features from the recordings and applying one of several algorithms. Computer learning involves adjusting the parameters within the algorithms to compensate for the difference between the output of the algorithm and the desired output.

The project builds on research done at the Meraka Institute, as described by Martirosian and Barnard [1]. Additional experiments have been done to determine the optimal combination of features and algorithms to achieve the best classification accuracy.

The accuracy of the system will be based on how well the system classifies the recordings in the test set: What percentage of recordings does the system classify the same as the human listeners did?

In terms of deliverables, the system will consist of a series of computer programs that can connect to MTN's call centre and download a number of recordings. The recordings will be analysed to extract various features and classify them according to emotional content through one of the techniques mentioned above. There will be a graphical interface through which the user can view and listen to classified recordings.

Chapter 2

Background

The topic of emotion recognition in speech signals have been receiving a significant amount of attention recently, for several reasons according to Petrushin [2]:

- Technological advancement in the recording, storing and processing of audio information.
- The development of non-intrusive sensors.
- The development of wearable computers.
- The urge to enrich human-computer interaction.
- The desire for lifelike computer agents that can express and understand emotions.

There are many motivations for the detection of emotions in speech. Dellaert, Polzin and Waibel [3] mentioned the improvement of speech-based human-machine interfaces as well as understanding how emotion is encoded in speech. For Razak, Yusof and Komiya [4] the prime importance lies in improving the naturalness and efficiency of human-machine interactions. They suggested a camera-less mobile video phone, where images of the speaker are synthesised on the listener's handset, interactive movies, as well as automatic dialogue in call centres and an intelligent communication system with talking head images. Wang and Guan [5] mentioned applications in education, entertainment and customer service. Petrushin [2]'s motivation was an application in call centres where emotion recognition can assist in the feedback and monitoring process as well as an application that can sort voice-mail messages according to the emotional content. Schuller, Rigoll and Lang [6] referred to applications in security, lie detection, games and psychiatric aid. Tato, Santos, Kompe and Pard [7]'s motivation for recognizing emotion in speech was for an application in the Sony Entertainment robot AIBO.

Polzin and Waibel [8] listed several needs to detect emotion in order to aid speech recognition: The meaning of words may be varied, special resources may need to be allocated to process the input and the output behaviour of the system may need to adapt, depending on the emotional state of the speaker.

According to them, emotion in speech has several distinctions: The attitude of the speaker towards the listener, the attitude of the speaker towards the message and the emotional state of the speaker. Their study focuses on the latter.

Psychologists have done numerous experiments and suggested several theories. In return, AI researchers contribute emotional speech synthesis, the recognition of emotions, and agents for decoding and expressing emotions (Petrushin [2]).

The overall process of developing a recognition system can be summarised in these steps:

- Obtain sample data from the domain in which the system will operate. A subset of the data samples is used to train the system and the remainder is used to test it.

- Preprocess the data and extract features. It may be necessary to do some processing of the data before it is useful for classification purposes. The extracted features reveal information about those upon which the classifier can base its decisions. Too many features may confuse the classifier, therefore it may be necessary to select the most important features first.

- Train the system using the training data. There are various techniques that can be used to classify a sample from the extracted features. There would be an error that indicates how well the samples are being classified and training involves a mathematical technique based on the actual and the desired outcome of the classifier to minimise this error.

- Test the system using the test samples. These samples are not used in training, but their desired outcome is known. Thus the performance of the classifier can be measured by comparing the output of the classifier to the desired outcome for each sample.

Therefore, the remainder of this chapter is divided into the following sections:

Data corpus describes how various researchers obtained the data they used for training and testing purposes.

Feature extraction describes how and why the researchers extracted features from their sample data. This section also describes some techniques used by the researchers to extract additional information from the sample data

Classification describes the various techniques used by the researchers to classify the sample data and the techniques used to train these systems.

Results summarises the results obtained by the various attempts.

Conclusion summarises what the various researchers achieved. It also lists the open issues suggested by some of the researchers.

2.1 Data corpus

In order to approach any pattern recognition problem, a set of sample data is required. The approaches used by researchers vary significantly.

The study by Lee, Narayanan, and Pieraccini [9] used what they referred to as the 'Wizard-of-Oz' approach: their subjects communicated with computers and classified their utterances as emotional or neutral. Their study also included samples from a call centre and actors feigning emotions.

In their later study, Lee and Narayanan obtained their data corpus from real users speaking to machine agents in a SpeechWorks call centre ([10]).

The study by Dellaert, Polzin and Waibel [3] used 50 sentences of participants in various emotional states. The studies of Cowie and Douglas-Cowie [11] and Petrushin [2] also relied on samples recorded by volunteers.

Polzin and Waibel [8] used 50 sentences divided into questions, statements and orders. These sentences consisted of two to twelve words (a mean length of 5.8 words) for a total of 291 tokens of 87 types. These sentences were recorded at 16kHz by five drama students, who portrayed them with five different emotions.

Wang and Guan [5] claimed that people convey their emotions differently and therefore a need exists for language, speaker and context independency. Their study contained 8 subjects from different backgrounds and cultures with different languages and accents who recorded 10 sentences, for a total of 720 utterances (120 for each of the emotions they classified).

Schuller, Rigoll and Lang [6] used a larger corpus with a slightly different approach. Four speakers acted out the emotions for their study. Their test set samples were recorded over a long period of time (six months per speaker) to prevent over-pronounciation and similarities. Interestingly, they also recorded samples from persons playing video games. Each of these samples was labeled according to the situation. These samples varied in length and content, ensuring greater independence.

Tato, Santos, Kompe and Pard [7]'s data corpus was based on a similar scheme: 40 commands in five emotions were recorded by 14 non-actor speakers, divided into seven male and seven female, for a total of 2800 utterances. One of the speakers' utterances were recorded twice in speaker dependent experiments. 80% of the samples were used for training and the remainder for testing. A *leaving-one-speaker-out* cross-validation algorithm was used for training and testing.

Razak, Yusof and Komiya [4] presented a database of short, everyday sentences in both the English and Malay languages. Each of these sentences was recorded using six different emotions.

2.1.1 Data mining

Once samples have been obtained, it is necessary to classify them manually in order to produce a training and a test set for the classifier.

Cowie and Douglas-Cowie [11] studied the speech of deaf people in an attempt to determine what the emotional variables might be. It is worth mentioning that the authors are from a psychology background and not a traditional pattern-recognition one. They found that for deaf people, intelligiblity is not a problem, but that through their attributions of emotion, they often evoke the wrong reactions. They suggested the following speech variables:

Judged stability Relatively slow change in the lower spectrum.

Judged poise Narrow variation in F0 and wide variation in intensity.

Judged warmth A predominance of relatively simple tunes, a tendency for change in mid-spectrum and a low level of consonant errors.

Competence Pattern of changes in the intensity contour.

Although these variabes were suggested in the context of deafness, the authors claimed that they can also be applied to emotion.

Many researchers deem it necessary to determine how well humans can classify emotions in speech. This allows them to set a benchmark against which they can compare their automated systems.

Petrushin [2] selected five data sets from their corpus where utterances were correctly classified by at least p percent by their human listeners (where p was set to 70%, 80%, 90%, 95% and 100%). Only 7.9% of their samples were correctly recognised by all their subjects, and they found that anger was the easiest emotion to portray and recognise.

In addition, Petrushin played their samples randomly to listeners to determine how well people can portray and recognise emotion and which emotions are easier or harder to recognise. They found that anger was the easiest to recognise while fear was the hardest. They also found that people tend to confuse sadness and fear, sadness and neutral, and happiness and fear. Their human listeners achieved only a 63.1% accuracy. They concluded that people understand better how to express or decode anger and sadness (based on the low variance in their results) and that the ability of humans to portray emotions is about the same as their ability to recognise it, but with a higher variance.

Razak, Yusof and Komiya [4] divided 360 samples into six sets and played them back to listeners, who decided on the intended emotion. Their human listeners achieved an overall accuracy of 62.35%, while anger was notably hard to recognise at only 45.99%.

Both Schuller, Rigoll and Lang [6] and Polzin and Waibel [8] did similar experiments. Schuller, Rigoll and Lang performed an informal experiment in which humans were asked to classify the emotions. Their human listeners achieved a 81.3% recognition rate. Polzin and Waibel found that human subjects, who listened to a speaker's utterances, were able to classify the emotions correctly with 70% accuracy.

A number of researchers also use speech recognition systems to establish baseline performances.

The study by Lee, Narayanan, and Pieraccini [9] used several samples from a computer-operated call centre. The data was mined based on several objective measures (Automatic

speech recognition accuracy, the total number of dialogue turns and the rejection rate), as well as subjective measures (Samples were played back randomly to listeners who had to classify them into one of the two emotional states used in their study).

Polzin and Waibel [8] also established a baseline for their experiments using the Janus speech recognition system. This system was 80% accurate with its own data corpus. With the neutral samples of this corpus, its accuracy dropped to 65.9% and it dropped a further 10% for all other emotions (except anger).

Since Lee and Narayanan [10]'s corpus came from real users speaking to machine agents, they had to narrow their inventory down to useful dialogues. For this, they used the automatic speech recognition accuracy, the number of dialogue turns and the rejection rate as parameters.

They then proceeded to classify their corpus into negative and non-negative emotions. They only considered two classes because of class-specific data sparsity. The dialogues were randomly played to users so that they were not influenced by the dialogue context. Most of their non-negative samples were neutral.

2.1.2 How many emotional states?

A couple of studies focus on only two emotional states: Negative and non-negative ([9], [11] and [10]). Lee, Narayanan, and Pieraccini [9]'s study focused on negative emotions since these serve as indications of how to improve service in a call centre. The features examined by Cowie and Douglas-Cowie [11] resulted in their samples to be distinguished into two emotional groups: afraid, angry and happy in the first group, and sad and neutral in the second group.

Lee and Narayanan [10] argued that emotional categories are vague (fuzzy) because of linguistic uncertainties about emotion classes. This leads to an overlap in acoustic information spaces and therefore they only examined the negative and non-negative classes. They also argued that though the ability to identify a wide variety of emotions may be attractive, it may also be unnecessary and impractical.

This was their justification for the use of a Fuzzy Inference System (FIS) based on the *Takagi-Sugeno-Kang* (TSK) method. The FIS allows for the use of overlapping class definitions while the fuzzy rules provide more insight into the classifier's structure.

Furthermore, they argued that one cannot rely on experts to determine the rules. Hence, they used the Fuzzy C Means (FCM) method, which is a fuzzy variant on the K-means method to determine their rules. They also used an Adaptive-Network-Based Fuzzy Inference System (ANFIS) to fine-tune their antecedent and consequent parameters.

Other articles study a wider variety of discrete emotions. There is no common standard for the classification of emotions, making comparisons between the various studies difficult. According to Schuller, Rigoll and Lang [6] the MPEG4 standard suggests six states: anger, disgust, fear, joy, sadness, surprise and a neutral state. Wang and Guan [5] also considered these states, claiming that all other emotions are variations on these.

Most of the remaining studies used, concentrate on five of these: happiness, sadness, anger, fear and normal (or neutral) ([2], [3], [8] and [11]).

Tato, Santos, Kompe and Pard [7] studied five emotions as well, but replaced fear with bored while Razak, Yusof and Komiya [4] decided on happiness, anger, sadness, disgust, fear and surprise.

2.1.3 How many samples are needed?

- Petrushin [2] used 700 recordings(140 per emotion). This study targeted telephone quality speech, so one hundred of their samples were sampled with 8 bits, while the remainder was sampled at 16 bits.
- Razak, Yusof and Komiya [4] used 200 samples for each emotion, giving 1200 in total. Of these, 240 samples (40 per emotion) were used for training.
- Wang and Guan [5] considered 720 samples sufficient.
- Schuller, Rigoll and Lang [6] used a larger set of 5250 samples in total. Of these, they used 100 utterances per emotion per speaker for their training set and 50 samples per emotion per speaker for their test set.
- Lee, Narayanan, and Pieraccini [9] drew attention to the problem of data sparsity as a critical challenge in the study. Their study used only 80 usable female and 62 usable male utterances from the 7200 samples they obtained from the call centre.
- Lee and Narayanan [10] used 776 female and 575 male samples. They considered male and female samples separately because of results from their previous work ([9]). They were left with 240 samples: 200 in their training and 40 samples in their test set.

2.2 Feature extraction

The first step of almost any pattern recognition system is to extract features upon which the classifier can base its decisions.

According to Tato, Santos, Kompe and Pard [7], emotions in speech can be represented in three dimensions: *Arousal* (activation), which is the degree of intensity in the speech signal; *pleasure* (evaluation), which describes whether the emotion is positive or negative, and *power*, which is the speaker's sense of control over the emotional state.

They elaborated that these distinctions in dimensions assist in emotion recognition. For example, happiness and anger lie close to each other in the arousal dimension, but opposite one another in the pleasure domain. Prosodic features gives information about the arousal dimension.

According to Razak, Yusof and Komiya [4], the prosodic features derived from the pitch, temporal, and amplitude structure contribute the most information towards recognizing emotion in speech. They argued that the fundamental frequency, F0, and the graph of F0 versus

time, the pitch contour, provide the clearest indication of the speaker's emotional state. Additional prosodic features can be obtained from the vocal energy, frequency, spectral features, formants (specifically F1 and F2), and the temporal features such as speech rate and pausing.

Almost all researchers therefore focus mainly on prosodic features that are derived from statistics of the pitch and energy signals ([9],[3],[6],[2] and [10]). The fundamental pitch signal F0 features prominently.

Dellaert, Polzin and Waibel [3] considered features derived from the rythm: The speaking rate (the inverse of the average length of the voiced part of the speech signal) and average length between voiced regions.

Lee, Narayanan, and Pieraccini [9] extracted the signal energy through a Hamming window. They suggested that the symantic, discourse and acoustic information are all valid features, though their paper only focused on acoustic information (from the pitch and energy of the signal). They used the mean, median, standard deviation, maximum and minimum of the pitch and energy of the signal.

Petrushin [2] considered the first few formant frequencies. They finally limited their features to statistical features (mean, standard deviation, minimum, maximum and range) of F0, energy, speaking rate, the first couple of formant frequencies (F1, F2, F3) and their bandwidths (BW1, BW2, BW3). The speaking rate was calculated from the inverse of the average duration of voiced parts in the speech. Another feature they used was the relative voiced energy versus the total energy in the utterance. They ended up with 43 features.

Lee and Narayanan [10] considered 21 similar features. They also did two separate experiments: one with the 15 best features, and one with the 10 best features.

Some researchers focus on some other diverse features. Some use mathematical approximations to the speech signal to derive more features:

Dellaert, Polzin and Waibel [3] approximated the pitch signals with cubic splines. These are smooth and differentiable, and therefore leads to new features: The maximum, minimum, median and standard deviation of the smoothed pitch curve, the mean minimum and mean maximum of individual voiced parts, and the mean positive derivative and negative derivatives of the individual slopes.

Petrushin [2] and Razak, Yusof and Komiya [4] also examined features from the smoothed pitch curve and its derivatives. They extracted the slope of F0 from linear regression of the voiced parts. Both examined features from the Linear Predictive Coding (LPC) parameters. Razak, Yusof and Komiya [4] used 14 LPC coefficients, which were convenient since they were used in the extraction algorithm and represented the phonetic features in speech recognition. In addition, they used pitch jitter, the speech duration and energy, and the zero-crossing rate.

Cowie and Douglas-Cowie [11]'s study relied on a system called ASSESS, which simplifies the representation of a speech signal. This system is based on landmark features in the speech signal (peaks and troughs, speech intensity, boundaries of pauses and fricative bursts). It provides quadratic functions that best fit information on the spectra.

Some features are less common.

Wang and Guan [5] considered 24 Mel-frequency Cepstral Coefficient (MFCC) and 6 Formant frequency features.

Cowie and Douglas-Cowie [11] looked at the Augmented Prosodic Domain (APD), claiming that variations in the APD are emotive, that is, information within the APD accounts for peoples' judgements based on speech.

They drew attention to the crescendo effect, where intensity rises after a few phrases, as well as the following properties of intensity signals in negative emotions: Variability in fricative bursts indicates emotional speech; the spectrum of non-fricative portions has a high midpoint when anger is present; the signal has a lower spectral midpoint for sadness; fricative bursts has high energy, a low mean, and negative slopes for anger, and a low variability and negative slopes for fear and happiness; and the distribution of pitch height along with the pitch duration are features associated with happiness.

Cowie and Douglas-Cowie [11] then discussed the prosody features in *Flattened Affect*. Flattened affect is the absence of emotional colour in speech, and is often used to diagnose schizophrenia. There are several contrasts between schizophrenics' speech and normal speech: a higher mean F0; lower pitch variability; a lower range of variation in intensity; and increased variability in the duration of silences. Schizophrenia patients have abnormal APD in their speech.

According to Schuller, Rigoll and Lang [6], most approaches to emotion recognition focus only on global statistics. They suggested looking at instantaneous features and aimed to compare the two approaches. Like many of the others, they extracted emotional information from pitch and energy contours. They extracted the pitch contours through the Average Magnitude Difference Function, which is a fast alternative to the autocorrelation function. It is also robust against noise, but susceptible to dominant formants.

Schuller, Rigoll and Lang [6] claimed that spectral characteristics depend too strongly on phonemes, which is a drawback since the phonemes are in turn dependant on the spoken content and language.

Tato, Santos, Kompe and Pard [7] aimed for a language independent classifier. Therefore they avoided contextual and linguistic information. Since they divided emotional space into three dimensions, they treated prosodic and quality features as different feature subspaces.

For the prosodic subspace, they identified 37 features, of which they used 26. They used 11 additional features including model jitter, tremor and pitch derivative statistics.

They listed articulatory precision, vocal tract properties (for example the formant structure), and phonatory quality parameters (the auditory qualities derived from variations in the glottal excitation) as sources of features from the quality subspace. They identified 16 quality features including the first three formants and their bandwidths, the harmonic to noise ratio, the spectral energy distribution, the voiced to unvoiced energy ratio, and the glottal flow.

Several researchers ([3], [4],[6] and [9]) normalise their input samples. This is a standard procedure for several classification techniques, which ensures that certain features are not regarded as too important or unimportant by the classifier.

Razak, Yusof and Komiya [4] removed the DC component from the signal after which they split the signal into 25ms segments (with 5ms overlap) and passed it through a zero-phase filter to remove low frequency drift. Afterwards, they used a 13th order *Linear Predictive Analyser* to extract their LPC coefficients. The coefficients were calculated using the orthogonal covariance

method.

Their speech signals were divided into segments to assist in the extraction of features. Each segment was classified as either voiced or unvoiced using the prediction error by setting a threshold. The first voiced segment was taken as the beginning of the speech period, while a sequence of unvoiced segments was taken as the end of the frame. The pitch period of each frame was calculated from the cepstrum of the voiced prediction error signal. The formants were calculated from the LPC coefficients: they are the roots of the LP polynomial. The zero-crossing rate was counted as the number of sign changes in the frame and was used to estimate the frequency.

This left them with 22 features for each of the frames in the speech period. These were averaged over each feature over all the segments within the whole voice sample.

Lee, Narayanan, and Pieraccini [9] used Principal Component Analysis (PCA) for dimensionality reduction, since many of the extracted features are redundant.

Both Lee, Narayanan, and Pieraccini [9] and Dellaert, Polzin and Waibel [3] used Promising First Selection (PFS) using the k-Nearest Neighbours method and Forward Selection (FS) to select the features to use in their study. These techniques alleviate the curse of dimensionality to some extent.

From their 43 features, Petrushin [2] used the RELIEF-F algorithm for feature selection. They ordered the feature sets according to importance and selected three feature sets with 8, 10 and 14 features respectively.

Tato, Santos, Kompe and Pard [7] also used feature selection algorithms. Their reason for this was that their data corpus was not large enough. They used linear regression models as well as quadratic regression models in some cases. They also duplicated their database of neutral samples to balance the input patterns.

Wang and Guan [5] implemented noise reduction on their samples through thresholding the wavelet coefficients. They removed leading and trailing silence (where the amplitude of the signal is below a certain threshold). They used the stepwise method from a system called SPSS which maximises the Mahalanobis distance between classes.

Lee and Narayanan [10] down-sampled their data set. Their data set contained more non-negative samples than negative samples, but no prior information is known about the distribution. Therefore their new data set contained an equal amount of negative and non-negative samples.

2.3 Classification

Several of the pattern recognition techniques developed through the years are deemed appropriate by researchers for their study of emotion recognition.

2.3.1 Statistical methods

Many researchers rely on traditional statistical methods for their classifiers.

Lee, Narayanan, and Pieraccini [9]'s study used two pattern recognition techniques: Linear Discriminant Classifier (LDC) and a k-Nearest Neighbour classifier. The LDC method assumes a Gaussian distribution in the input features.

Dellaert, Polzin and Waibel [3] used several standard pattern recognition techniques to determine a base performance. These were a Maximum Likelihood Bayes (MLB) classifier, a Kernel Regression classifier as well as a k-Nearest Neighbours classifier. The best performance was obtained from the k-Nearest Neighbours classifier.

They proceeded to determine methods for increasing the base performance of their classifier. They attempted to find a distance metric for their KNN classifier through Population Hillclimbing (which is a variation on evolutionary strategies). This increased the performance of their classifier, but it is a very expensive operation. Their reasoning was that the performance landscape is very rugged and traditional optimization techniques, such as gradient descent, are likely to fail.

Their study then proceeded to determine additional methods for improving the performance of their KNN classifier. Their reasoning was that decreasing the number of features removes information content from the data while increasing it hampers KNN's ability to use the available data. They compromised through a technique called Majority Voting of Subspace Specialists, in which several classifiers (subspace specialists) examine subsets of the features and cast a vote on how to classify the sample. The majority vote yields the final classification. Specialists are chosen so that they perform well in combination through the techniques of Selective Composition (which is similar to PFS), and Cooperative Composition (which is similar to FS).

Wang and Guan [5] used several different algorithms: A MLB classifier; a Gaussian Mixture Model (GMM); a Neural Network classifier; a KNN classifier; and a Fisher's Linear Discriminant Analysis (FLDA) classifier. The parameters for their GMM classifier were estimated through the Expectation Maximization algorithm.

2.3.2 Neural network-based methods

Neural networks seem to be a natural candidate for use in the classification of emotions.

Wang and Guan [5]'s Neural Network was a three layer feed forward network trained through the back propagation algorithm. It had 6 output neurons, one for each emotion, and 10 hidden layer neurons.

Petrushin [2] created a KNN and a two layer back propagation NN classifier. They used three different sets of features (8, 10 and 14 features) in each classifier and compared the performances.

Furthermore, Petrushin [2] used ensembles of NN classifiers. These consisted of an odd number of NN classifiers trained on different subsets of the training set. A decision is made through majority voting. Their ensembles consisted of a number of 7 to 15 networks.

Petrushin also experimented with a *Set of Experts* which consisted of a set of neural networks, each trained to recognise only one emotion. These experts reached an average accuracy of 70% per emotion, but the accuracy dropped to only 60% when the emotion associated with

the expert whose output was closest to one was selected. They attempted to improve this situation by feeding each expert's output into another NN.

Their classification system determined the emotional state as follows: First, the utterance was recognised through a emotion independent recognition system. Then an emotion dependent recognition system was used to determine the highest probability that the sentence recognised in the first step was produced by the model i according to

$$P(speechsignal|sentence, emotiondependantmodel_i)$$

The model with the highest probability was chosen.

The classifier used by Tato, Santos, Kompe and Pard [7] looked at two of the three emotional dimensions: arousal and pleasure. They used two separate classifiers, one for each dimension. The final decision was made based on the results of both classifiers.

In their NN-based approach, classification was based on the highest output node. They used a magnitude pruning algorithm which makes the networks smaller by removing the node with the smallest weight. Their neural networks had two layers. They used the *R-prop* algorithm, which trains the network based on the behaviour of the error function.

They started by doing a speaker dependent experiment to assess the confusion between emotional states. In the arousal dimension, they used all 37 prosodic features and found that happy and angry, and sad and bored, are close to each other in the arousal dimension. In the pleasure dimension they found that happy and angry are removed from each other, while sad and bored remained close together.

2.3.3 Fuzzy methods

Because of Lee and Narayanan [10]'s reasoning that emotional categories are vague, it makes sense to use fuzzy inference systems to classify emotions in speech.

They used an ANFIS system. Their rules used product composition for *and* operators in rules, and *min* inference. They used the FCM algorithm used for fuzzy-clustering to generate their initial fuzzy rules' antecedents. They encountered two difficulties: How to set the FCM cluster centres, and how to determine the optimal number of clusters. The former was solved by randomly placing the centres while the latter was solved through maximizing a *partition coefficient* (PC). They generated their rules' consequents through Least Squares estimation and by keeping them constant (implying a zero'th order TSK).

They fine-tuned their rules in Matlab through ANFIS (a neuro-fuzzy approach). The antecedents through gradient descent (backpropagation) then adjusted the consequent parameters through the Least Squares method.

They did two experiments: One with two fuzzy rules (FIS-2) and one with four rules (FIS-4).

Razak, Yusof and Komiya [4] also used a fuzzy modelling technique. The means and variances for all features were taken over all training samples, separated into the six emotions. A membership function was used to calculate the likelihood that a sample was a member of a given emotion, and the emotion for which this function was the highest was the emotion at which the sample was classified.

2.3.4 Hidden Markov model-based methods

Hidden Markov models are closely associated with speech recognition. Therefore it seems natural to apply them to recognizing emotion in speech.

Schuller, Rigoll and Lang [6] used two approaches namely a single state HMM (which is a GMM) and a Continuous Hidden Markov Model (CHMM) in order to compare the performance of approaches that focus on global statistics against approaches that use instantaneous features. Their GMM approach used 4 mixtures, each modelled by one GMM, with the choice of emotion made through maximum likelihood.

Their CHMM approach increased the temporal complexity and took the sort time behaviour of speech into account through direct analysis of low level contours. Unvoiced sounds were eliminated. This caused a loss of temporal information, but improved the independency of content. They filtered noise through a low pass Symmetrical Moving Average filter and normalised their contours. This yielded a six dimensional feature vector for each frame. Their CHMM was trained through Baum-Welch re-estimation. This finally resulted in one model for each of the seven emotions. These were left-right models modelling the advances in time.

The study by Polzin and Waibel [8] used *Suprasegmental Hidden Markov Models* (SPHMM) which summarise several states within a HMM into a suprasegmental state. This allows observations at rates appropriate to the phenomena they are intended to model.

In their classifier they modelled accoustic events through conventional HMMs and prosodic events (at phone, syllable, word and utterance levels) through suprasegmental states. The suprasegmental states allow observations at time scales appropriate for prosodic phenomena. These states provide information about the duration of a segment, the fundamental frequency and the intensity. This information becomes more meaningful if it can be observed over the duration of a syllable or a word. Additional observations, such as mean and variance, can also be derived from these states.

To be practical, however, SPHMM's must be computationally efficient and robust with respect to noise and idiosyncrasies of speakers.

SPHMM training is similar to HMM training with the added suprasegmental models on top of the acoustic models. Polzin and Waibel used four models (for happiness, sadness, anger and fear) per word. 70% of their corpus was used for training.

2.4 Results

Due to these problems it is difficult to compare the results and identify a baseline: The fact that there is no agreement on which emotions should be classified causes some researchers to aim for context, gender and language independency while others do not, and the manner in which their results are presented differs significantly.

Lee, Narayanan, and Pieraccini [9]'s study showed a significant gender dependency in the performance of the classifier. Their KNN classifier performed more consistently (though not necessarily better) than their LDC. They argued that the Gaussian distribution assumption of

the LDC method may not be valid. In their study the PFS and FS methods of dimensionality reduction performed better than PCA.

Dellaert, Polzin and Waibel [3]'s MLB classifier also performed poorly. They agreed with Lee, Narayanan, and Pieraccini [9] that the assumption of Gaussian densities in the features may be invalid. Their Majority Voting of Subspace Specialists classifier, however, yielded a definite and large performance benefit.

Of all Wang and Guan [5]'s classifiers, their FLDA classifier achieved the best accuracy at 67%. It used 21 features as selected through their stepwise method. They found that the addition of MFCC and Formant features improved the performance of their KNN and FLDA classifiers but hampered their MLB, GMM, and Neural Network classifiers. They also confirmed that prosodic features are important.

Schuller, Rigoll and Lang [6] trained and tested both of their classifiers with the same samples. Their global statistics approach, using the GMM classifier, achieved a 86.8% recognition rate. Their CHMM classifier (with 64 states and 4 mixtures) achieved a recognition rate of 77.8%. They also reported a general increase in performance when using more states. It should be mentioned that their results were reported to be notably dependent on the speaker. When using different speakers, they experienced a drop of 20% in recognition rate.

Petrushin [2] achieved the following results: Their KNN classifier achieved an average accuracy of 55%. Their NN classifier achieved its best accuracy of 65% with the set of 8 features. The ensembles of NN classifiers approach achieved a 70% accuracy while their set of experts approach achieved only 63% accuracy at best.

Using just acoustic models, Polzin and Waibel [8] reported an overall accuracy of 65.4% with anger recognised with 83% accuracy, fear recognised with 73% accuracy, happiness recognised with 60% accuracy, and sadness recognised with 46% accuracy.

Using their suprasegmental models, they improved performance to 72.25% overall. They concluded that the extra prosodic information appeared to help the detection of happiness and sadness. In this experiment anger was recognised with 77.9% accuracy, fear recognised with 60% accuracy, happiness recognised with 93.8% accuracy, and sadness recognised with 59.6% accuracy.

Tato, Santos, Kompe and Pard [7]'s accuracy was slightly better than 60%. Their experiment also aimed to eliminate context and language dependencies.

Razak, Yusof and Komiya [4]'s fuzzy inference system achieved an overall accuracy of 68.59%, which compares favourably with the results of their experiment with human listeners (62.35%). The variance in their results should be noted: their system classified happiness and disgust the most accurately (91.53% and 80% respectively), yet it classified anger with only 18.33% accuracy. The humans in their experiment also found anger to be the hardest emotion to classify.

2.5 Conclusion

Dellaert, Polzin and Waibel [3] concluded that their smoothing spline approximation to the speech curve contained enough information to classify emotional content in a speech signal.

Later, Polzin and Waibel [8] concluded that the combination of acoustic and prosodic information with a HMM approach yields an improvement in performance (7% in their experiments).

Cowie and Douglas-Cowie [11] claimed that variations in the APD are emotive, since it is a departure from a well-controlled emotional state. They also drew attention to the fact that it is sometimes difficult for human listeners to classify emotions and that there is a difficulty to classify emotion through phonetics.

Wang and Guan [5] concluded that language, accent and context independent classification is indeed possible. Petrushin [2] found that the emotions detected were affected by the cultural, social and intellectual characteristics of their subjects.

Petrushin [2] further concluded that NN classifiers are useful for detecting emotions in speech. Lee and Narayanan [10] concluded that FIS improved performance over a LDC and a KNN classifier. Fewer input features lead to improved performance. Each of their two FIS rules corresponded to one of the two emotional classes. They concluded that FIS is a promising tool in emotion recognition.

Schuller, Rigoll and Lang [6] found that some emotions are easily confused with others, while others are more easily recognised. They noted that their test subjects may have had problems feigning certain emotions. They found little correlation between the results of their GMM and CHMM approaches.

Razak, Yusof and Komiya [4] hypothesised that the inability of their system to classify anger may have been due to insufficient samples. They reasoned that anger has many variations (such as hot anger and cold anger) which leads to many variations in the accoustic features. Both humans and computers tended to confuse happiness, sadness and anger with fear. They concluded that humans vary their recognition technique according to the context of the situation, while computers rely on good reference data to base their decisions.

Tato, Santos, Kompe and Pard [7]'s division of the emotional space led them to conclude that the arousal dimension can be analysed using prosody features, but some emotions (for example, happy and angry) are close together on this axis. Therefore the pleasure dimension can be analysed through quality features to make the final decision. They found that real neutral emotions are very rare in speech.

Petrushin [2] contradicted this by concluding that anger is the easiest emotion to recognise. According to them, anger is also the most important emotion for business applications. They also noted that there are certain variants on the accoustic features that depend on the specific type of anger.

Most researchers are pleased with their results, but some of them are disappointed by their data corpus.

Razak, Yusof and Komiya [4] argued that an accuracy of 68% is sufficient compared to the rate achieved by humans, but that the large difference in recognition rates could be rectified through better sample data in future studies. Their test samples also contained sentences

identical to samples in their training database. This will require further study for random everyday speech.

Lee, Narayanan, and Pieraccini [9] mentioned that most research focuses on features extracted from speech rather than the speech itself. Lee and Narayanan later argued that language and discourse information should also be investigated (see [10]).

Dellaert, Polzin and Waibel [3]'s results were speaker dependent and their emotions were portrayed by actors. They plan to determine which features are speaker dependent in a future study.

Polzin and Waibel later suggested that a speaker's choice of words may indicate the emotional state (refer to [8]). They suggested adding emotion dependent language models to the system. They also suggested adding visual data to which SPHMM's are well-suited.

Several researchers speculate that better classification techniques may yield better results.

These researchers question the appropriateness of the labels attached to the emotions, and suggest looking at unsupervised or semi-supervised classification techniques.

Schuller, Rigoll and Lang [6] suggested that combining their HMM approach with Neural Networks may yield better results.

A last important open issue is the practical implementation of emotion recognition systems. Petrushin [2] used their research to implement a practical emotion recognition system, which prioritises telephone quality voice in call centres. Since anger is an important emotion for business purposes, their recogniser was designed to distinguish between agitation (anger, happiness and fear) and calm (neutral and sadness). They used a corpus of 56 telephone messages from 18 speakers that were split into 1 to 3 second segments. They used ensembles of 15 neural networks with 10 to 20 hidden nodes, using the 8, 10 and 14 feature feature sets. They achieved the best accuracy of 77% using 8 features and 10 hidden nodes. The structure of their system reads recorded files every few seconds, and analyses the emotions so that the recordings can be assigned to agents who return the calls.

Chapter 3

Classification

This chapter describes the research that has been done to achieve the aims of this project. Our approach had these steps:

- Obtain a data corpus from the call centre.
- Have human listener listen to each of these recordings and classify them according to their emotional content.
- Extract features that may indicate the emotion of the speaker from these recordings.
- Develop the classifier. This stage consisted of experimentation with both k-Nearest Neighbours and neural network-based approaches in order to determine the optimal method.

3.1 Data corpus

The data corpus used for this project was shared by the research done by Martirosian and Barnard [1].

It was obtained from MTN's call centre. Non-English samples were ignored because human listeners would have to classify each of these recordings.

Each of the recordings were subdivided into samples of between 2 and 20 seconds in duration. There were 2745 samples in total. Human listeners listened to each of these samples and classified them.

The original labels were based on the labels chosen by other researchers, as discussed in section 2.1.2 on page 8.

The decision was also made to supplement these with some new emotional labels where necessary. It was considered more conducive to have too many emotional labels at this stage than too few, because it would be easier to combine labels than it would be to examine already classified samples and reclassify them.

The initial choice of emotional labels were:

Neutral samples are samples that do not contain any emotional content. These are typically recordings where the caller made general enquiries, and are the most common. Any sample that cannot be classified as containing emotional content is classified as neutral.

Happy samples are recordings where the speaker is satisfied because of good customer service.

Friendly samples are samples where the speaker is polite towards the call centre agent, but not necessarily because of good service.

Angry are samples where the caller is clearly expressing anger at either poor service, or the situation which prompted him/her to phone the MTN call centre.

Frustrated are samples where the speaker is clearly dissatisfied with his/her situation, but is not angry

Impatient are samples where the speaker expresses impatience

Fear is used for samples where the speaker is concerned for something that may happen (such as an account being barred) or have already happened (such as a broken cell phone).

Sad are samples where the speaker is clearly unhappy or disappointed with the service or another aspect of the business.

Depressed are samples where the speaker is disappointed in the customer service, but does not necessarily express anger.

When the human listeners were done with the classification a total of 723 (26.34%) were classified as containing emotional content, divided as follows:

- 57 (2.08%) were classified as happy.
- 34 (1.24%) were classified as friendly.
- 70 (2.55%) were classified as angry.
- 68 (2.48%) were classified as frustrated.
- 190 (6.92%) were classified as impatient.
- 201 (7.32%) were classified as fear.
- 99 (3.61%) were classified as sad.
- 4 (0.15%) were classified as depressed.

There are three major problems with this classification of samples: Firstly, there was a concern that there were not sufficient samples to be able to train a classifier properly. Secondly, the emotional labels were very subjective. Different human listeners may classify the same sample very differently. Thirdly, a human listener may classify the recordings according to the context: *what* is being said, not *how* it is being said, while a computer classifier will only be able to infer the emotion from features in the speech samples (which relate to *how* it is being said).

In order to alleviate these problems, the original classified samples were regrouped as follows:

- *happy* and *friendly* were combined into a broader *happy* category containing 91 samples (3.32% of the total).
- *angry*, *frustrated* and *impatient* were combined into a broader *angry* category containing 328 samples (11.95% of the total).
- The *fear* category remained unchanged, with 201 samples (7.32% of the total).
- *sad* and *depressed* were combined into a broader *sad* category containing 103 samples (3.75% of the total).

This new classification also aligns more closely to what other researchers have found, as described in section 2.1.2.

There was still a concern over whether there were enough samples to train a classifier adequately. In order to ease this concern and to tie in more closely with the research done by Martirosian and Barnard [1], the *happy*, *angry* and *fear* categories were grouped together as the *active* grouping that contains 620 samples (22.59% of the total), while *sad* made up the *passive* grouping with 103 samples (3.75% of the total). The remaining samples were still classified as *neutral*.

Because there are so many more neutral samples than samples from the other classes, a classifier may become biased towards neutral samples, and may even learn to classify everything as neutral. When designing the classifier, it would therefore be necessary to limit the number of training samples to the class with the least samples. That is, because we only have 103 passive samples, we are limited to using only 103 active and 103 neutral samples in order to eliminate any bias.

Unfortunately, this limits us to using only 11.25% of the available samples.

3.2 Feature extraction

In order to classify the recordings according to emotion, it is necessary to extract features from the audio recordings. The classifier can then make a decision on the emotion of the sample based on the properties of these features, especially in terms of how they compare to similar features of other samples in the same emotional class.

The SPiL program described in section 4.4 was used to extract the speech features for this project, and it in turn was based on source code from the open source PRAAT program ([12]), and is described in detail in section 4.4.2.

Ververidis and Kotropoulos [13] provide a thorough overview of the features typically extracted for research in emotion recognition. Most features relate to the pitch, vocal tract and energy of the waveform.

Pitch is also known as the glottal waveform, and can provide information about the emotional content because it depends on the tension of the vocal folds and subglottal air pressure. Pitch is produced by vibration of the vocal folds.

Typically, two features derived from pitch are widely used: The *pitch period* T, which measures the time elapsed between two successive vocal fold openings, and the *pitch frequency* F_0, which is the vibration rate of the vocal folds, and is also known as the *fundamental frequency of phonation*.

Several algorithms exist for pitch estimation. These algorithms are based on, amongst others, autocorrelation and wavelet transforms of the waveform. PRAAT uses acoustic periocity detection based on an accurate autocorrelation method to extract the pitch from a waveform ([12]).

The features that relate to the vocal tract are the formants, the cross section area when the vocal tract is modelled as a series of lossless tubes and the coefficients derived from frequency transformations.

The formants are a representation of the vocal tract resonances, which in turn depend on the shape and physical dimensions of the vocal tracts. They "form" the overall spectrum, and hence the term formant. Formants are typically estimated through linear prediction analysis.

Formants are characterised by their centre frequencies and bandwidths. It has been found that subjects under stress do not articulate voiced sounds with as much effort, and therefore the formants can be an indication of the emotional content.

Approximating a vocal tract as a series of tubes provides a realistic model of the vocal tract for a large number of tubes, but it has the drawback that it cannot be modelled in realtime.

The coefficients derived from frequency transformations relate to the energy of certain frequency bands. Unfortunately, researchers do not agree on which bands. The Mel-frequency Cepstral Coefficients (MFCC) provides a better representation of the signal, but experimental results show that MFCC achieve poor emotional classification results in practice because pitch information is filtered. Log-frequency Power Coefficients (LFPC) has been suggested as better features because they do not discard pitch information.

The speech energy relates to the arousal level of emotions, and can therefore also be a valuable feature for emotional classification.

Ververidis and Kotropoulos [13] stated that the contours of the short-term features are affected by emotion because they describe the temporal characteristics of emotion.

They suggested several statistics to identify emotion in the speech signal: The mean, range and variance of the pitch contour trends, the mean and range of the intensity contour and the rate of speech and transmission between utterances. The speech rate is the inverse of the duration of the voiced parts of speech, as determined by the presence of pitch pulses.

The same features extracted by Martirosian and Barnard [1] were extracted from the data corpus. The extracted features are

- The largest and second largest intensity standard deviation measured over 300ms intervals.
- The largest and second largest pitch standard deviation measured over 300ms intervals.
- The mean, standard deviation, maximum and minimum of the first, second and third formants.
- The mean, standard deviation, maximum and minimum of the pitch.
- The mean, standard deviation, maximum and minimum of the intensity.
- The largest and second largest pitch gradient.
- The largest and second largest intensity gradient.
- The number of peaks in the intensity curve.
- The largest and second largest peak to peak values of the pitch curve.
- The largest and second largest peak to peak values of the intensity curve.
- The largest and second longest times between peaks in the pitch curve.
- The largest and second longest times between peaks in the intensity curve.
- The highest and second highest second derivatives (acceleration) in the pitch curve.
- The highest and second highest second derivatives (acceleration) in the intensity curve.
- The mean pitch standard deviation as measured over 300ms intervals.
- The mean intensity standard deviation as measured over 300ms intervals.
- The mean intensity gradient.
- The mean pitch gradient.

In addition, the following features were also extracted:

- The root mean square value of the sound samples.
- The sound power and energy.
- The sound power and energy in the air.

- The sound intensity in dB.
- The number of voiced frames in the pitch divided by the duration of the sample. This gives an indication of how fast the speaker is talking, which correlates to the speech rate mentioned in Ververidis and Kotropoulos [13].
- The overall pitch range (The difference between the maximum value of the pitch and the minimum).
- The overall intensity range.
- The overall range of the first, second and third formants.

According to Boersma and Weenink[12], if the unit of sound amplitude is Pa (Pascal) then the unit of the power will be Pa^2 and the unit of the energy will be $Pa^2 \cdot s$. The *power in air* and *energy in air* takes air density and the velocity of sound in air into account, and is measured in $Watt/m^2$ and $Joule/m^2$, respectively. The *in air* energy and power features are proportional to the air-independent features.

Although the ranges are derived from other features, their values are useful for the k-Nearest neighbours classifier described in section 3.3.2.

In all, there were 57 features.

3.2.1 Pre- and additional processing

As per Bishop [14], the data was normalised using equation (3.1)

$$\tilde{x}_i^n = \frac{x_i^n - \overline{x}_i}{\sigma_i} \tag{3.1}$$

where x_i^n is the value of the ith feature for the nth sample. $\tilde{x}_i^n$ is the new, scaled value for x_i^n.

$\overline{x}_i$ and σ_i are the mean and standard deviation of the ith feature, respectively, as given by equations (3.2) and (3.3).

$$\overline{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_i^n \tag{3.2}$$

$$\sigma_i^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_i^n - \overline{x}_i)^2 \tag{3.3}$$

Scaling the features results in all features having a mean of zero and a standard deviation of one, meaning that all the features have the same order of magnitude, and intitialisation of the neural networks are simplified.

3.2.2 Feature Selection

Because there are 57 different features, the search space for a classifier is a 57-dimensional space, and some of these dimensions may not even have a correlation to the emotional class of the sample.

It is desirable to reduce the dimensionality of the search space by eliminating the features that are not relevant to emotional classification, in order to reduce the complexity of the search space.

Peng, Ding and Long ([15]) described the advantages of feature selection:

- It reduces the dimensionality of the search space, which reduces the computational cost.
- It reduces noise in the data, which improves the classification accuracy.
- It leads to more interpretable features.

They distinguished between two feature selection approaches: Filters and wrappers.

Filters select features based on their relevance to the target classes. They are not correlated to the classification method and therefore generalises well when new data is introduced.

Wrappers wrap feature selection around the classification method and are dependant on the accuracy of the classification method. It is possible to find a small set of features that produces a high accuracy, but this comes at an expensive computational cost.

Sequential Forward Selector

In an initial attempt to determine the best features, the Sequential Forward Selector method was used, partly because Martirosian and Barnard [1] reported using a SFS to choose the best features. SFS is a wrapper method as defined by Peng *et al.* [15]).

The SFS method is the simplest feature selection method described in Bishop [14], and was implemented as follows:

1. Put all the features in a list of features F.
2. For each of the features f_i in F, create a training and a test set, train a neural network, and measure the error e_i, as determined by the cross-validation technique described in section 3.3.1, of the classifier against the test set. In the experiment the neural network was set up to have twice as many hidden layer neurons as inputs.
3. Place the feature f_i for which e_i was the lowest into the list L and remove it from F.
4. Repeat from step 2 until L contains the desired number of features or F is empty.

At the end of this process, the list L will contain the features selected from F sorted according to the feature's perceived relevance.

At a later stage, because the gender of the speaker can be reliably determined (See section 3.2.3 below), the gender of the speaker was added as a feature to the selection process.

Initially, the experiment was run without the cross-validation step, but it was found that running the experiment a number of times would produce vastly varying sets of features on each run. The cross-validation was added to obtain a better estimate of the error.

The SFS method was used in a couple of different experiments in this project to select features for different classifiers. The results are mentioned where the experiment is described.

mRMR

Some experiments were also performed with the *minimum Redundancy Maximum Relevance Feature Selection* (mRMR) program presented by Peng, Long and Ding ([15] and [16]) [1].

They described the problem with filter methods that features may be correlated among themselves: Suppose features a and b are correlated, and feature a correlates well with the classification data, then a filter method will choose both features a and b even through b is redundant. Therefore, *"the m best features are not the best m features"* (as they describe it in [16]) when those features are chosen by a conventional filter method.

The mRMR method attempts to minimise this redundancy.

It functions as follows: Suppose that f_i is the ith feature, c the target classes of the classifier and $I(x, y)$ the mutual information between x and y. The redundancy between two features f_i and f_j can then be expressed as $I(f_i, f_j)$ while the relevance of feature f_i to the target classes can be expressed as $I(c, f_i)$.

The mRMR method then attempts to choose a feature set S which minimises $I(f_i, f_j)$ for all f_i and f_j in S while maximizing $I(c, f_i)$ for all f_i in S. In order to prevent a time-consuming exhaustive search, an incremental search is used to find a near optimal set S.

mRMR does not suffer from the same problem as SFS, in that several runs of mRMR will select the same features.

3.2.3 Gender Determination

It was deemed worthwhile to classify the recordings according to the gender of the speaker in order to supplement the feature extraction. If the gender of the speaker could be accurately estimated by a separate gender classifier, then the output of the gender classifier could be used as an input to the emotion classifier.

To reinforce this notion, Ververidis and Kotropoulos [13] suggested that, for example, males express anger with a higher energy and slower speech rate than females. Lee, Narayanan, and Pieraccini [9]'s study also found a gender dependency in the performance of their classifier.

In order to classify recordings according to the gender of the speaker, it was necessary for a human listener to classify all the sample recordings according to the gender of the speaker.

The SPiL program (described in section 4.4 on page 38) was modified for this purpose. A special script was used to play back the first sample in each group and prompted the user to press either 'M' or 'F' on the keyboard (because the recordings were grouped by speaker it was sufficient to play only the first recording in each group in order to save time).

[1] Available online at http://research.janelia.org/peng/proj/mRMR/index.htm

There were 1513 male samples and 1232 female samples in total.

The mRMR program described in the previous section was used to select features that would be suitable for gender classification. The ten most relevant features it selected were:

- The mean of the pitch.
- The minimum pitch value.
- The second largest difference between peaks in the pitch contour.
- The second largest gradient in the pitch contour.
- The pitch standard deviation.
- The longest time between pitch peaks.
- The second largest pitch standard deviation, measured over 300ms intervals.
- The largest pitch standard deviation, measured over 300ms intervals.
- The largest pitch gradient
- The mean pitch standard deviation, measured over 300ms intervals.

A variety of experiments were performed to determine the optimal number of features to use and the number of hidden layers to use. Neural networks are explained in section 3.3.3 below.

The complete results of these experiments are given in section 5.1. In summary, the gender classifier reached about 90% accuracy, and therefore it is feasible to use the gender of the speaker as an input to the emotion classifier.

3.3 Classification

In order to classify the recordings, two approaches were attempted, namely a K-Nearest neighbours approach and a Neural Network-based approach. The K-fold cross-validation technique described in Bishop [14] was used to validate the test results.

3.3.1 Cross-Validation

A problem encountered with the original SFS implementation as presented in section 3.2.2 is that running the SFS experiment several times selects a different set of features on each run of the experiment. The reason for this is that the weights of the neural network is initialised randomly and therefore different runs of the experiment tend to find different local minima in the error curve, and consequently identify different features as the best.

The cross-validation technique described in Bishop [14] alleviates this problem, because it trains several neural networks with the same parameters (but with different initial weights), and then uses the average of the error function to gauge the performance of the classifier.

In K-fold cross-validation method works by dividing the data set into K segments. A training set is then constructed from $K-1$ of these segments and a classifier trained. The remaining segment is then used as the test set and classified with the trained classifier, and the MSE recorded. This process is repeated K times using a new segment as the test set on each iteration so that each of the segments is used as a test set once.

The final error is the average of the errors of each of the K iterations. This approach has the advantage of using a high proportion of the available samples to train the networks while using each of the training samples in a test set, a property that is very useful when the available training data is limited. The main disadvantage of the technique is that the training of the classifier needs to be repeated K times.

A value of $K = 10$ was used in all the experiments in this project as suggested by Bishop.

3.3.2 K-Nearest Neighbours Approach

As described in section 2.3.1, several other researchers (specifically [3], [9] and [5]) experimented with k-Nearest neighbours classifiers.

The KNN classifier is described in detail by Bishop [14]. This classifier can be summarised as follows: Suppose each of the N samples in the training set is represented as a M-dimensional vector $\mathbf{x}^n$ for $n \in 1, N$ where M is the number of features. The features of a sample that is to be classified are then represented a vector $\mathbf{x}$.

The K vectors from $\mathbf{x}^n$ for which the distance between $\mathbf{x}$ and $\mathbf{x}^n$, $|\mathbf{x} - \mathbf{x}^n|$, are the smallest are selected, and then the classification is based on the class of those K features.

In a practical implementation the method involves calculating the distance between $\mathbf{x}$ and each of the N vectors $\mathbf{x}^n$, sorting the results according to the distances and then basing the classification on the class of the first K vectors in the sorted set.

In the practical implementation, samples also had to be discarded when setting up the KNN model to ensure that there are the same number of samples of each class in the M-dimensional space. This was necessary because there are more neutral samples (2022) than active samples (620) and more active samples than passive samples (103) and this may have lead to situations where the K nearest neighbours of a sample were mostly neutral simply because there were more neutral samples. Therefore random active and neutral samples were discarded until only 103 samples of each class remained.

Cross-vallidation had to be used to assess the accuracy of the classifier.

Despite Bishop [14]'s comments that the KNN technique requires large amounts of memory and processing power, the performance of the basic implementation of the technique was found to be adequate because there are not many samples in the training set. None of the more sophisticated versions of the algorithm suggested by Bishop, which use more sophisticated sorting and searching techniques, were deemed necessary.

The high dimensionality of the search space may result in a lower accuracy. Furthermore, all samples are deemed to be equally important in the N-dimensional space. Therefore two separate experiments were performed: In the first, all N features were used in the classifier. In the second, only a subset of the features $\tilde{N}$ were used. This subset consisted of the first 24 features selected by the mRMR program.

The results of both KNN classifier experiments are presented in section 5.2 on page 55.

3.3.3 Neural Network-Based Approach

An artificial neural network (ANN) is a computer program that uses a mathematical function to model the functionality of biological neurons found in the brain.

In a traditional neural network, training is done by showing a selection of training samples to the network, evaluating the output, and then adjusting the parameters (weights) within the function according to the error in order to reduce it.

Based on the enthusiasm of other researchers over neural network-based approaches (See [1], [5], [2],[2] and [7]), it seemed natural to use a neural network classifier.

The software used to implement the neural network was based on a third party library, FANN, and is described in section 4.4.3.

A variety of experiments were done to attempt to maximise the classification accuracy of the neural networks.

Because of the wide variety of parameters that can be set in neural networks in general and in FANN in particular, it was decided to choose a basic set of parameters as a default and use them throughout all the experiments in order to have a common baseline.

In general, the neural networks tended to converge fairly quickly, and therefore in all the experiments the neural networks were trained for only 7500 epochs. The training algorithm used was the RPROP algorithm provided by FANN [17], which is described as an adaptive algorithm. Each neural network had 3 layers: An input layer with a number of neurons equal to the number of inputs into the network, a hidden layer with a number of neurons dependant on the specific experiment performed, and a number of output neurons equal to the number of outputs of the network.

The results of the neural network based classifiers are presented in section 5.3 on page 56.

SFS Feature Selection Using All Emotional States

Initially all three emotional states (active, passive and neutral) were used to develop a classifier.

In order to reduce the dimensionality of the input space, the SFS approach described in section 3.2.2 was used to select the best features

Demonstrating the drawback of the SFS technique described in section 3.2.2, running the SFS experiment twice produced some varying results. In the first run, the eight best features selected by were, in order of importance:

- The largest intensity gradient.

- The longest time between pitch peaks.
- The gender of the speaker.
- The second longest time between intensity peaks.
- The sound intensity.
- The second largest intensity second derivative.
- The standard deviation of the third formant.
- The sound power in the air

Running the experiment a second time resulted in these features being selected:

- The largest intensity gradient.
- The sound intensity.
- The intensity mean.
- The maximum of the first formant.
- The pitch standard deviation.
- The second largest difference between pitch peaks.
- The sound power.
- The mean pitch gradient.

Although the selected features vary, the intensity gradient, the sound power and the pitch of the sound all seem to play a role in the proper classification of the emotion.

The results of the experimentats performed with these features are presented in section 5.3.1.

mRMR Feature Selection Using All Emotional States

A collection of features were selected using the mRMR program described in section 3.2.2.

The eight best features selected by mRMR were the following:

- The second largest intensity second derivative.
- The mean of the first formant.
- The gender of the speaker.
- The maximum of the second formant.

- The sound energy.
- The mean intensity standard deviation measured over 300ms intervals.
- The largest second derivative of the intensity curve.
- The overall pitch range.

SFS Feature Selection Using Only Two Emotional States

Martirosian and Barnard [1] reported that the lack of training data resulted in initially disappointing results of the neural network-based classifier. They therefore developed a second classifier that only classified between active and neutral emotional states.

The reason for discarding the passive state was twofold: Firstly, there are only 103 passive samples, therefore keeping the passive set limits the training and test sets to at most 103 samples of each class in order to eliminate the bias (discussed in section 3.1). Using only the active and neutral samples allows us to use 620 active and neutral samples.

Since this project used the same training samples, a second experiment was done in the same vein to attempt to improve the classification results.

It was decided to reclassify the passive samples as neutral samples instead of discarding them, so as not to loose their relevance.

The SFS technique was applied twice to select features. In the first run the selected features were:

- The largest intensity gradient.
- The intensity maximum.
- The Largest pitch standard deviation, measured over 300ms intervals.
- Maximum of the second formant.
- Largest pitch gradient.
- second longest time between intensity peaks.
- second largest pitch standard deviation, measured over 300ms intervals.
- Standard deviation of the second formant.

Running the experiment the second time selected these features:

- The largest intensity gradient.
- The RMS of the sound.
- The sound intensity.

- The sound power in the air.
- The sound energy in the air.
- The largest intensity second derivative.
- The sound power.
- The mean intensity standard deviation, measured over 300ms intervals.

The features of the intensity curve (its gradient and second derivative) featured again. Features of the pitch curve seemed to play a larger role. In the second run, the first two formants also played a role.

The results of the experiments performed with these features are presented in section 5.3.2.

mRMR Feature Selection Using All Emotional States

As before, a separate experiment was performed with features selected by the mRMR program.

In this case, the eight best features selected by mRMR were the following:

- The second largest intensity second derivative.
- The gender of the speaker.
- The mean of the first formant.
- The sound energy.
- The maximum of the second formant.
- The mean intensity standard deviation measured over 300ms intervals.
- The overall pitch range.
- The largest second derivative of the intensity curve.

This selection of features is very similar to the features selected when all three emotional classes were used (refer to section 3.3.3).

3.3.4 Including Prior Information

In section 3.1 it was mentioned that, based on the work by Martirosian and Barnard [1], an equal number of samples from each emotional class were used in the experiments to eliminate bias that may result from the fact that there are so much more neutral samples than active and especially passive samples.

The fact that 73% of samples are neutral is prior information that may or may not add to the capability of the neural networks to classify samples according to their emotional content.

To determine the effect of this prior information, another experiment was performed where all the samples were used during training and testing. In this experiment all 2745 samples were used. K-fold cross validation was used to determine the performance of the network, with 10% of samples kept in reserve at each iteration as the validation set. The features selected by mRMR were used as inputs to the neural networks.

The results of this experiment is presented in section 5.3.3.

Chapter 4

Implementation

This chapter describes the technical work that was done.

Because part of the motivation for this project is the development of a system that can be used practically, a lot of care and attention was devoted to the implementation of the underlying software that drives the theoretical work described in chapter 3.

4.1 Overview

Figure 4.1 on page 35 shows a high level overview of how all the programs in the system are connected. There are three main programs in the system

- The downloader application (which has been named Downlode), which is responsible for downloading recordings from the call centre.
- The Graphical User Interface (GUI), which provides feedback to the user. It shows recordings that have been downloaded by the downloader and shows how those recordings have been classified. It also gives the user the opportunity to listen to those recordings.
- The classification application (which has been named SPiL), which is responsible for analyzing downloaded recordings and classifying them according to their emotional content.

The general high-level operation of the system is as follows: MTN's call centre uses a suite of applications provided by NICE Systems[1] to store and manage recordings made in their call centre. The downloader application, Downlode, is scheduled to query the NICE database each night to get a list of recordings made in the call centre during the previous day. Downlode then downloads those recordings onto the file system of the server on which the system is deployed and writes an entry into the local database that a recording has been downloaded but not yet analysed. The SPiL application runs continuously and queries the local database for recordings that have been downloaded and not yet processed. It finds the recordings downloaded

[1] http://www.nice.com/

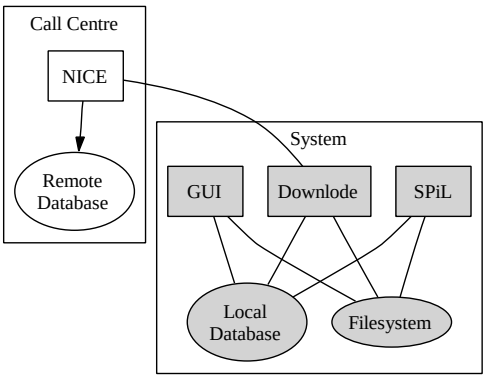

Figure 4.1: Layout of the system

by Downlode on the local filesystem, opens them and classifies them according to emotional content. It updates the local database with the classification result. The user then uses the GUI to look at recordings that have been classified recently and listen to them. The GUI also directs the user towards where the recording originated from in NICE so that the user can locate the call centre agent that handled the call and so forth.

What is not shown in the figure is that the classification application, SPiL, is also used to train its own neural network. This was achieved by building SPiL around a scripting language interpreter. The SPiL application's functionality can then be changed from an application that trains a neural network to one that uses the neural network to classify new recordings by changing the scripts that control the application. This is described in more detail in section 4.4.1.

There are two databases in figure 4.1. The first one is the remote database in the call centre where the NICE application stores recordings. The second database is the local database which forms a part of the system. The remote database is of little interest and therefore in the remainder of this document the term *database* will refer to the local database unless otherwise stated. More detail of how the call centre is laid out is given in section 4.2 on page 36.

To implement the local database, the SQLite database engine was chosen.

SQLite[2] is a well known open source database engine that can be embedded into applications.

At its core, SQLite is a C library that provides a database engine to C programs that link against it through an API. The API is straightforward and this makes it simple to integrate database functionality in applications. The database engine is also atomic, consistent, isolated and durable, which means that it is robust even during power failures and system crashes.

SQLite offers some other desirable features: Because it is embedded into the application, there is no separate database server to install and configure; it is portable between hardware and operating system platforms (see section 4.5.2); it is simple to build from source code and its source code has been placed in the public domain.

[2] http://www.sqlite.org/

SQLite stores its database on a single file on the local hard-drive. In this project, the multiple programs all access this database file by linking them to the SQLite library.

A perceived disadvantage of SQLite is that it does not follow a client-server model. However, SQLite allows multiple applications to access the same database simultaneously, and for the application at hand, the simplicity provided by SQLite's ease of configuration was considered a fair trade-off.

4.2 Downloader Application

MTN uses a suite of products collectively called NiceCLS (Call Logging Systems) provided by NICE Systems to record calls in their call centre and to manage these recordings. NICE Systems provides an API called eNiceLink to query the NiceCLS database in which the recordings are stored (the remote database in figure 4.1).

The Downlode program is responsible for using the eNiceLink API in order to query NiceCLS for new recordings, downloading them and storing references to them in the local database.

The eNiceLink API takes the form of a COM component, which is essentially a DLL compiled in such a way that programs written in any programming language that supports COM can call functions within it.

The eNiceLink API comes with several examples written in Visual Basic, but Downlode had to be written in C/C++ to be able to use SQLite.

Unfortunately the examples and documentation for the eNiceLink API was incomplete or out of date in several instances, so getting the Downlode application to work took some effort. In summary, this is what had to be done:

- The OLE View program that forms part of the tools that are provided with Microsoft Visual C++ 6.0 was used to browse through all the COM components on the system on which the eNiceLink API was installed in order to locate the `eNICE` class.
- The OLE View program was then used to generate an Interface Definition Language (IDL) file for the `eNICE` class. The IDL file describes the COM component in the same way in which a class describes a software object in an object-oriented programming language.
- The IDL file had to be modified by hand before it could be compiled by the MIDL compiler (which is Microsoft's IDL compiler implementation).
- The output of the MIDL compiler is a C++ header file that describes the eNiceLink COM component as a C++ class named `IeNICE` that can be instanciated through the `CoCreateInstance()` function of the COM API.

The `IeNICE` class provides methods for connecting to NiceCLS, querying for new recordings and downloading those recordings. In principle, the Downlode program is quite simple:

- Initialise the COM API.

- It creates an instance of the `IeNICE` class through COM.
- It uses the `IeNICE` instance to connect to NiceCLS.
- The query method of the `IeNICE` instance is used to query NiceCLS' database for new recordings.
- It retrieves the details of each recording returned in the result of the query.
- Each recording in the query result is saved as a wave file onto the local filesystem. The details of the recording is stored in the local database (as described in section 4.1 on page 35)
- The `IeNICE` instance's disconnects from NiceCLS.
- To clean up, the `IeNICE` instance is released and the COM API is closed.

Downlode runs in the background during off-peak hours meaning that the user never notices its performance.

To be able to use the COM functionality, Microsoft's Visual C++ compiler had to be used. MS Visual C++ 6.0 was conveniently available under a student license and used to compile the application.

Downlode is not an interactive application. It requires no user input in order to function properly. It is configured through an INI configuration file that is read by the iniParser library[3]). This file contains several parameters that tells Downlode where to locate the NICE server, how to choose recordings to download and where to save the downloaded files.

4.3 Graphical User Interface

The GUI is a Windows application that provides the user with a view of the local database. The open source toolkit wxWidgets[4] was chosen as the platform on which to develop the GUI for a variety of reasons:

- Its portability. wxWidgets applications can run on a variety of platforms including Windows and Linux.
- Its liberal open source license. wxWidgets is licensed under the LGPL, with a special exception for static linking.
- Its ease of use.

[3] http://ndevilla.free.fr/iniparser/
[4] http://www.wxwidgets.org/

wxWidgets is a collection of libraries that provide a common API for writing graphical user interfaces on a variety of platforms, providing access to native controls where they are available and emulating when they are not, allowing applications to have the look and feel of the platform on which they are running, and the same functionality across all platforms. wxWidgets also provides some additional functionality that is commonly used in GUI applications. In particular, it provides functions for reading and writing INI configuration files and access to the underlying platform's audio playback facilities, both used in the GUI.

The GUI is presented as a grid similar to those found in spreadsheet applications through wxWidgets' wxGrid control.

The GUI uses SQLite to query the local database (described on page 35) and populate this grid. Within the grid each row represents a recording with each column representing some attribute of that recording, such as its filename and duration. Some of the columns can be used to locate the original recording in the NICE database, as well as the call centre agent attending the call.

The most important column is the column that shows how the recording was classified by the classifier application.

When selecting a row in the GUI, it looks up the selected recording in the database and allows the user to listen to it through a playback dialog.

The GUI can run under both Microsoft Windows and Linux because of the portability of both wxWidgets and SQLite.

It would have been preferable to compile the Windows version of GUI with the open source Mingw compiler because MinGW is used to compile SPiL (section 4.4) and is freely available, but the wxWidgets libraries are particularly complex. Writing anything other than a trivial application requires complex linking, and this task was easier achieved through the Microsoft Visual C++ compiler.

4.4 Classification Application

The classification application, named SPiL, consists of these four major componenents:

Database The classification application needs to be able to query the database populated by Downlode for downloaded recordings that have not yet been processed. It also needs to write the result of classification to the database so that these results can be viewed in the Graphical User Interface.

Sound Driver Once downloaded recordings have been identified, the sample data they contain need to be loaded from the disk for processing.

Speech Processing Once the sound file has been loaded, it needs to be processed to extract features. This component should be able to extract the pitch, intensity and Formants and other interesting characteristics of the voice recordings to aid in classifying them.

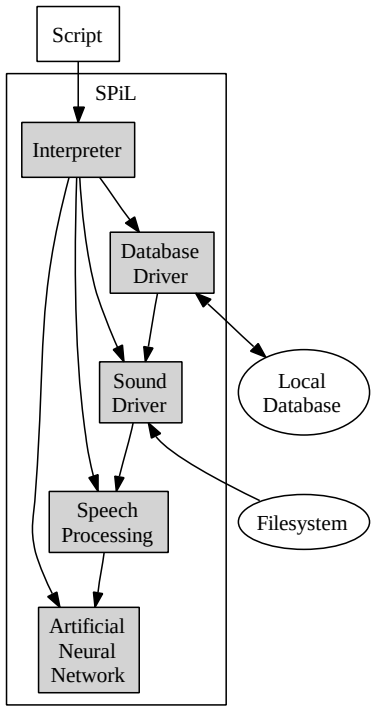

Figure 4.2: Architecture of SPiL, the classification application

Neural Network Once features have been extracted, they can be passed to a neural network which will classify the recording according to its emotional content.

The classification application was intended from the beginning to be a flexible speech processing application. One of the reasons for this requirement is that classification applications built around neural networks typically require two major parts: The first part trains the neural network on training data and the second part applies the trained neural network to the production data to perform its classification role. Having a flexible speech processing application would allow for both of these parts to be performed by the same program, with only a change in how the program is configured.

Having a flexible speech processing application also has the benefit of being easy to reconfigure in order to experiment with different neural network (and, to a lesser extent, speech processing) configurations and parameters.

The solution to this requirement for a flexible speech processing application was found in binding all the various aspects of the application with a scripting language interpreter. The application takes as input a script that is passed to the interpreter which performs actions such as querying the database, loading sound files, processing the sound file and classifying it through a neural network based on instructions in the script file. This architecture is displayed in figure 4.2.

It was highly desirable to use open source components wherever possible in this project. In the final implementation of SPiL, these components were used:

Interpreter The Lua scripting language was chosen as the interpreter for the application. The name of the classification application, SPiL, is an acronym for *Speech Processing in Lua*.

Speech Processing As already mentioned, the PRAAT tool was chosen to perform speech processing.

Neural Network The Fast Artificial Neural Network Library (FANN) was chosen as the neural network implementation.

Database As described in section 4.1 SQLite was chosen as the local database, because this database is shared between SPiL, Downlode and the GUI.

Sound Driver An open source library called Libsndfile was chosen as the sound driver that would be used to load recordings from the file system for use within SPiL.

An advantage of reusing open source components is the fact that these components are all mature projects that have been developed by contributions from people with expertise in the problem domain of the components.

These components have all undergone several years of development and as such, have undergone intense public scrutiny by users. Having the source code available also helps the community around these projects identify problems and to improve the products.

Furthermore, some of these projects are so well established that they don't have closed source competition anymore. There are, for example, no well-known widely used closed-source alternatives to either SQLite or Lua.

The use of open source components also opened possibilties that would not have been possible otherwise. Using an interpreter in this project would have been impossible were it not for the availability of open source interpreters such as Lua.

Likewise, the data storage components used among the programs in this project would probably have consisted of a series of flat files that would have been complex to manage and organise were it not for the availability of the SQLite database engine.

The only disadvantage to using open source components is the potential confusion about the wide variety of licenses and the implications of using them in the same project. More is said on this issue in section 4.5.1.

In the initial design a different open source tool, SPTK (see [18]), would have been used to do the processing of the voice recordings. The primary reason for this decision was that SPTK's source code seemed more easily adaptable for use in SPiL than the alternative, PRAAT, which is also an open source tool. SPTK is also more liberally licensed than PRAAT. Later during the project the decision was made to drop SPTK in favour of PRAAT for a couple of reasons. The name SPiL was originally meant to be an acronym for *SPTK in Lua* but it has been changed to be a more general acronym for *Speech Processing in Lua*.

These components are shown in figure 4.3 and described in the following sections.

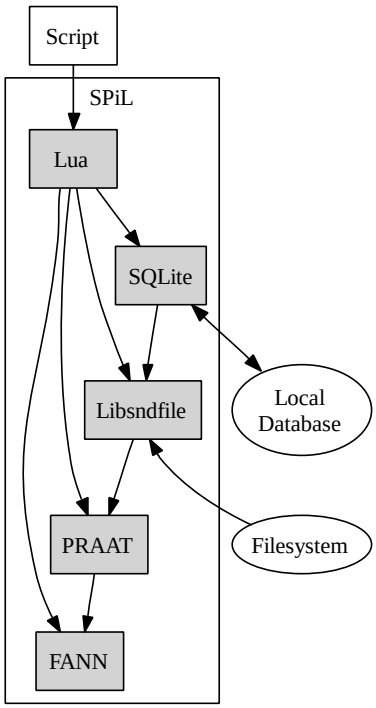

Figure 4.3: Components of SPiL

4.4.1 Interpreter

The powerful Lua scripting language[5] was chosen as the interpreter for SPiL.

Lua is a well-known open source, embeddable interpreter for a Pascal-like scripting language. In addition it provides several features commonly found in modern languages like dynamic typing, associative arrays and object-oriented symantics.

Lua is fast because of it register-based virtual machine. It is portable to just about any platform that has a C compiler. The interpreter is small. Lua's source code is distributed under the liberal MIT open source license. Although the language provides powerful constructs, the core language is simple and easy to learn.

Lua is said to be embeddable because it compiles to a library that can be linked with other C/C++ *host* applications to provide them with scripting functionality. These host applications can provide the Lua interpreter with functions written in C/C++ in such a way that these functions are callable from a Lua script (refer to [19] for more information). This process is referred to as creating Lua *bindings* for the C/C++ host application's functions.

Major effort was made during the development of SPiL in the creation of such Lua bindings for the sound file library, the database driver (both described in section 4.4.4), the neural network (described in section 4.4.3) and, most importantly, the functions to analyse the voice

[5] http://www.lua.org

recordings (described in section 4.4.2).

The following is an example of a Lua script that opens a sound recording in a file named `sample.wav`, uses the PRAAT bindings (described in section 4.4.2) to analyse the pitch and print some statistics of the pitch (the mean, mean strength and the standard deviation) to the console:

```
snd = sndfile.open("sample.wav")

pitch = PRAAT.getPitch(snd)

print("Mean Pitch..............: " .. pitch:getMean())
print("Mean Pitch Strength.....: " .. pitch:getMeanStrength())
print("Pitch Std Deviation.....: " .. pitch:getStdDev())
```

SPiL also has a *console mode* wherein Lua statements can be entered from the keyboard instead of running a script from a file. This makes experimenting with different commands or different parameters of the various subsystems a lot simpler. These commands are logged to a file for future reference.

As with other popular scripting languages, Lua has strong support for regular expressions. Taking advantage of this, a Lua script was written to extract comments from SPiL's C source code and generate HTML documentation from it as part of the build process. Special character sequences in the comments are replaced with HTML tags to style the document. This process ensures that SPiL's documentation is always up to date as long as the C source code comments are kept up to date.

4.4.2 Speech Processing

The PRAAT program from Boersma and Weenink [12] was used for processing the speech recordings.

PRAAT is a computer program that provides several different kinds of speech analysis tools. PRAAT is open source and licensed under the GPL[6] .

PRAAT provides these features that were desirable for the purposes of SPiL:

- Pitch analysis.
- Intensity analysis.
- Formant analysis.

PRAAT also provides several other features that are useful for speech research in general, but not relevant for this specific project:

- Graphical representation of sound properties and output to PostScript for printing.

[6] http://www.gnu.org/licenses/gpl.html

- Speech synthesis.
- Speech manipulation.
- Programmability through its own scripting language.
- Statistics such as multidimensional scaling, principal component analysis and discriminant analysis.

PRAAT is also portable across platforms, making it desirable for reasons discussed in section 4.5.2.

Because PRAAT's source code is available, the useful portions of its code were extracted into a separate library to which bindings for Lua were written to make up the feature extraction portion of SPiL.

The biggest problem with PRAAT is that it is a monolithic graphical user interface program that makes extensive use of the C preprocessor to be portable. This makes it very difficult to reuse PRAAT's source code for anything other than PRAAT.

Daniel van Niekerk of the Meraka institute provided the source code of a small program he called *Getpitch* that uses PRAAT's source code to extract the pitch and intensity of an audio file. This program provided the necessary insight into the layout of PRAAT's source code to reuse it within SPiL.

For the purposes of SPiL, the *Getpitch* program's source code was modified and reorganised to compile to a library to which SPiL links. The pitch and intensity extraction in this library was made reentrant in order to simplify the Lua bindings.

Lua bindings were added to call functions within the new *Getpitch* library. This gave SPiL the ability to extract the pitch and intensity from an audio file.

Functions that extract the statistics (mean, standard deviation, minimum and maximum) from the pitch and standard deviation were extracted from PRAAT's source code and added to the *Getpitch* library.

The *Getpitch* library still lacked the ability to extract and process the formants from the recording. These were also extracted from PRAAT and added to the *Getpitch* library, along with functions to extract their statistics.

A final addition was the ability to extract parts of the audio file, such as a 300ms window.

SPiL was linked against the *Getpitch* library. The required Lua bindings for the functions within *Getpitch* were added, giving SPiL most of the fundamental functionality required to analyse speech signals.

The result is that with one Lua function call the pitch or intensity of an audio file can be extracted and with a second function call the statistics can be extracted. An example of a script that extracts statistics from an audio file's pitch was given in section 4.4.1

The SPTK Alternative

As mentioned in section 4.4 the initial design of SPiL preferred the SPTK program (from [18]) to PRAAT.

SPTK is a set of small speech processing programs, where each program performs a specific task and the output of one program is piped to the input of the next through the standard Unix-style piping mechanisms.

SPTK was initially preferred for two reasons, namely:

- SPTK's source code is much simpler than PRAAT's. Although both are written in C, SPTK is a collection of small command line programs that link to a common utility library. The idea was to link SPiL against the utility library and to create Lua bindings to SPTK based on each of the small programs. In contrast, PRAAT's source code is complex and difficult to read.

- SPTK is licensed under the MIT open source license, which is a more permissive open source license than the GPL under which PRAAT is licensed. Had SPTK been used, SPiL would not have had to be licensed under the GPL.

Later during the project it became clear that using SPTK would complicate tieing in with the work done at the Meraka institute. The researchers at Meraka use PRAAT extensively and while the features of SPTK overlap with the features of PRAAT, there were some PRAAT features that SPTK lacked, most notably the extraction of formants.

Therefore the decision was made to abandon the SPTK bindings in SPiL in favour of bindings to PRAAT's source code.

4.4.3 Neural Network

The Fast Artificial Neural Network Library (FANN) (refer to [17] and [20]) was chosen as the artificial neural network implementation with which the neural network experiments described in section 3.3.3 were implemented.

The FANN library provides an API for creating sophisticated neural network structures. It is written in C and portable to a variety of platforms. FANN was chosen because it is feature-rich, easy to use and freely available as open source software.

Although FANN is written in C, it has bindings for several other programming languages, including Java, Python, Perl and Matlab. Unfortunately it does not have bindings for Lua, therefore Lua bindings had to be written for this project.

The following is an example of a Lua script that uses these bindings to create a neural network with two inputs, and one output. It then loads a set of training data from a file named `xor.data`, adjusts the parameters of the network and trains it according to the training data. It finishes by saving the trained network to a file named `myxor.net`. The training set in the file `xor.data` trains the network to learn the exclusive-OR function.

```
-- Create a network with three layers; two inputs, two hidden layer
-- neurons and one output
ann = fann.create_standard(3, 2, 2, 1)
```

```
-- Load training data from "xor.data"
train = fann.read_train_from_file("xor.data")

-- Set some parameters
ann:set_activation_steepness_hidden(1)
ann:set_activation_steepness_output(1)

ann:set_activation_function_hidden(fann.FANN_SIGMOID_SYMMETRIC)
ann:set_activation_function_output(fann.FANN_SIGMOID_SYMMETRIC)
ann:set_train_stop_function(fann.FANN_STOPFUNC_BIT)

ann:set_bit_fail_limit(0.01)

-- Initialise the weights based on the training data
ann:init_weights(train)

-- Train the network on the training data
ann:train_on_data(train, 500000, 1000, 0.001)

-- Save the trained network to a file
ann:save("myxor.net")
```

The FANN bindings that were written for Lua have been cleaned up during the course of the project and donated to the open source community. A project under the name of LuaFann has been set up on the LuaForge[7] web site which is a host for a variety of Lua-based open source projects.

4.4.4 Additional Bindings

SQLite Bindings

There are SQLite bindings for Lua available on the Internet, but all of these seemed immature at the onset of this project, and so a simple set of SQLite bindings for Lua were developed.

The following is an example of a Lua script that uses these SQLite bindings to open a database named `recordings.db`, query a table named `files`, print the results on the console and delete all rows in the in the table where the `num` field's value is 30:

```
db = sqlite.open("recordings.db");

result = db:query("select num,name,path,duration from files");
```

[7] http://luaforge.net

```
print("Rows returned: " .. result.rows);

for i=1,result.rows do
    c = ""
for j=1,result[i].cols do
c = c .. result[i][j] .. ", "
end
print(c)
end

result = db:query("delete from files where num=?;", 30);
print("Rows returned: " .. result.rows);

db:close();
```

PRAAT itself also provides some neural network functionality, but a dedicated neural network in the form of FANN was preferred. FANN proved to be easier to integrate into SPiL than any of the re-used PRAAT components.

Libsndfile Bindings

The open source library Libsndfile[8] was chosen to load sound files. It supports a wide variety of audio formats. As with the other open source libraries used within SPiL, I wrote Lua bindings for the functions within Libsndfile.

The Libsndfile binding exposes only one function, `open()`, to Lua. An example of it usage is the statement `sndfile.open("sample.wav")`, which will open a file called `sample.wav`.

Special care had to be taken to ensure that the internal format of the loaded sound file was compatible with the format used by the PRAAT bindings described in section 4.4.2.

libcsv

At a late stage during the development of SPiL, the decision was made to add support for comma-separated values (CSV) files.

The CSV file format is convenient for storing the types of data generated by this project because CSV files are easy to generate programmatically, human readable in standard text editors and editable in a variety of spreadsheet applications.

CSV files are mainly used to store the features extracted from the data corpus as described in section 3.2, where each row represents a recording and each column in the file represents a feature of that recording.

Having a method to process CSV files from SPiL was highly desirable, because it would allow the creation of a variety of Lua scripts to process the extracted features. The creation of

[8] http://www.mega-nerd.com/libsndfile/

FANN training files and the normalisation described in section 3.2.1 are the most prominent examples of this.

An open source library, libcsv[9], was chosen for this purpose because of its simplicity. Lua bindings had to be written around the libcsv API for reading a CSV file from the disk, getting and setting the values of individual *cells* within the file and writing the file to disk.

Lutils

Additionally a small module of utility functions was written. This module is called Lutils and provides some supplemental functionality not provided by the standard Lua libraries, such as changing directory and listing all the files in a directory.

Lutils was written in such a way that it compiles on Windows and Linux and provides the same functionality on both platforms. It uses the C preprocessor to conceal the differences between the two platforms.

The Linux version uses only POSIX functions and is expected to compile on other POSIX platforms with little or no modification. This could unfortunately not be verified.

4.5 Other Considerations

There were some other considerations in the choice of third party components. The most notable of these were the licensing issues, which would affect how the system could be used and deployed, the portability issues, which would affect where the system could be used and deployed, and the stability issues that would determine how well the system would work once it has been deployed.

4.5.1 Licensing Issues

Because one of the goals of this project was examining the practical implementation of a emotion recognition system, care had to be taken with regard to how the licenses of the third party components affects the licensing terms of the source code of this project itself.

At the onset of the project the various licenses used by open source packages had to be scrutinised:

- Non-copyleft[10] open source licenses were preferred, because they place less restrictions on how the final software can be redistributed. These are licenses such as the MIT license under which both Lua and iniParser are distributed. SQLite has been placed in the public domain, so it also falls into this category.

[9] http://sourceforge.net/projects/libcsv/
[10] http://www.gnu.org/copyleft/

- LGPL-licensed components are acceptable because the copyleft terms contained therein does not affect the program linking against it as long as such linking is dynamic. wxWidgets, Libsndfile, FANN and libcsv are distributed under this license.
- Copyleft licenses such as the GPL were to be avoided, because they require that the resulting software also be released under copyleft conditions.

The problem with copyleft licenses are that they require the source code be made available. In this project specifically, if the system is to be installed at MTN, the GPL requires that, at the very least, an offer should be made to provide MTN with the source code, and nothing would prevent MTN from supplying the source code to other parties. While this corresponds to the spirit of free software, complications concerning the ownership of the intellectual property may arise, since this is an academic project.

One of the reasons why SPTK was preferred to PRAAT is that SPTK is licensed under the MIT license while PRAAT is licensed under the GPL. However, because of the technical reasons mentioned in section 4.4.2, PRAAT became the preferred choice for speech processing, and because PRAAT is licensed under the GPL, SPiL now also has to be licensed under the GPL.

A potential pitfall in adopting the GPL was that other components may have used licenses which are incompatible with the GPL. Fortunately, all the other licenses used by the other components in the project are GPL-compatible.

4.5.2 Portability Issues

With regard to SPiL, the portability of the third party components was one of the criteria for their selection. All the major components (Lua, PRAAT, FANN and SQLite) are written in the C language and this benefits their portability a great deal.

Because these components are cross-platform, the SPiL program compiles under Windows exactly as it would compile under Linux, with only a minor modification to its Makefile. The potential also exists for it to compile under other operating systems, but this has not yet been attempted.

To compile SPiL under Microsoft Windows, the MinGW compiler was used exclusively. The MinGW compiler is a Windows port of the open source GCC compiler. Care was taken to write the SPiL C source code in a way that uses only ANSI standard C constructs and library functions.

The graphical user interface was built using the wxWidgets toolkit, which is a cross-platform graphical user interface toolkit. The graphical user interface uses wxWidgets exclusively to provide an abstraction of the underlying platform.

The GUI runs under both Microsoft Windows and Linux. It also has the potential to run under other platforms supported by wxWidgets and SQLite, such as Mac OS X.

Because of its reliance on the eNiceLink API, which is a COM component, Downlode is limited to compile and run on Windows only. Downlode is therefore the only program developed for this project that does not have at least the potential to be run cross-platform.

4.5.3 Stability Issues

In order to ensure that SPiL runs without memory leaks and crashes it was extensively tested with Valgrind[11] throughout its development. Valgrind is a tool that detects memory leaks and memory corruption bugs in programs by emulating the underlying hardware.

Running SPiL with all its test programs through Valgrind, especially after major changes and additions, and eliminating any problems reported by Valgrind helps to ensure that SPiL will run without problems.

Valgrind is a Linux-only tool, but since the SPiL code does not vary much between the Linux and Windows versions there is still a lot of confidence in the Windows version.

In addition, SPiL's makefile includes a debug option that compiles SPiL with no optimizations and with debugging symbols. It also enables runtime assertions.

Compiling the program with the debug option allows the easy debugging of SPiL with the GNU Project Debugger, GDB[12].

4.6 Alternative Designs

As with any project, there were several alternative design ideas to consider.

The Python[13] programming language has become very popular as of late and would have been a very suitable alternative to Lua. Like Lua, Python can be embedded into C/C++ applications and extended with modules written in C/C++.

Python is becoming particularly popular in the scientific community and this community has produced several modules that would have been useful in this project, such as graphing modules.

Python comes standard with bindings for SQLite, so it would have been unnecessary to write SQLite bindings as have been done for Lua.

The FANN project also offers bindings for Python on its website, and so the bindings described in section 4.4.3 would not have had to be written.

In the end, Python may have been a better option than Lua had it not been considered too late in the project. This is not to say that Lua was a poor choice, just that Python may have been a better choice for this particular application.

Python also has bindings to wxWidgets called wxPython. Using Python would have decreased the comlexity of the source code of the graphical user interface. Using an interpreted language such as Python typically results in slower programs than when using a compiled language like C++, but in the development of non-computationally intensive graphical user interfaces this is hardly ever a noticeable problem.

As described in section 4.4.2, the SPTK program was considered a viable alternative to using PRAAT during the early stages of this project. Section 4.4.2 also mentions why PRAAT

[11] http://valgrind.org/
[12] http://www.gnu.org/software/gdb/
[13] http://www.python.org/

eventually replaced the SPTK components.

Another alternative to PRAAT was the Tcl/Tk based program WaveSurfer[14] which provides a Tk-based user interface that is similar to PRAAT's.

WaveSurfer is licensed under the BSD open source license, which is also an ideal permissive license. WaveSurfer itself is written in Tcl/Tk, but its underlying mechanisms are written in C, and it may have made a viable alternative. Unfortunately, its documentation is not very accessible, and therefore it was not seriously considered.

There are alternative open source databases available, like PostgreSQL and MySQL, which would have been suitable for this project. Both of these are client-server databases with the associated advantages, like flexibility in how they are deployed, and disadvantages, such as added complexity in the installation and management. In the end, the simplicity offered by SQLite far outweighed the flexibility of any client-server databases.

4.7 Determining the Speech Intervals

In a typical call centre conversation a recording may start with the agent greeting the customer, the customer stating his or her problem, the agent performing an action, during which time no speech occurs, the agent reporting back to the user and the call being terminated.

Therefore, a final problem that has to be addressed for a practical implementation is the determination of the segments in the recording where speech occurs.

This is important because the call centre agent and the customer may have different emotional states. The agent may be calm while the customer irritated. Therefore it would be advantageous to classify the emotional content of these segments separately.

Furthermore it eliminates the silent segments in the conversation, such as when the agent performs a lookup on the computer.

The following approach has been taken to address this issue:

1. The recording is split into 100 ms intervals.

2. For each interval, the Root Mean Square (RMS) of the amplitude, r_i, is calculated to produce a series of RMS values $\mathbf{r} = \{r_1, r_2, \ldots, r_n\}$.

3. The series of RMS values $\mathbf{r}$ are then filtered through a reasonably simple filter according to the equation

$$\tilde{r}_i = \frac{1}{R} \sum_{j=-\lfloor R/2 \rfloor}^{\lfloor R/2 \rfloor} r_j \tag{4.1}$$

where R indicates the order of the filter, and is typically a small, odd number. The filtering serves two purposes: It eliminates noise in the RMS signal and it helps the algorithm to account for small pauses in the speech between syllables.

[14] http://www.speech.kth.se/wavesurfer/

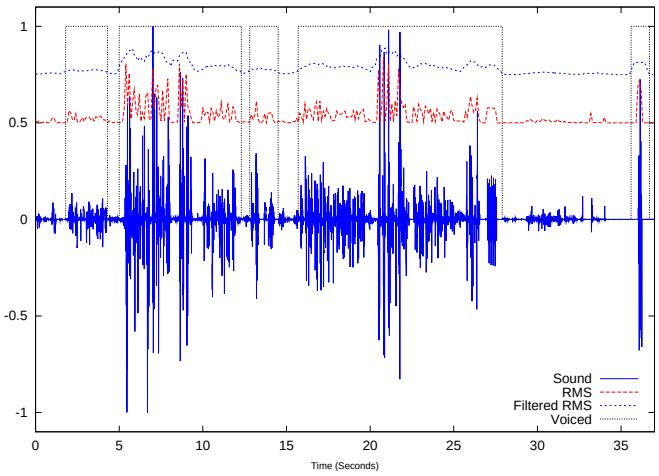

Figure 4.4: Determining the Speech Intervals

4. The filtered RMS value is then compared to a threshold T_{RMS} and a new set of values $\mathbf{v} = \{v_1, v_2, \ldots, v_n\}$ created such that

$$v_i = \begin{cases} 1 & \text{if } \tilde{r}_i > T_{RMS} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

5. Each v_i now represents a 100ms interval where the algorithm believes to contain speech. The last step in the algorithm is to isolate the individual segments. All segments where 10 or more consecutive v_is have a value of 1 are isolated because they indicate speech segments longer than 1 second in duration. Segments where the speech duration is less than 1 second are discarded.

In the final implementation the order of the filter, R, was set to 7 and T_{RMS} was set to 0.015. These values were determined through experimentation by examining recordings in the audio processing application Audacity[15].

Figure 4.4 demonstrates this process graphically for an actual recording. In the figure the dotted red line shows the RMS, r_i, at each interval, the dotted blue line shows the filtered RMS values, $\tilde{r}_i$, and the black dotted lines labeled "Voiced" indicate the areas where the filtered RMS exceeds T_{RMS}, that is, the intervals this algorithm believes to contain voiced speech. (The values of r_i and $\tilde{r}_i$ have been offset by 0.5 and 0.75 in the graphic for clarity).

Unfortunately, this technique cannot distinguish between actual speech and beeps and other sounds of the telephone system. In the example presented in figure 4.4 the interval identified between 35.6 and 36.7 seconds is in fact a telephone ringing sound which the algorithm cannot distinguish from normal speech, but the sound between 29 and 33 seconds is background noise which the algorithm removes effectively.

[15] http://audacity.sourceforge.net/

In further research, a more sophisticated approach could use additional features from the recording, such as the pitch and the formants, in order to circumvent the shortcomings of the approach presented in this section.

Chapter 5

Results

In this chapter, the results of the experiments performed in chapter 3 are documented.

- We present the results of the gender classification experiments of section 3.2.3 in section 5.1 on page 53.
- We then present the results of our k-Nearest Neighbours experiments in section 5.2 on page 55. The experiments themselves are described in section 3.3.2.
- We lastly describe the results of the neural network experiments from section 3.3.3 in section 5.3 on page 56.

5.1 Gender Classification

Classifying the recordings according to the gender of the speaker described in section 3.2.3 turned out to be remarkably effective.

The experiments were aimed at determining the optimal number of features and the number of hidden layer neurons: It started by selecting the first feature selected by mRMR (see section 3.2.3). It then used cross-validation to measure the average classification rate of neural networks with 5, 10 and 15 hidden layer neurons. It then repeated this using the first two, three, four, and so on, features selected by mRMR.

Table 5.1 and figure 5.1 presents the results.

As can be seen from the graph, using 5 hidden layer neurons seems to be generally better, and five features seem to be the optimal number, regardless of the number of features.

Table 5.2 shows the confusion matrix of one of the trained neural networks. In the confusion matrix, the row corresponds with the actual class of the sample while the column corresponds with the class it was classified as. Thus, the values in the main diagonal of the matrix represent the samples that were correctly classified.

As can be seen from the confusion matrix, 88.09% of samples in the test set were correctly classified.

Number of Features	5 Hidden Neurons	10 Hidden Neurons	15 Hidden Neurons
1	87.55%	87.48%	87.66%
2	87.85%	87.70%	87.77%
3	88.72%	88.32%	88.28%
4	88.58%	88.18%	87.96%
5	91.72%	90.95%	90.04%
6	91.06%	89.82%	89.64%
7	91.39%	89.85%	89.20%
8	90.84%	89.85%	87.96%
9	90.22%	90.15%	88.72%
10	90.95%	89.71%	88.54%

Table 5.1: Confusion matrix for the neural network gender classifier.

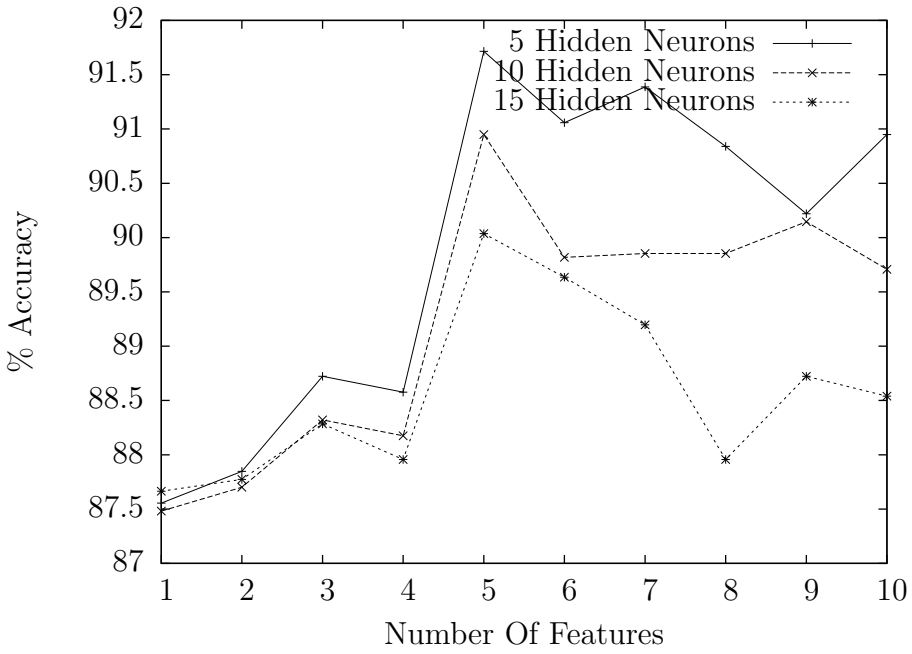

Figure 5.1: Results for the gender experiments.

	Male	Female
Male	373	22
Female	76	352

Table 5.2: Confusion matrix for the neural network gender classifier.

It can also be seen that more female samples were incorrectly classified as male than male samples classified as female.

5.2 k-Nearest Neighbours Approach

5.2.1 Using All The Features

Initially in the KNN experiments, a KNN model was built using all 57 features from section 3.2, so that the nearest neighbours would be searched in a 57-dimensional space.

In the first experiment, the parameter K was varied from 1 to 30 and the accuracy measured for each KNN mode through cross-validation.

The results of this experiment is shown in table 5.3 and figure 5.2.

K	Accuracy	K	Accuracy
1	32.99%	16	33.63%
2	43.96%	17	34.54%
3	37.11%	18	36.60%
4	39.45%	19	33.39%
5	35.52%	20	32.36%
6	33.39%	21	35.37%
7	35.77%	22	31.81%
8	37.08%	23	33.35%
9	37.79%	24	33.05%
10	34.53%	25	32.97%
11	38.06%	26	31.21%
12	35.97%	27	29.62%
13	34.15%	28	29.64%
14	33.73%	29	29.18%
15	33.70%	30	29.79%

Table 5.3: Results for the KNN model when varying the value of K.

5.2.2 Using A Subset Of The Features

Because of the high dimensionality of the feature space, a second experiment was performed where subsets of the features were used to produce a new feature vector $\tilde{x}^n$ was created from x^n.

The features selected were the first 24 features selected by the mRMR program described in section 3.2.2.

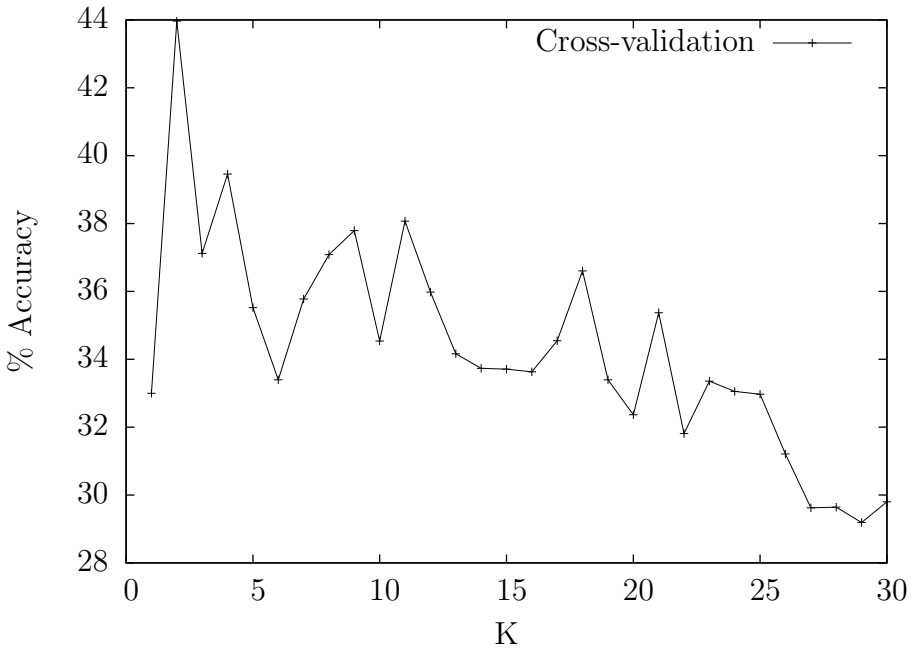

Figure 5.2: Results for varying K.

The experiments performed started with one feature, measuring the accuracy when K is 5, 10 and 15 using cross-validation, and then repeating the process, adding the next feature until all 24 features have been evaluated.

Table 5.4 and figure 5.3 presents the results.

Figure 5.3 appears to indicate very little correlation between the number of features and the accuracy of the KNN model (for example, it reaches a peak at 21 features, and then a valley at 22 features).

The correlation between the accuracy and the value of K is due to the fact that for each experiment the same subset of samples were used to create the KNN model, while when adding a new feature a new set of samples were chosen.

The best results are achieved when K is 5 using 21 features. Figure 5.5 shows a confusion matrix of such a KNN model when it was tested against all the sample data.

5.3 Neural Network-Based Approach

This section presents the neural network experiments described in section 3.3.3 results.

Number of Features	$K = 5$	$K = 10$	$K = 15$
1	43.16%	40.95%	40.54%
2	44.29%	43.10%	42.19%
3	41.28%	38.52%	40.70%
4	42.88%	39.49%	36.54%
5	36.41%	34.79%	33.89%
6	41.24%	39.67%	38.53%
7	39.10%	35.75%	36.26%
8	42.21%	42.10%	39.29%
9	41.81%	43.01%	40.41%
10	33.22%	33.81%	38.97%
11	42.90%	39.16%	35.66%
12	45.97%	42.45%	38.25%
13	39.51%	40.70%	44.85%
14	44.96%	39.71%	37.99%
15	42.99%	42.77%	36.51%
16	32.79%	29.47%	30.93%
17	40.41%	38.49%	36.69%
18	42.30%	38.69%	37.15%
19	42.74%	36.27%	37.07%
20	39.90%	35.45%	33.38%
21	51.30%	43.28%	39.08%
22	33.47%	25.37%	25.00%
23	36.99%	32.48%	33.48%
24	50.09%	39.93%	33.90%

Table 5.4: Results of using a subset of the features for varying the parameter K.

	Neutral	Active	Passive
Neutral	1060	417	545
Active	237	182	201
Passive	18	9	76

Table 5.5: Confusion matrix for the KNN classifier.

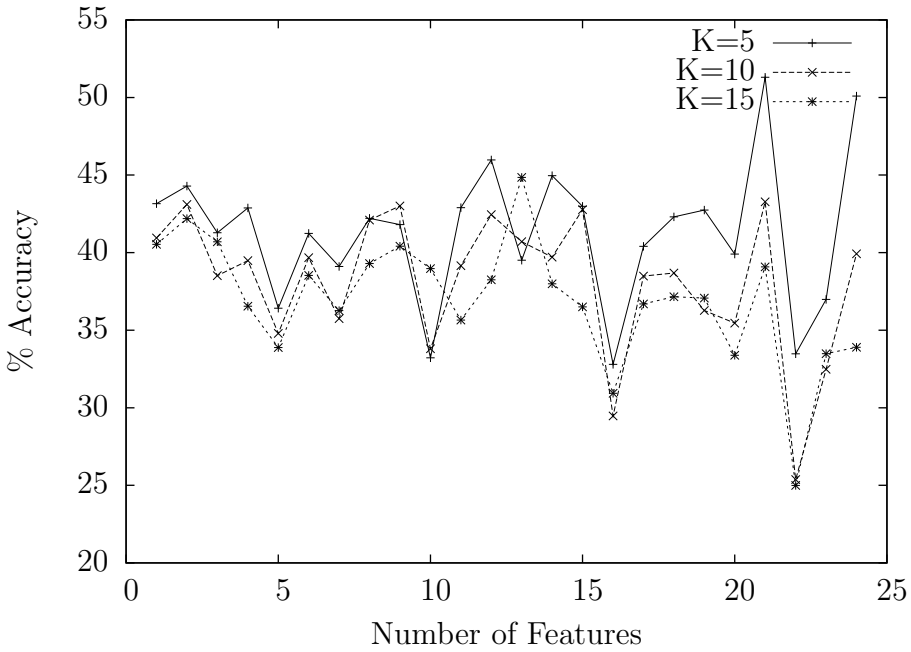

Figure 5.3: Results for varying K when using a subset of the features.

5.3.1 Using All Emotional States

Using the SFS Features

Having chosen a good set of features using the SFS method, the next step is to choose the optimal number of features and an optimal number of hidden layer neurons for the neural network.

In the first experiment the number of features were varied from 1 to 8 in the order they were selected by the SFS method. On each iteration neural networks with 1, 3, 5, 10 and 15 hidden layers were trained. The performance of the model was evaluated with the K-fold cross-validation.

Table 5.6 and figure 5.4 presents the results of this experiment on the first set of features chosen by SFS, as listed in section 3.3.3. In the columns, the parameter H is the number of neurons in the hidden layer.

The percentages listed are the average percentage of samples correctly classified for each iteration of the cross-validation. For cross-validation, the paramter K was kept at 10.

The experiment was then repeated for the second set of SFS features from section 3.3.3. The results are presented in table 5.7 and figure 5.5.

Both runs of the neural network with the SFS-selected features seem to be roughly similar. The second set of features seems to provide slightly better results.

The best results achieved were 52%, with the first two features of the second feature set and 5 hidden layer neurons. Table 5.8 shows a confusion matrix for this case.

Features	$H = 1$	$H = 3$	$H = 5$	$H = 10$	$H = 15$
1	38.33%	40.33%	46.67%	48.33%	44.67%
2	40.33%	44.33%	43.33%	41.33%	44.67%
3	42.00%	45.33%	39.67%	45.33%	38.00%
4	50.67%	43.67%	43.00%	42.33%	39.33%
5	44.33%	48.67%	50.00%	38.00%	44.00%
6	44.00%	43.33%	44.67%	45.00%	40.33%
7	41.00%	49.00%	48.67%	47.00%	41.67%
8	36.00%	47.67%	46.33%	41.67%	42.67%

Table 5.6: Results for the first set of SFS features

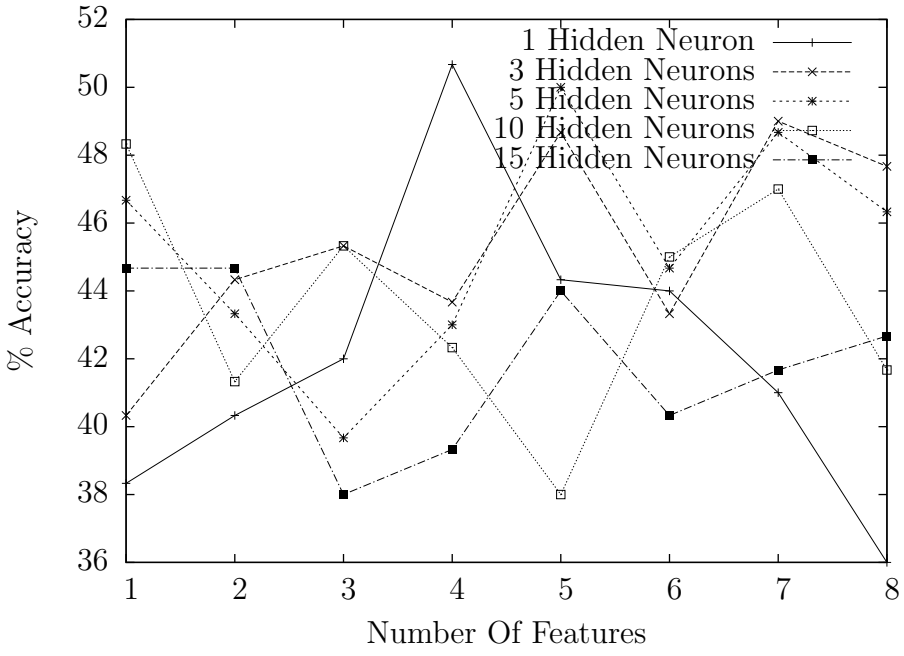

Figure 5.4: Results for the first set of SFS features

Features	$H = 1$	$H = 3$	$H = 5$	$H = 10$	$H = 15$
1	41.00%	43.00%	44.33%	42.67%	44.67%
2	42.00%	47.00%	52.00%	41.33%	45.00%
3	51.67%	52.33%	46.67%	43.00%	46.67%
4	43.33%	47.00%	47.67%	43.00%	48.67%
5	43.00%	44.00%	50.00%	49.67%	41.33%
6	48.67%	49.00%	44.00%	40.00%	43.33%
7	43.00%	51.33%	46.00%	45.00%	39.67%
8	46.33%	47.00%	48.33%	43.00%	36.67%

Table 5.7: Results for the second set of SFS features

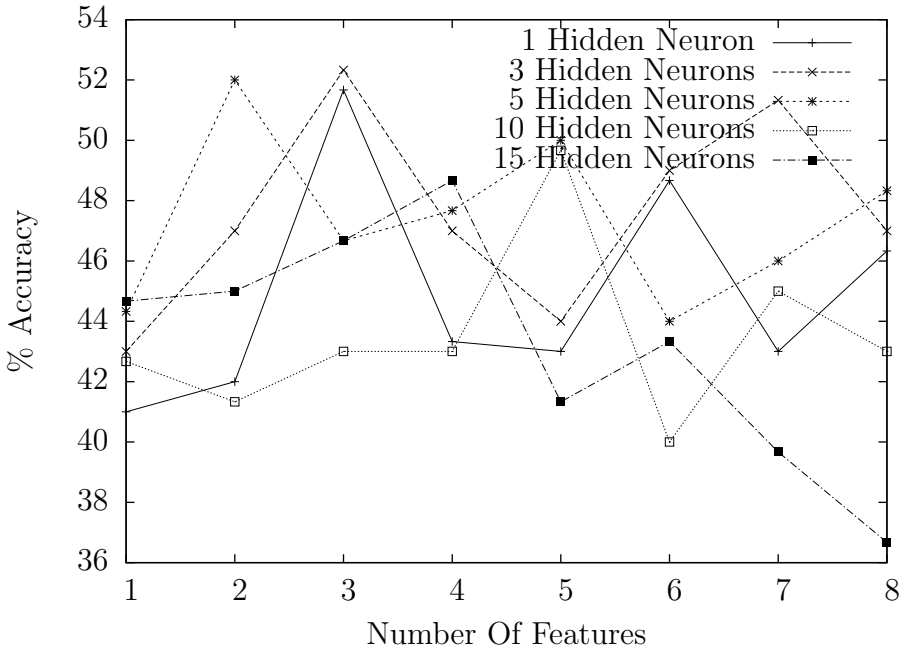

Figure 5.5: Results for the second set of SFS features

	Active	Neutral	Passive
Active	65	14	21
Neutral	49	35	16
Passive	41	9	50

Table 5.8: Confusion matrix for typical neural network experiment with the first two features of the second feature set and 5 hidden layer neurons

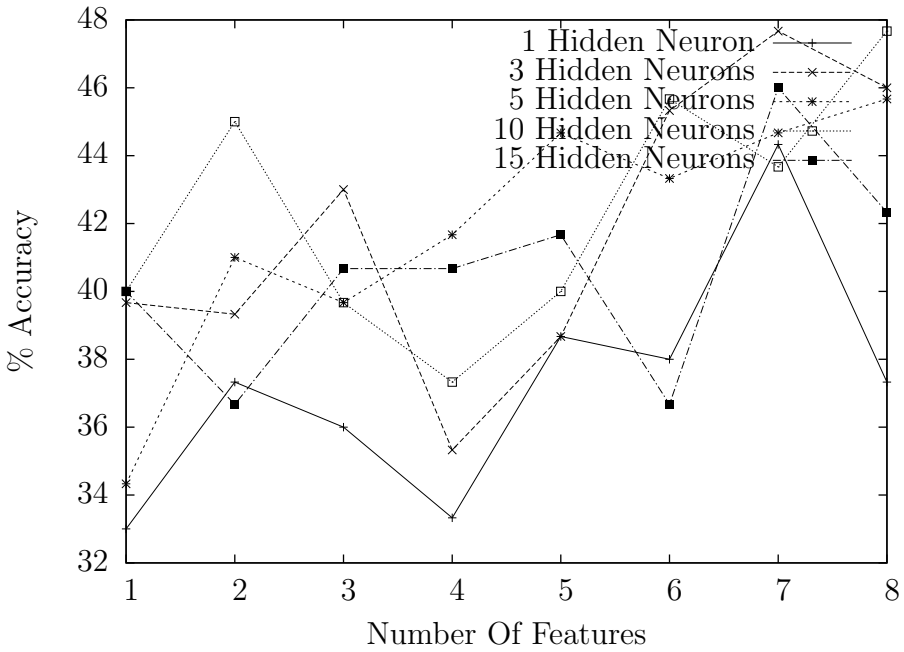

Figure 5.6: Results for the set of mRMR features

Using the mRMR Features

A similar experiment was then performed using the features selected by mRMR in section 3.3.3.

The results of this experiment is presented in table 5.9 and figure 5.6.

Features	$H = 1$	$H = 3$	$H = 5$	$H = 10$	$H = 15$
1	33.00%	39.67%	34.33%	40.00%	40.00%
2	37.33%	39.33%	41.00%	45.00%	36.67%
3	36.00%	43.00%	39.67%	39.67%	40.67%
4	33.33%	35.33%	41.67%	37.33%	40.67%
5	38.67%	38.67%	44.67%	40.00%	41.67%
6	38.00%	45.33%	43.33%	45.67%	36.67%
7	44.33%	47.67%	44.67%	43.67%	46.00%
8	37.33%	46.00%	45.67%	47.67%	42.33%

Table 5.9: Results for the set of mRMR features

The performance of the neural networks with mRMR-selected features seems to be slightly worse than the neural networks trained with the SFS-selected features. This can be seen by comparing table 5.9 to tables 5.6 and 5.7 leads one to conclude that the mRMR-selected features do not distinguish between the features as well as the SFS features does.

5.3.2 Using Only Two Emotional States

Using the SFS Features

As before, a first experiment was performed to determine the optimal number of features to use.

The experiment was performed the same as before: The number of features were increased from 1 to 8, selecting them in their order of importance according to the SFS algoritm, while the number of hidden layer neurons were varied from 1, 3, 5, 10 and 15. The experiment was repeated for both sets of features selected by SFS listed in section 3.3.3 on page 31.

The results of this experiment for the first set of features are presented in table 5.10 and figure 5.7. Table 5.11 and figure 5.8 presents the results for the second set of SFS features.

Features	$H = 1$	$H = 3$	$H = 5$	$H = 10$	$H = 15$
1	62.74%	62.66%	63.63%	64.60%	62.90%
2	64.19%	63.31%	61.61%	62.90%	62.02%
3	64.35%	63.87%	62.74%	60.08%	61.13%
4	63.87%	63.95%	62.82%	61.13%	60.73%
5	65.00%	59.76%	62.42%	60.08%	57.90%
6	62.98%	62.58%	62.26%	59.92%	57.42%
7	63.63%	60.40%	62.42%	60.32%	56.61%
8	63.79%	64.92%	60.65%	59.84%	55.40%

Table 5.10: Results for the first set of SFS features, using only two emotional classes

Features	$H = 1$	$H = 3$	$H = 5$	$H = 10$	$H = 15$
1	63.47%	62.50%	62.50%	63.63%	63.15%
2	64.11%	64.35%	62.82%	61.69%	60.73%
3	64.03%	63.63%	62.66%	62.58%	61.77%
4	65.24%	63.95%	64.03%	60.81%	61.77%
5	64.27%	62.50%	63.95%	62.18%	61.77%
6	64.60%	63.79%	63.55%	60.73%	58.79%
7	63.55%	63.95%	65.40%	59.44%	58.95%
8	63.71%	62.10%	62.98%	58.15%	56.94%

Table 5.11: Results for the second set of SFS features, using only two emotional classes

It is clear that the results have improved significantly from the previous case with classification accuracies in the order of 65% achieved in various combinations of the number of inputs and hidden layer neurons when using the both feature sets.

In both cases it seems as though there is little correlation between the number of features and the classification rate of the neural network when using 1, 3 and 5 hidden layer neurons,

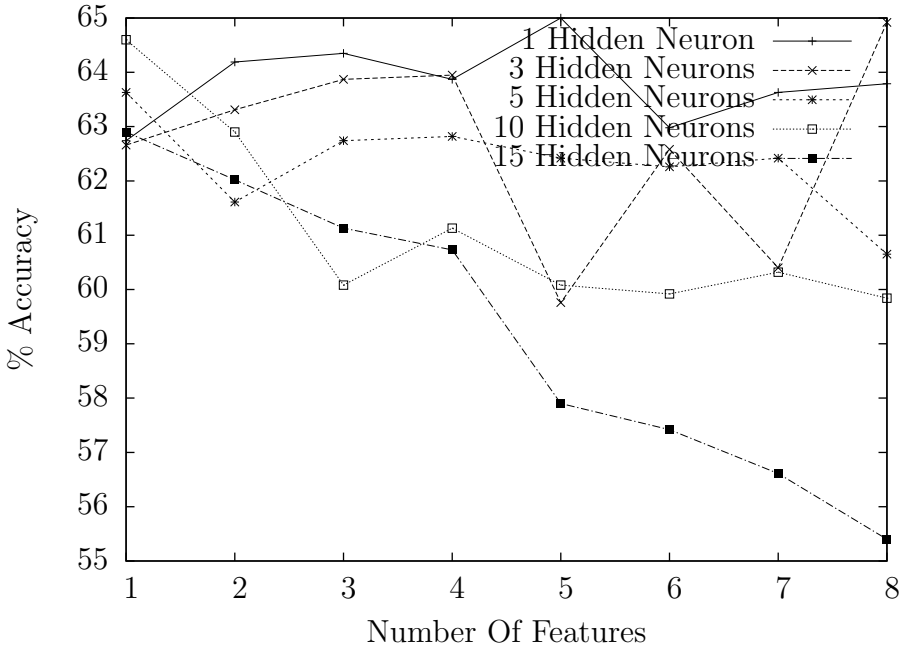

Figure 5.7: Results for the first set of SFS features, using only two emotional classes

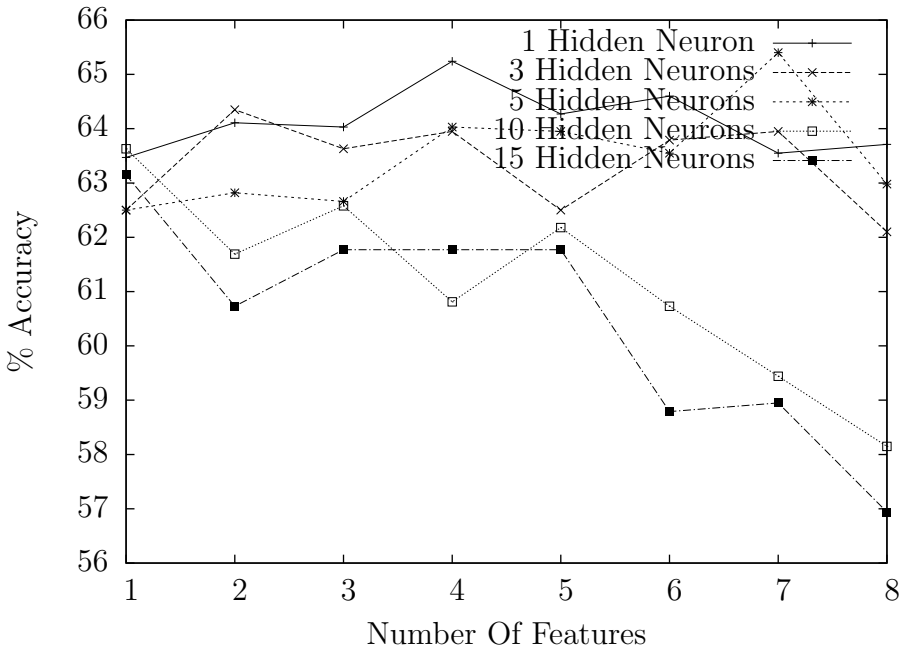

Figure 5.8: Results for the second set of SFS features, using only two emotional classes

but the classification rate obviously decreases as the number of features increases when using 10 and 15 hidden layer neurons.

The best results were 65.40% when using the first seven features of the second feature set and five hidden layer neurons. Table 5.12 shows a confusion matrix for this case.

	Active	Neutral
Active	400	220
Neutral	227	393

Table 5.12: Confusion matrix for typical neural network experiment with the first seven features of the second feature set and five hidden layer neurons

From the confusion matrix in table 5.12 it appears as though the classifcation is fairly symetric: 400 out of 620 active samples were correctly classified and about the same number of neutral samples were correctly classified.

Using the mRMR Features

A similar experiment was then performed using the features selected by mRMR in section 3.3.3.

Features	$H = 1$	$H = 3$	$H = 5$	$H = 10$	$H = 15$
1	59.60%	60.00%	58.71%	56.29%	60.48%
2	59.03%	58.79%	59.44%	58.63%	58.95%
3	61.05%	59.27%	61.13%	58.63%	57.34%
4	58.55%	59.35%	58.95%	56.77%	57.18%
5	60.32%	61.13%	60.48%	57.66%	55.24%
6	59.84%	59.11%	58.06%	56.13%	57.90%
7	60.32%	61.69%	56.61%	54.60%	56.94%
8	62.98%	59.92%	56.85%	57.26%	57.50%

Table 5.13: Results for the set of mRMR features

Again it seems as though the neural networks trained with the SFS-selected features performs better than the neural networks trained with the mRMR selected features.

The best result was nearly 63% classification accuracy with the eight features of the mRMR feature set and a single hidden layer neuron. Table 5.14 shows a confusion matrix for this case.

In this case, a larger percentage of active samples were misclassified, while the number of neutral samples classified correctly remained about the same.

5.3.3 Using Prior Information

The experiment described in section 3.3.4 was performed next. As before, the experiment was performed on an increasing number of features, with 1, 3, 5, 10 and 15 hidden layer neurons

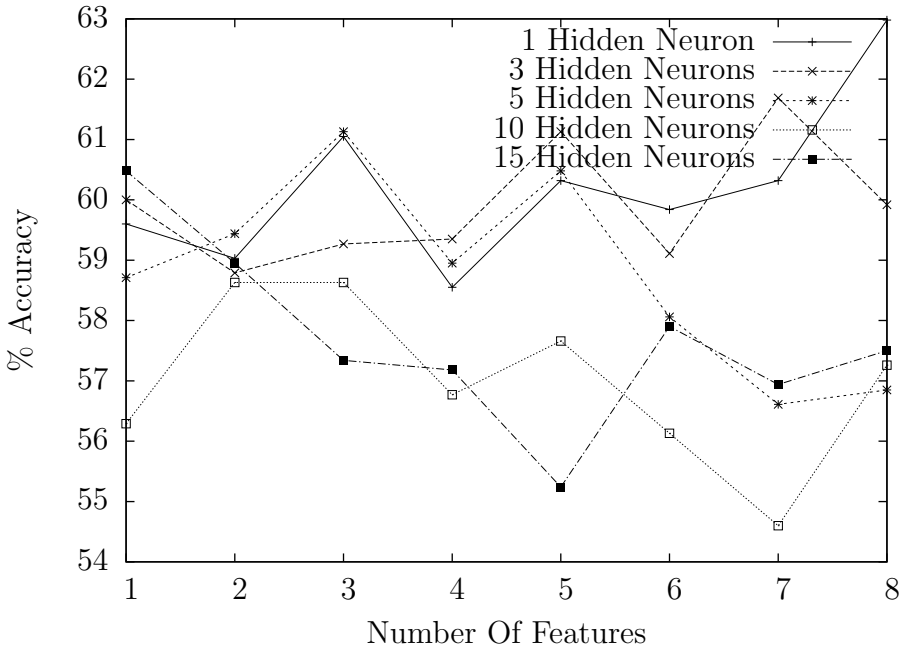

Figure 5.9: Results for the set of mRMR features

	Active	Neutral
Active	354	266
Neutral	217	403

Table 5.14: Confusion matrix for typical neural network experiment with the eight features of the mRMR feature set and one hidden layer neuron

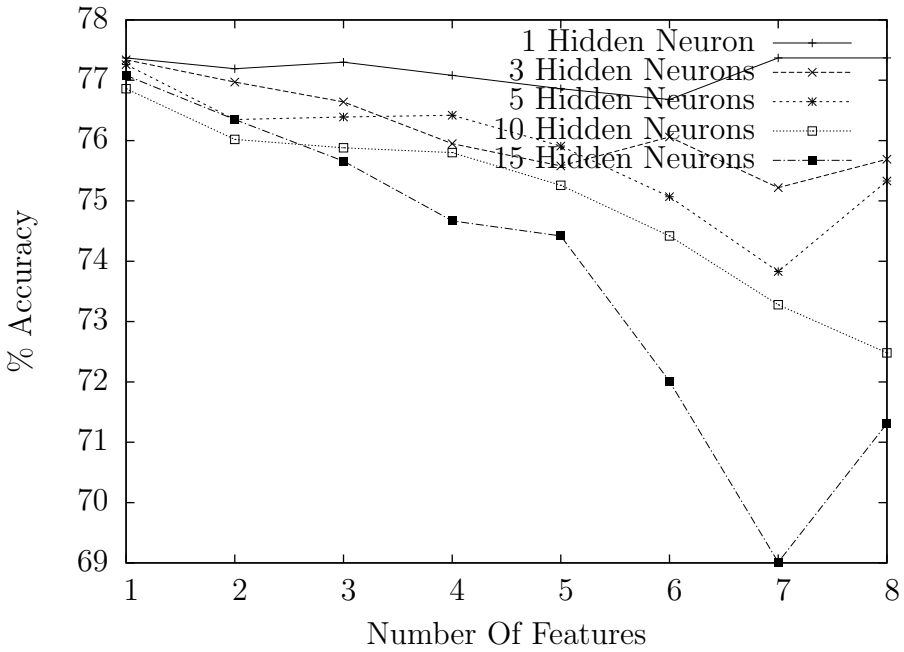

Figure 5.10: Results for the set of mRMR features when prior information is included

at each stage, and cross-validation performed to gauge the performance of each combination of features.

Table 5.15 and figure 5.10 presents the results of this experiment.

Features	$H = 1$	$H = 3$	$H = 5$	$H = 10$	$H = 15$
1	77.37%	77.34%	77.26%	76.86%	77.08%
2	77.19%	76.97%	76.35%	76.02%	76.35%
3	77.30%	76.64%	76.39%	75.88%	75.66%
4	77.08%	75.95%	76.42%	75.80%	74.67%
5	76.86%	75.58%	75.91%	75.26%	74.42%
6	76.68%	76.06%	75.07%	74.42%	72.01%
7	77.37%	75.22%	73.83%	73.28%	69.01%
8	77.37%	75.69%	75.33%	72.48%	71.31%

Table 5.15: Results for the set of mRMR features when prior information is included

The major problem with this approach is that if every sample is classified as neutral, the neural network will have a 73.66% accuracy, but will not be useful for anything, since it is the active emotional samples that a user is interested in.

The confusion matrix for a typical run of the experiment is presented in table 5.16. The particular confusion matrix shown here is the sum of all the cross-validation confusion matrices when 4 features with 5 hidden layer neurons were used.

	Active	Neutral
Active	24	596
Neutral	50	2070

Table 5.16: Confusion matrix for a when prior information is included (4 features, 5 hidden layer neurons)

As can be seen from table 5.16, the neural networks trained to classify most of the samples as neutral. It achieves a 76.42% accuracy by classifying 2666 of the 2740 samples as active.

Only 24 out of the 620 active emotional samples, or 3.87%, were correctly classified. Even more curious is the fact that 50 neutral samples were correctly classified.

Chapter 6

Conclusion

From the preceding chapters, several conclusions can be drawn.

Gender classification turned out to be remarkably effective. Even with only a single feature, accuracy in excess of 87% could be achieved, and using more features or a different number of hidden layer neurons did not improve the result by much.

As shown in section 3.2.3, the features selected by mRMR indicated that the features of the pitch contour are the primary indicators of the gender of the speaker.

The overall performance of the K-Nearest Neighbours method proved to be disappointing.

The best performance was barely 50% accuracy. This corresponds to the 55% achieved by Petrushin [2]'s KNN classifier, but it still misclassifies 1 in every 2 samples.

It can be hypothesised that the performance of the KNN classifier would have improved had more samples been available. The same arguments that support this hypothesis for neural networks hold for the KNN classifier and will be discussed later.

When classifying with all three emotional classes, the performance was roughly comparable to that of the KNN classifier.

It can be concluded that the inability of the neural network to achieve higher accuracy was due to the lack of training samples.

Training the network to classify between all three emotional classes using only 103 samples from each class was clearly inadequate. This can be deducted from the increase in performance when the number of samples of each class was increased to 620 in the next experiment.

Training the network to classify only between active and neutral emotional states led to improved results, but even here the 620 samples of each class seemed inadequate.

Evidence on how well the gender-classifying neural network trained certainly seems to support this hypothesis. The gender classification neural network was trained using a total of 2745 samples, and was able to reach classification rates of 90% in certain conditions.

Unfortunately, by the end of the project, there was insufficient time to obtain more training samples to attempt to rectify this situation.

The addition of prior information at first glance appeared to improve the results, but examining the confusion matrices revealed that the neural networks simply learned to classify most samples as neutral. Since 73% of all the samples are neutral, a neural network that learns to

classify everything as neutral will achieve 73% accuracy, but will not be of much use since the intended user of this project is interested in active emotions.

The results of the experiments where prior information was added therefore have little practical significance except to prove that it is better to remove the bias from the training data.

In the end a 65% classification rate (which corresponds to the best neural networks of sections 3.3.3 and 5.3.2) is still useful when one considers the context in which this project is designed to be used: If 73% of all recordings contain neutral emotions and a listener is interested in the 27% of recordings that contain active content, then approximately only 1 in every 4 recordings will be of interest. If the analyst has a system that can classify recordings according to emotional content with 65% accuracy, then approximately 2 out of every 3 recordings which she listens to become interesting, corresponding to a 41.7% increase in productivity.

6.1 Practical Application

Several worthwhile conclusions can be drawn from the practical implementation of this project.

If a company were to utilise this system, the system would be set up in such a way that Downlode is scheduled to run every night and download recordings from the particular company's database. When it completes SPiL will be executed with a script to look in the local database for newly downloaded recordings and classify them. The results of the analysis can then be presented to the customer service analyst through the graphical user interface in the morning.

The Downlode application has been written specifically for MTN's call centre which uses the NICE system to record call centre conversations. Portions of Downlode will have to be rewritten if the system is to be installed at a company that uses a different platform. The tables within the local database are fairly generic so the other programs will not require any changes.

The software components of the system are portable across platforms. SPiL and the GUI can both be run on Windows and Linux. The exception is the Downlode program which depends on a third-party tool to access the MTN's database and is therefore limited to Windows.

SPiL itself must be released under the terms of the GPL open source license because the PRAAT code that has been incorporated into SPiL is released under this license. This has the implication that the source code to SPiL must be made available to the company where the system is installed.

6.1.1 Scripting

The use of a scripting language proved to be a very good decision. The SPiL program makes it easy to experiment with several different configuration parameters for the neural network and feature sets. This is because changing the program to experiment with different parameters does not require the application to be recompiled.

Scripting in SPiL also allows the same program executable to be used for various different purposes by simply using different scripts as input. For example, the KNN and neural network experiments were all implemented as different Lua scripts input into SPiL. Lua's module system also allows the same scripts to be re-used.

The main advantage of scripting is that it brings the user closer to the problem domain. The Lua bindings allows the SPiL user to program in terms of the problem domain, such as describing how to extract pitch from a recording and classify it through a neural network, rather than worrying about the mundane low level details of memory allocation and file I/O.

Because of the scripting, the SPiL program also allows for the rapid implementation of the various experiments performed during the course of this project. For example, the K-Nearest Neighbours Approach described in section 3.3.2 was implemented entirely as a Lua script. It took an afternoon to get the basic functionality working and thereafter experiments with the various parameters were quickly implemented by altering a line in the script and rerunning SPiL.

6.1.2 Use of and Contribution to Open Source

Using open source software provided several benefits. In the context of this project, the most important benefit was probably cost savings. For instance, instead of investing in expensive database software, SQLite provides a free and functional database engine that supports all the features needed for this project.

Another benefit is that the use of open source software eliminated a significant amount of complex work from the project. For example, while a simple neural network can be easy to implement, implementing a neural network with all the features of FANN would have wasted time while contributing nothing significant to this project.

Another advantage is that all the third party components were mature and well-supported by their communities.

In turn, this project also contributes to open source software. The bindings for the Fast Artificial Neural Network library created for the SPiL program has been donated to the Lua and FANN communities. The Lua bindings for FANN have been converted to a proper open source project named LuaFann and are now hosted by LuaForge at http://luafann.luaforge.net/.

6.2 Contribution

This project enhances understanding about the theoretical and practical aspects of emotion classification. These aspects include the following contributions:

- Promoting an understanding of the issues involved in implementing emotion classification in a telecommunications environment where languages are mixed and sample rates are low. This project builds on work by Martirosian and Barnard [1] to identify several features that can be used to classify a speech recording according to its emotional content.

- Summarizing the general methods by which the problem of emotion classification in speech can be approached and used, including both a K-Nearest Neighbours Approach, described in section 3.3.2 on page 28, and an Artificial Neural Network-Based Approach described in section 3.3.3 on page 29.
- Contributing to an understanding of the issues involved in classifying a recording according to the gender of the speaker. The approach involved using a neural network with several features from the speech signal and is described in section 3.2.3 on page 26.
- Contributing to a suggestion for approaching the problem of identifying segments of speech in a recording, as described in 4.7 on page 50. Although the approach presented there has some shortcomings, it provides a basis for further research.
- Suggesting a practical approach to the creation of a classifier through its use of a scripting language that can train the classifier through one script and use this trained classifier in another script to classify unknown recordings. Both scripts share a third script to extract features from a recording.
- Alerting to the practical issues involved in implementing an emotional classifier. This includes elements such as the following:
  - Accessing the recordings in the call centre database. The approach of the project was to download the recordings to the machine upon which the emotion classification would be done.
  - The interaction between the various components. This project ensured that the various components communicate with each other through a database shared by all the various components.
  - Presenting the results to a user. This project presented its results to the user in a graphical user interface that allows him or her to view the classifier's output as well as listen to the recordings.
  - The programs developed for this project were also reasonably platform independent, supporting both Windows and Linux.
- Because the SPiL program's functionality can be altered by changing its input script, the project also presents a tool that can be useful in a variety of future speech research projects.

6.3 Further Work

As with any similar project, several opportunities for further research have been identified.

The most obvious further research that can be done is to try and improve the accuracy of the neural network approach. In the light of the conclusion above, the apparent way to achieve better results is to obtain more samples.

There are a number of features and feature extraction algorithms that other researchers used, as described in section 2.2, that can be examined in future research. These include the Mel-Frequency Cepstral Coefficients (Wang and Guan [5]) and the Augmented Prosodic Domain and Flattened affect (Cowie and Douglas-Cowie [11]).

The articulatory precision, vocal tract properties, and phonatory quality parameters which were suggested by Tato, Santos, Kompe and Pard [7] may be able to identify the emotion in speech independent of the spoken language. This approach ties in with the goal of developing an emotional classifier that is independent of the language or accent of the speaker.

Schuller, Rigoll and Lang [6] used the AMDF algorithm to extract pitch because the algorithm is less susceptible to noise. This is a desirable property because of the low sample rate of the telephone network.

Some of the other approaches to dimensionality reduction may also yield a better selection of features. These approaches are Promising First Selection ([9] and [3]) and Principal Component Analysis ([9]).

Section 2.3 described several other classification techniques that can be used to attempt to improve the classification accuracy. These include fuzzy methods and hidden Markov models.

The following outstanding issues that can also be addressed in future research:

- Speaker identification. It is desirable to be able to distinguish between the two speakers (agent and customer) that are typically involved in a call centre conversation. The customer may be emotional while the agent may be calm, which may confuse a classifier. Being able to classify the emotion of the two speakers independently may improve the overall accuracy of the system.

- Language identification. Speakers of different languages may articulate emotions differently, and being able to use the language of the speaker as an input to the emotion classifier may improve the classifier performance in the same way that adding the gender information has done.

- The algorithm for determining when speech occurs in the recording described in section 4.7 is somewhat crude and has some shortcomings, such as not being able to distinguish between actual speech and rings and beeps on the telephone line. A more sophisticated algorithm could attempt to use pitch and other information in the waveform to attempt to distinguish between these sounds and actual speech.

Bibliography

[1] O. Martirosian and E. Barnard, "Speech-based emotion detection in a resource-scarce environment," *South African Computer Journal*, pp. 18–22, Jun 2008.

[2] V. A. Petrushin, "Emotion in speech: Recognition and application to call centers," in *Proceedings of the 1999 Conference on Artificial Neural Networks in Engineering*, 1999, pp. 1–4.

[3] F. Dellaert, T. Polzin, and A. Waibel, "Recognizing emotion in speech," in *Proceedings. Fourth International Conference on Spoken Language*, vol. 3. ICSLP, Oct 1996, pp. 1970–1973.

[4] A. Razak, M. Yusof, and R. Komiya, "Towards automatic recognition of emotion in speech," in *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology*. ISSPIT, Dec 2003, pp. 548–551.

[5] Y. Wang and L. Guan, "An investigation of speech-based human emotion recognition," in *IEEE 6th Workshop on Multimedia Signal Processing*, Oct 2004, pp. 15–18.

[6] B. Schuller, G. Rigoll, and M. Lang, "Hidden markov model-based speech emotion recognition," in *Proceedings. 2003 International Conference on Multimedia and Expo*, vol. 1, Jul 2003, pp. 401–404.

[7] R. Tato, R. Santos, R. Kompe, and J. M. Pard, "Emotional space improves emotion recognition," in *Proceedings of ICSLP*, 2002, pp. 2029–2032.

[8] T. S. Polzin and A. H. Waibel, "Detecting emotions in speech," in *Proceedings of the CMC*, 1998, pp. 1–7.

[9] C. Lee, S. Narayanan, and R. Pieraccini, "Recognition of negative emotions from the speech signal," *IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 240–243, Dec 2001.

[10] C. M. Lee and S. Narayanan, "Emotion recognition using a data-driven fuzzy inference system," in *Eurospeech 2003, Geneva*, 2003, pp. 1–4.

[11] R. Cowie and E. Douglas-Cowie, "Automatic statistical analysis of the signal and prosodic signs of emotion in speech," in *Proceedings. Fourth International Conference on Spoken Language*, vol. 3. ICSLP, Oct 1996, pp. 1989–1992.

[12] P. Boersma and D. Weenink, "Praat: doing phonetics by computer [computer program]," 2008, http://www.praat.org/. Last accessed on 15 October 2009.

[13] D. Ververidis and C. Kotropoulos, "Emotional speech recognition: Resources, features and methods," *Speech Communication*, vol. 48, no. 9, pp. 1162–1181, 2006.

[14] C. M. Bishop, *Neural Networks For Pattern Recognition*, 1st ed. Oxford, UK: Oxford University Press, 1995.

[15] C. D. H. Peng and F. Long, "Minimum redundancy maximum relevance feature selection," *IEEE Intelligent Systems*, vol. 20, no. 6, pp. 70–71, 2005.

[16] F. L. H. Peng and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[17] S. Nissen and contributors, "Fast artificial neural network," 2008, . Last accessed on 1 January 2009.

[18] K. Tokuda, T. Masuko, K. Koishida, S. Sako, and H. Zen, "Speech signal processing toolkit (sptk) [computer program]," 2008, http://sp-tk.sourceforge.net/. Last accessed on 1 January 2009.

[19] R. Ierusalimschy, "Programming in lua, second edition," 2006, http://www.inf.puc-rio.br/∼roberto/pil2/. Last accessed on 1 January 2009.

[20] S. Nissen, "Neural networks made simple," 2006, http://fann.sf.net/fann_en.pdf. Last accessed on 1 January 2009.