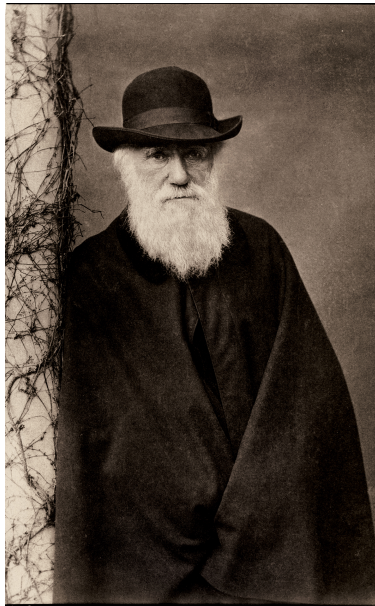


Tomas Pfister
tjp35@cam.ac.uk



EMOTION DETECTION FROM SPEECH

COMPUTER SCIENCE TRIPOS PART II

GONVILLE & CAIUS COLLEGE

2009-2010

The portrait on the cover shows Charles Darwin (1809–1882), who pioneered research in emotions with his book *The Expression of the Emotions in Man and Animals*.

Proforma

Name and College:	Tomas Pfister, Gonville & Caius College
Project Title:	Emotion Detection from Speech
Examination:	Computer Science Tripos 2010
Word Count:	Approx 10,000
Project Originator & Supervisor:	Prof P. Robinson

Original Aim

To implement a real-time classification algorithm for inferring emotions from the non-verbal features of speech. It is expected to successfully extract a set of features and use them to train and detect emotions from speech. As an optional extension, the emotion detector will be applied to perform speech quality assessment.

The underlying theory of machine learning requires an understanding of mathematics in machine learning and optimisation not formally covered in the Computer Science Tripos. A significant amount of time would be devoted to acquire this background knowledge.

Work Completed

Both the core and extension parts of the project are completed. The emotion detector identifies simultaneously occurring emotion states by identifying correlations between emotions and features such as pitch, loudness and energy. Pairwise classifiers are constructed for nine classes from the Mind Reading corpus, yielding an average cross-validation accuracy of 89% for the pairwise machines and 86% for the fused machine. The system achieves real-time performance. The resulting classifier outperforms a recent PhD thesis and a number of papers.

Special Difficulties

None.

Declaration

I, Tomas Pfister of Gonville & Caius College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed:

Date:

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Challenges	3
1.3	Previous work	3
1.4	Reading guide	4
2	Preparation	5
2.1	Theoretical background	5
2.1.1	Support Vector Machines	6
2.1.2	Practically computable version	7
2.1.3	Extended version – mislabelling and kernels	8
2.1.4	Grid search	8
2.1.5	Pairwise fusion	9
2.2	Requirements analysis	9
2.2.1	Priorities, risks and specification	9
2.2.2	Programming language	11
2.2.3	Libraries	11
2.2.4	Corpora	12
2.2.5	Software engineering techniques	13
3	Implementation	15

3.1	Training	17
3.1.1	Preprocessing	17
3.1.2	Feature extraction	20
3.1.3	Feature selection	21
3.1.4	Scaling	23
3.1.5	Grid search	23
3.2	Classification	24
3.2.1	Segmentation	26
3.2.2	Support vector machines	28
3.2.3	Pairwise classification	28
3.2.4	Pairwise fusion mechanism	29
3.2.5	Speech quality assessment	31
3.2.6	Front-end interface	33
4	Evaluation	35
4.1	The system implementation	35
4.1.1	Overall accuracy	35
4.1.2	Real-time Performance	36
4.1.3	Functionality and testing	37
4.2	Classification results	38
4.2.1	Experiment 1: Comparison to previous work	38
4.2.2	Experiment 2: Machine learning algorithms	39
4.2.3	Experiment 3: Grid search for SVM parameter optimisation	41
4.2.4	Experiment 4: Pairwise fusion methods	41
4.2.5	Experiment 5: Speech quality assessment	47
5	Conclusion	49

Bibliography	51
A Sample source code	55
A.1 C++ code	55
A.2 Bash script	57
A.3 Unit tests	58
B Extracted features	59

Acknowledgements

This work could not have been completed without support from my supervisor, Prof P. Robinson.

I also owe a big thank you to the Cambridge-based speech coach company Eloquential, who kindly agreed to label their speech tutoring recordings and allowed them to be used in this project. I also thank Tal Sobol Shikler, who helpfully located some recordings used for evaluating the system. Finally, I wish to thank Sophia Zhang for her encouragement and support.

Chapter 1

Introduction

This dissertation describes the creation of a framework for performing real-time analysis of speech for detecting emotional states and assessing the quality of speech. It shows how the classification is accomplished in real-time and how the system outperforms recent research in speech emotion detection. Both the core and the extension part of the project were completed on schedule.

The core part of this project exploits a balanced mixture of theory and implementation to create small, compact, highly optimised pairwise emotion classifiers. Experiment 1 shows that the framework achieves high classification accuracies that outperform previous research. A closer look at machine learning algorithms is taken in Experiment 2. The results indicate that highly optimised support vector machines achieve higher cross-validation accuracies for emotion detection than other algorithms.

The high improvement in the classification accuracy resulting from using a parameter optimisation algorithm is illustrated in Experiment 3. Finally, the fusion of pairwise classifiers is described in Experiment 4. Results show that using maximum probability as a criterion for classifier fusion achieves highest accuracies for single-label outputs.

The extension part of the project was also completed. Namely, the emotion detector is extended to support speech quality assessment. Experiment 5 shows that a high classification accuracy is achieved.

Section 1.1 explains the key motivations for studying emotions. Section 1.2 describes some of the expected challenges, and Section 1.3 gives a short overview of the previous work in the field. Finally, Section 1.4 gives a brief guide for reading

the rest of this dissertation.

1.1 Motivation

Emotions are fundamental for humans, impacting perception and everyday activities such as communication, learning and decision-making. They are expressed through speech, facial expressions, gestures and other non-verbal clues.

Speech emotion detection refers to analysing vocal behaviour as a marker of affect, with focus on the nonverbal aspects of speech. Its basic assumption is that there is a set of objectively measurable parameters in voice that reflect the affective state a person is currently expressing. This assumption is supported by the fact that most affective states involve physiological reactions which in turn modify the process by which voice is produced. For example, anger often produces changes in respiration and increases muscle tension, influencing the vibration of the vocal folds and vocal tract shape and affecting the acoustic characteristics of the speech [25]. So far, vocal emotion expression has received less attention than the facial equivalent, mirroring the relative emphasis by pioneers such as Charles Darwin.

In the past, emotions were considered to be hard to measure and were consequently not studied by computer scientists. Although the field has recently received an increase in contributions, it remains a new area of study with a number of potential applications. These include emotional hearing aids for people with autism; detection of an angry caller at an automated call centre to transfer to a human; or presentation style adjustment of a computerised e-learning tutor if the student is bored.

A new application of emotion detection proposed in this dissertation is speech tutoring. Especially in persuasive communication, special attention is required to what non-verbal clues the speaker conveys. Untrained speakers often come across as bland, lifeless and colourless. Precisely measuring and analysing the voice is a difficult task and has in the past been entirely subjective. By using a similar approach as for detecting emotions, this report shows that such judgements can be made objective.

1.2 Challenges

This section describes some of the expected challenges in implementing a real-time speech emotion detector.

Firstly, discovering which features are indicative of emotion classes is a difficult task. The key challenge, in emotion detection and in pattern recognition in general, is to maximise the between-class variability whilst minimising the within-class variability so that classes are well separated. However, features indicating different emotional states may be overlapping, and there may be multiple ways of expressing the same emotional state. One strategy is to compute as many features as possible. Optimisation algorithms can then be applied to select the features contributing most to the discrimination while ignoring others, creating a compact emotion code that can be used for classification. This avoids making difficult *a priori* assumptions about which features may be relevant.

Secondly, previous studies indicate that several emotions can occur simultaneously [14]. For example, co-occurring emotions could include being happy at the same time as being tired, or feeling touched, surprised and excited when hearing good news. This requires a classifier that can infer multiple temporally co-occurring emotions.

Thirdly, real-time classification will require choosing and implementing efficient algorithms and data structures.

Despite there existing some working systems, implementations are still seen as challenging and are generally expected to be imperfect and imprecise.

1.3 Previous work

A number of sources were useful in learning about the area and implementing the project. In particular, Sobol Shikler's [27] emotion detector that processes speech samples as batch jobs using pair-wise decision machines has influenced some of the design decisions. This report will show that the emotion detector in this project achieves higher accuracy and does so in real-time.

Significant fundamental work on speech emotion detection was done by Dellaert et al. [6], who proposed the use of statistical pattern recognition techniques for emotion detection and set the basic system architecture still used today. Since then, the accuracy has continually been improved [3, 9, 28], albeit the focus has

been on categorising two to five emotions, in contrast to nine in this work.

The closest implementation to the approach in this project is probably open-EAR [7], a recently released open-source emotion and affect recognition toolkit. Its feature extraction library openSMILE is used in this work as it is one of the most extensive free speech feature extraction libraries. It differs from this project in that it does not support pair-wise machines or grid search for support vector machine parameter optimisation. Moreover, the corpora used in this project are different.

1.4 Reading guide

This project builds upon the following lecture courses:

- Artificial Intelligence I and II
- Information Theory and Coding
- Mathematical Methods for Computer Science
- Digital Signal Processing
- Natural Language Processing
- Programming in C/C++
- Software Engineering
- Algorithms I and II

In addition to these courses, a significant amount of knowledge in machine learning was gained. A summary of the relevant theory is given in Chapter 2.

The next chapters of the dissertation are organised as follows. Chapter 2 describes the preparation undertaken before the emotion detector was implemented. In addition to describing some of the theory required for implementation, it describes the software engineering concepts applied when designing the emotion detector.

Chapter 3 explains how the emotion detector is implemented, including descriptions of some interesting challenges. Chapter 4 evaluates the emotion detector by analysing its performance and comparing its accuracy with previous work. Finally, Chapter 5 presents a concise conclusion.

Chapter 2

Preparation

This chapter illustrates the preparation undertaken before implementing the emotion classifier. Section 2.1 gives a description of the theoretical results upon which the implementation is built.

Section 2.2 discusses the requirements of the emotion classifier, justifying some of the implementation choices and showing evidence that software engineering concepts are thoroughly applied. It outlines the audio features used for classification, including measures such as signal energy, pitch and voice quality. It also justifies the choice of labels for emotion classification (absorbed, excited, interested, joyful, opposed, stressed, sure, thinking and unsure) and for speech quality assessment (clear, credible, competent, dynamic, persuasive and pleasant).

2.1 Theoretical background

This section describes the theory behind the machine learning algorithm used in this work. Firstly, Subsection 2.1.1 describes the Support Vector Machine classification algorithm which lies at the core of the emotion classifier.

Subsection 2.1.2 shows how the classifier can be implemented, and Subsection 2.1.3 shows how the algorithm can be extended to allow mislabelling and the use of different kernels. Finally, Subsections 2.1.4 and 2.1.5 describe how the classifier can be optimised and how multiple classifiers can be fused.

2.1.1 Support Vector Machines

This section defines the classification problem and shows how the Support Vector Machine classifier presents an approximate solution for it.

A classification task, such as classifying the emotional state, involves a set of training data S_{train} and testing data S_{test} , each containing a set of data instances $i \in S$. Each instance i is a tuple (l_i, \mathbf{f}_i) , where $l_i \in \{1, -1\}$ is a class label, with 1 and -1 indicating the class, and $\mathbf{f}_i \in \mathbb{R}^n$ is a set of feature attributes. The goal of classification is to produce a model using S_{train} which predicts the class labels of S_{test} given a set of features \mathbf{f}_i .

In this work, it was found that the Support Vector Machine (SVM) machine learning algorithm produced the highest emotion classification accuracy. In the case of SVMs, we create the model by constructing an N -dimensional hyperplane that optimally separates data into two categories. Optimality is taken to be the maximal separation between the two classes. Any such hyperplane can be written as the set of points $\mathbf{x} = \mathbf{f}_i$ satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0$$

where \mathbf{w} is the normal vector perpendicular to the hyperplane.

We want to choose \mathbf{w} and b such that the distance between two parallel hyperplanes separating the data between the two classes is maximised. These hyperplanes can be described by

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$

$$\mathbf{w} \cdot \mathbf{x} - b = -1$$

with the distance given by $\frac{1}{\|\mathbf{w}\|}$. Thus to maximise the distance between the two planes, we wish to minimise $\|\mathbf{w}\|$, while satisfying the constraints

$$\mathbf{w} \cdot \mathbf{x} - b \geq 1$$

$$\mathbf{w} \cdot \mathbf{x} - b \leq -1$$

for the first and second class respectively, or

$$l_i(\mathbf{w} \cdot \mathbf{x} - b) \geq 1$$

since $l_i \in \{1, -1\}$. These relations are shown graphically in Figure 2.1.

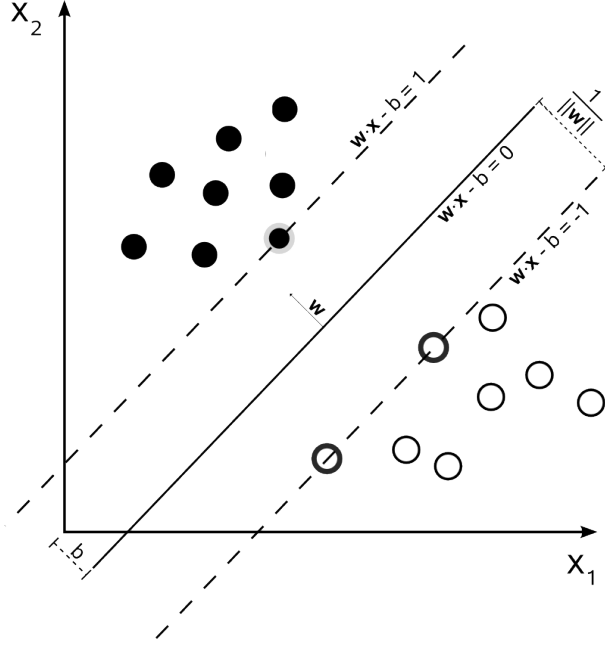


Figure 2.1: Maximum separating hyperplane with margins for two classes.

2.1.2 Practically computable version

This section shows how the computation of an optimal hyperplane reduces to solving a Lagrange multiplier optimisation problem.

The norm $\|\mathbf{w}\|$ is difficult to compute since it involves a square root, so in practice $\|\mathbf{w}\|$ is replaced by $\frac{1}{2}\|\mathbf{w}\|^2$, with factor $\frac{1}{2}$ used for convenience [8]. The solution is derived by solving the Lagrange multiplier optimisation problem

$$\min_{\mathbf{w}, b, \beta} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \beta_i [l_i (\mathbf{w} \cdot \mathbf{x}_i - b) - 1] \right\}. \quad (2.1)$$

where $n = |S_{train}|$ and β_i are Lagrange multipliers.

2.1.3 Extended version – mislabelling and kernels

In this work, an extended version of SVMs that allows for mislabelled examples is used. This Soft Margin [29] approach will choose a hyperplane as cleanly as possible even if there is no hyperplane that can split the two classes. We measure this degree of misclassification by the variable ξ_i and require the solution of the optimisation problem

$$\min_{\mathbf{w}, b, \xi} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \right\} \quad (2.2)$$

under constraints

$$\begin{aligned} l_i(\mathbf{w} \cdot \mathbf{x}_i - b) &\geq 1 - \xi_i \quad 1 \leq i \leq n \\ \xi_i &\geq 0. \end{aligned}$$

where $C > 0$ is the penalty for mislabelled examples. This can be solved using Lagrange multipliers as in Equation 2.1.

Moreover, rather than using a linear classifier, this work will use a non-linear classifier. This replaces the linear dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ by a kernel function that transforms the original input space into a higher-dimensional feature space, allowing the SVM to be non-linear and thus potentially better separate the two classes. After trialling several possible kernel function candidates, including the polynomial $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$, the Radial Basis Function (RBF)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \quad (2.3)$$

with exponentiation coefficient $\gamma > 0$, was found to yield the most promising results. Using a method described in previous research [16], probability estimates are attached to the classification results.

2.1.4 Grid search

When using the Radial Basis Function SVM kernel, choosing the penalty C for mislabelled examples and the exponentiation constant γ in equations 2.2 and 2.3 becomes important. Because the optimal values are model-specific, a search algorithm is needed for finding a near-optimal set of values.

The goal is to identify good (C, γ) values so that the classifier can accurately predict unseen testing data, S_{test} , rather than choosing them to maximise prediction accuracy for the training data, S_{train} , whose labelling is already known. To achieve this, pairs of (C, γ) can be tried sequentially in a range, picking the pair with the highest accuracy on S_{test} .

2.1.5 Pairwise fusion

To generalise SVMs to more than two classes, pairwise classification is used. A single multiclass problem is reduced into multiple binary problems by building a classifier for each pair of classes, using only instances from two classes at a time. The pairwise machines then output the probabilities for the two classes.

In order to determine the most probable class, the probabilities of the multiple binary SVMs need to be fused. A number of ways to fuse the probabilities into a ranked list exist, including for example:

1. Majority voting, where each pairwise decision is counted as a vote for a class.
2. Maximum combined probability, where classes are ranked according to their total probability from the binary SVMs.
3. Votes above a threshold, where classes that receive pairwise votes above a certain threshold are returned.

2.2 Requirements analysis

This section aims to describe the project planning process. Some important tradeoffs and design choices are explained and justified.

2.2.1 Priorities, risks and specification

A list of requirements and their associated risk, priority and difficulty is given in Table 2.1, detailing the work to be done to achieve the goals set in the original project proposal.

The overall system design is shown in Figure 2.2. Brief justifications of the choices made are given in the subsections below.

Component Description	Risk	Priority	Difficulty
Understanding underlying theory	Medium	High	High
Build tools for pre-processing corpora	High	High	Low
Implement SVM training tools	Medium	High	Medium
Implement optimisation algorithms for SVMs	Medium	Medium	High
Implement real-time audio segmentation algorithm	Medium	High	Medium
Implement pairwise classification algorithm	High	High	High
Implement voting algorithms for classification	Medium	Medium	Medium
Build front-end for visualising results	Medium	Low	Medium

Table 2.1: Requirements analysis.

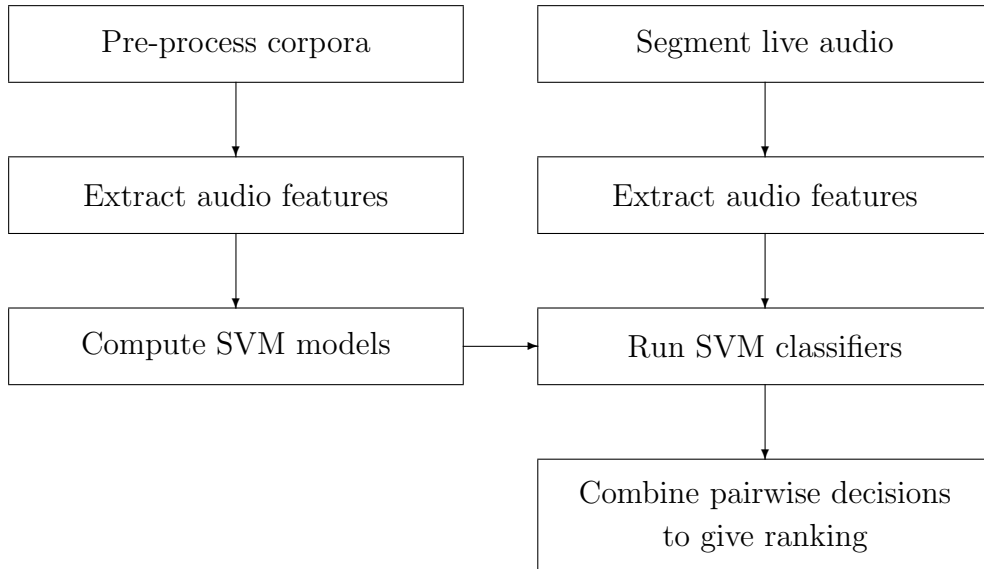


Figure 2.2: Schematic flowchart of the functionality implemented for emotion detection from speech.

2.2.2 Programming language

The program is implemented in C++. This is done to allow interfacing with the openSMILE feature extraction library and the highest performing machine learning libraries (see Section 2.2.3).

2.2.3 Libraries

A set of libraries are used, for machine learning, digital signal processing and graphical user interface development. This section justifies the choices.

Machine learning

In previous work on emotion recognition from speech [27], support vector machines (SVMs) and tree algorithms such as C4.5 have been found to be effective. Other methods such as the Naive Bayesian Classifier and Perceptrons were evaluated using the Weka data mining toolkit [12], but the results were unsatisfactory. Experiment 2 in Section 4.2.2 shows a comparison of C4.5 and SVMs, the two classifiers achieving highest accuracies. SVMs gave the most promising results by a considerable margin. The LibSVM [8] support vector machine library is chosen as it is widely used and well maintained.

Feature extraction

Three feature extraction algorithms were considered:

1. Edinburgh Speech Tools [15]. It is a commonly used open-source C++ speech processing library that provides many feature extraction algorithms. However, the code has very little documentation and lacks modularity. The set of feature extraction algorithms included many irrelevant features used for speech recognition and speech synthesis, and lacked useful features such as pitch strength, voice quality (harmonics to noise ratio), spectral centroid and spectral flux.
2. Matlab [18] with toolkits. By extending the basic functionality of Matlab with the Signal Processing Toolbox and VOICEBOX [5], a large numbers of features could be extracted. However, Matlab did not achieve real-time performance.

3. OpenSMILE [7] is chosen. It is an open-source C++ feature extractor whose algorithms can be run in real-time. The code is freely available and well documented, has a highly modular structure, and provides the capability of doing on-line incremental processing by freely interconnecting feature extractor components. It extracts features such as energy, loudness, pitch, mel-spectra, voice quality, mel-spectrum frequency coefficients, and can calculate various functionals such as means, extremes, peaks, percentiles and deviations. A list of supported features is given in Appendix B.

Graphical user interface

The graphical user interface is written in Qt4. Qt is object-oriented, cross-platform and provides a thorough documentation [20]. For plotting, the Qwt library [24] is used. It provides a 2D plot widget for showing the detection results graphically in real-time.

2.2.4 Corpora

A core part of the project is the choice of training corpora for emotion detection and speech quality analysis. This section justifies the choices and briefly describes the corpora.

For emotion detection, the Mind Reading corpus is used [2]. Speech quality assessment required a new corpus, named Speech Tutor, to be defined as part of this work.

Mind Reading corpus

The Mind Reading corpus is part of the Mind Reading DVD, which is an interactive emotion guide aimed at autistic children. It was developed by psychologists led by Prof Baron-Cohen at University of Cambridge Autism Research Centre, aiming to help autistic children to recognise both basic and complex emotions. The corpus consists of 2927 acted sentences, covering 442 different concepts of emotions, each with 5–7 sentences, in 24 emotion groups. Its number of samples is high and the labelling has been thoroughly evaluated [11]. It also allows comparison of the results with previous work [27].

afraid	angry	bored	bothered ¹	disbelieved
disgusted	excited ²	fond	happy ³	hurt
interested ^{4,5}	kind	liked	romantic	sad
sneaky	sorry	sure ⁶	surprised	think ⁷
touched	unfriendly ⁸	unsure ⁹	wanting	

Table 2.2: The 24 emotion groups in the Mind Reading corpus presented by Baron-Cohen [11]. The superscripts indicate the main groups from which the subset of affective states are selected for this project. These were originally chosen by Sobol Shikler [27], and are used here to allow comparison of the results.

The main emotion groups of Mind Reading are shown in Table 2.2. For the classifier, a subset of 9 categories representing a large variety of emotions is chosen. These subsets are: absorbed, excited, interested, joyful, opposed, stressed, sure, thinking and unsure.

Speech Tutor corpus

Applying machine learning to speech quality assessment is a new idea proposed in this dissertation. As such, there were no labelled samples available for training the system. An experienced speech coach was asked to label 124 one-minute-long samples from 31 different people. The chosen six labels are ones that the professional is accustomed to using when assessing the speech quality. The samples were given a score on a scale 4–10 for each six classes shown in Table 2.3.

clear	competent	credible
dynamic	persuasive	pleasant

Table 2.3: The six speech quality classes chosen for labelling by the speech coach for use with the Speech Tutor corpus.

2.2.5 Software engineering techniques

This subsection gives evidence that the development of the emotion detector complies with sound software engineering principles.

Development models

The program is written using Xcode, which provides an efficient source-code processing interface.

The *Iterative development model* is chosen as the implementation strategy. Initially, a basic prototype of the system in Figure 2.2 is designed, implemented and tested. At consecutive stages increasing layers of complexity are added. Each iteration is thoroughly tested before the next iteration, allowing module-specific debugging. The cross-validation and computation speed are closely monitored at each iteration, allowing both to be maximised.

Unit testing

Modularising the system into separate components allows a set of unit tests to be created for each module. The Bash script for running the tests is given in Appendix A.3.

Redundancy

The Subversion version control system is used to allow going back to a previous revision in case changes need to be undone. Physical redundancy is guarded for by automated hourly backups to an external hard disk, and automated daily backups to PWF and another laptop.

Chapter 3

Implementation

The previous chapter gave a description of some important theoretical ideas upon which the implementation is built and discussed some software engineering choices. This chapter describes how the theory is applied in practice to create a real-time emotion detector that achieves high accuracy. The overall system architecture and the design of the main components are outlined.

All components were successfully implemented, resulting in a fully-working emotion detector and speech quality assessor. An example view of the system is shown in Figure 3.1.

The implementation work can be roughly divided into two parts. Their respective key components to be described in this chapter are:

1. **Training**

1. **Preprocessing**

- Describes the conversion of a single corpus into pairwise corpora, and discusses the chosen noise reduction and corpus splitting solutions.

2. **Feature extraction**

- Presents the openSMILE feature extraction library and summarises how it is integrated into the project.

3. **Feature selection**

- Introduces the feature selection algorithm used to reduce the number of features.

4. **Scaling**

- Briefly discusses the decision to scale the input.

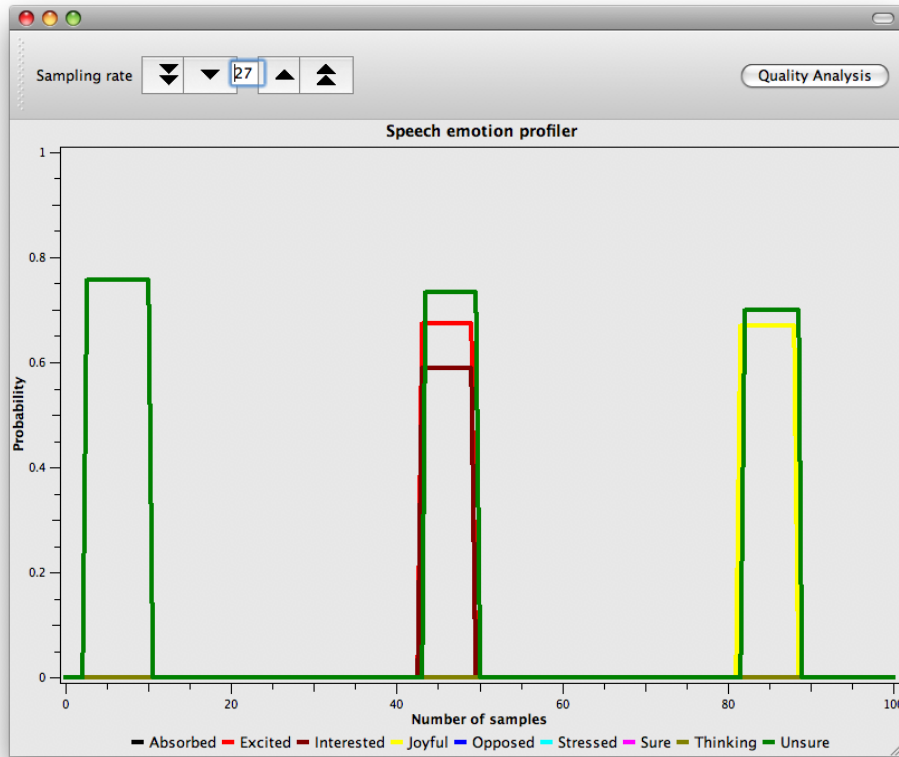


Figure 3.1: Screenshot showing the user interface. The peaks represent the detected emotions with their associated probability. New results appear on the right and move left with a rate of 27 samples per second. In this example, thresholding is used for fusion of the pairwise machines, allowing co-occurring emotions to be detected. A history of 100 time samples is shown. The Quality Analysis button switches the classifier over to do speech quality assessment.

5. Grid search

Describes the grid search algorithm implemented to find the optimal SVM parameters.

2. Classification**1. Segmentation**

Describes the audio segmentation algorithm employed for splitting input audio into samples ready to be processed.

2. SVM model computation

Briefly describes the library used for computing the SVM model and how it is integrated into the project.

3. Pairwise classification

Introduces the pairwise classification algorithm used to combine the outputs from multiple SVMs.

4. Pairwise fusion mechanism

Explains how the pairwise classification algorithm is used to determine the winning labels.

5. Speech quality assessment

Describes how the emotion detector was retrained to perform speech quality assessment for speech tutoring.

6. Front-end interface

Describes the implementation decisions for the front-end interface for visualising the output.

3.1 Training

The training phase of a supervised machine learning algorithm looks for patterns in the features extracted from a labelled corpus and computes models that can later be used for classification. As it can be done as a pre-processing step, runtime performance is not critical. The training system architecture is shown in Figure 3.2. Its main components are discussed below.

3.1.1 Preprocessing

Before training the system, some preprocessing is required. This section describes the work undertaken to prepare the corpora for feature extraction and training.

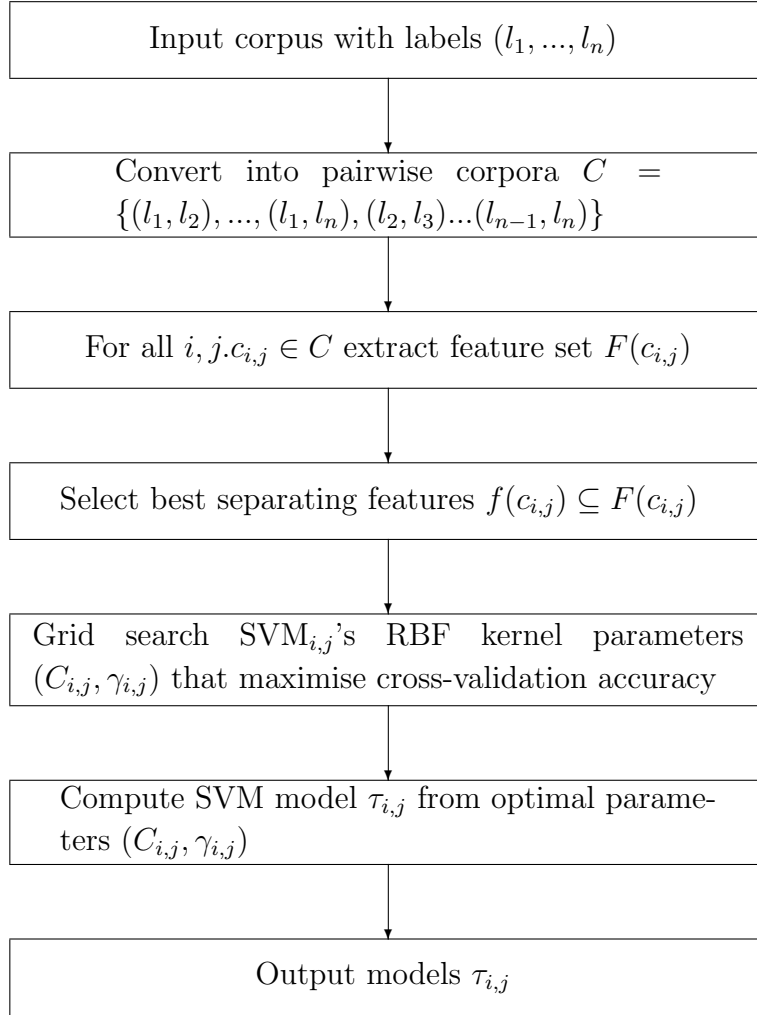


Figure 3.2: The training program architecture. $SVM_{i,j}$ represents the support vector machine for comparing l_i with l_j .

Conversion into pairwise corpora

This section shows how the binary SVM classifier is extended to perform multi-label classification.

A proposed direct multi-class extension of SVMs leads to a very complex optimisation problem [22], and did not yield satisfactory results in a preliminary study. Therefore, the multi-class problem was solved by training several binary SVM classifiers and fusing the outputs to derive a global classification decision.

One of the simplest multi-label classification schemes is the one-against-all approach. In this method, n pairwise classifiers are built in a way that each classifier separates one class from the others. However, it results in very heterogeneous training data sets, leading to poor performance [19].

Previous studies have showed that a more efficient decomposition is pairwise classification [10]. Here a single multiclass problem is reduced into multiple binary problems. A classifier is built for each pair of classes, using only instances from the chosen two classes. Each pairwise machine then outputs the probabilities for the two classes it has compared. Experiments on the corpora in this study confirmed that this was the case, so pairwise classification was chosen for implementation.

Given k classes, pairwise classification creates $\binom{k}{2} = \frac{k!}{2!(k-2)!} = k(k-1)/2$ SVM classifiers. Because each SVM only involves instances of two classes, dividing n instances evenly among k classes given $2n/k$ instances per problem. Although this sounds unnecessarily computationally expensive, suppose that SVM runs in $O(n)$. Then the pairwise execution is

$$O\left(\frac{k(k-1)}{2} \cdot \frac{2n}{k}\right) = O((k-1)n)$$

so the pairwise method scales linearly with the number of classes.

The chosen multilabelling scheme classifies k labels with $k(k-1)/2$ binary SVMs. In the training stage, this required conversion of the corpus with labels (l_1, \dots, l_n) into a set of pairwise corpora $C = \{(l_1, l_2), \dots, (l_1, l_n), (l_2, l_3) \dots (l_{n-1}, l_n)\}$. For this, a script (given in Appendix A.2) is written that generates training corpora for all $\binom{k}{2}$ binary combinations of the classes.

Noise reduction

The Mind Reading corpus used for emotion detection was recorded with high-quality equipment and is noise-free. However, the Speech Tutor corpus was recorded in different environments and contained background noise. To avoid the noise affecting the feature extraction and the construction of the hyperplanes, it was necessary to remove noise from the corpus.

Various noise reduction algorithms, such as Power subtraction [4] and Time frequency block thresholding [30] were tried. The former models the speech signal as a random process to which uncorrelated random noise is added. The noise is measured during a silence period in the speech, and the estimated power spectrum of the noise is then subtracted from the noisy input signal. This method resulted in an artefact which sounds like random musical notes caused by narrowband tonal components that appeared in unvoiced sound and silence regions after the noise reduction.

Time frequency block thresholding dynamically adjusts spectrogram filter parameters using the Stein risk estimator, which gives an indication of the estimator's accuracy. It was found to eliminate the musical noise of the Power subtraction method, and was thus used as a pre-processing filter for the Speech Tutor corpus.

Sample splitting

The Speech Tutor corpus consisted of 1–1.5 min long samples. Since the tutor is to work in real-time and some of the feature extraction algorithms depended on the sample length, the samples are split into sentences by the segmentation algorithm described in Section 3.2.1.

3.1.2 Feature extraction

Feature extraction algorithms are one of the main components of the emotion detection system. For this work, the openSMILE [7] algorithms are used.

OpenSMILE provides sound recording and playback via the open-source PortAudio library, echo cancellation, windowing functions, fast Fourier transforms and autocorrelation. Moreover, it is capable of extracting features such as energy, loudness, pitch, mel-spectra, voice quality, mel-spectrum frequency coefficients, and can calculate various functionals such as means, extremes, peaks, percentiles

and deviations. The full set of features and functionals is given in Appendix B. The set of features to be extracted can be specified in a configuration file.

The ten most commonly used class-differentiating features are shown in Table 3.1. These consist of functionals calculated for two main categories.

The first group consisted of the magnitude of Fourier transforms. The Fourier transform converts the input signal from time domain into frequency domain. This allows analysis of the frequency content of speech. The second common group is Mel-frequency cepstral coefficients (MFCCs). MFCCs are coefficients that collectively make up a Mel-frequency cepstrum (MFC). MFCs represent the short-term power spectrum of sound based on the linear cosine transform of a log power spectrum on a nonlinear mel frequency scale.

Machines	Feature name	Feature group
12	mfcc_sma[12]_range	Mel-frequency cepstral coefficients
9	pcm_Mag_fband250-650_sma_de.centroid	Fast Fourier Transform (FFT) magnitude for a frequency band
8	pcm_Mag_melspec_sma_de.de[4]_quartile3	FFT magnitude Mel spectrum
8	mfcc_sma_de.de[4]_qregerrQ	Mel-frequency cepstral coefficients
8	mfcc_sma[8]_range	Mel-frequency cepstral coefficients
7	pcm_Mag_melspec_sma_de[2]_quartile2	FFT magnitude Mel spectrum
7	mfcc_sma[4]_minameandist	Mel-frequency cepstral coefficients
7	mfcc_sma[4]_iqr1-2	Mel-frequency cepstral coefficients
6	pcm_Mag_melspec_sma_de.de[4]_iqr2-3	FFT magnitude Mel spectrum
6	pcm_Mag_melspec_sma_de.de[19]_minPos	FFT magnitude Mel spectrum

Table 3.1: Ten most used features in emotion detection. The first column shows the number of pairwise machines in the total $\binom{9}{2} = 36$ that used the feature.

3.1.3 Feature selection

Since a large feature set is to be extracted from the speech, it is expected that there would be some irrelevant and redundant data that would not improve the SVM prediction performance. It is known that classification algorithms are unable to attain high classification accuracy if there is a large number of weakly relevant and redundant features, a problem known as *the curse of dimensionality* [1]. Algorithms also suffer from computational load incurred by the high

dimensional data. Hence, a feature selection algorithm is used to reduce the number of features and to remove redundant and irrelevant data [17].

One strategy is to compute as many features as possible. Optimisation algorithms can then be applied to select the most differentiating features and reduce their number. This avoids making difficult *a priori* decisions about which features may be relevant.

The approach taken is to use all openSMILE feature extractors, and then pick the most relevant ones using feature selection. For this, the Correlation-based Feature Selection (CFS) algorithm [13] is used. It uses a heuristic based on the assumption that good feature sets contain features highly correlated with the class and uncorrelated with each other. It defines the score for a feature subset S as

$$Merit_S = \frac{kr_{cf}}{\sqrt{k + k(k-1)r_{ff}}}$$

where k is the number of features in S , r_{ff} is the average feature-feature intercorrelation and r_{cf} is the average feature-class correlation. After discretising the features, CFS calculates feature-class and feature-feature correlations using a symmetric form of information gain

$$H_{sym}(X; Y) = \frac{2I(X; Y)}{H(X) + H(Y)}$$

for random variables X, Y where

$$\begin{aligned} H(X) &= - \sum_{x \in X} p(x) \log_2 p(x) \\ H(X|Y) &= - \sum_{y \in Y} p(y) \sum_{x \in X} p(x|y) \log_2 p(x|y) \end{aligned}$$

with $H(X)$ representing the entropy of X and $H(X|Y)$ the entropy of X after observing Y . The information gain is

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

$H_{sym}(X; Y)$ fixes the problem of $I(X; Y)$ assigning a higher value for features with a greater number of values although they may be less informative. It also normalises the values to the range $[0, 1]$.

After calculating the correlations, CFS starts with an empty set of features and applies forward best first search, terminating when it encounters five consecutive fully-expanded subsets that show no improvement.

3.1.4 Scaling

After the feature selection stage, the features $f(c_{i,j}) \subseteq F(c_{i,j})$ are scaled from \mathbb{R} to $[-1, +1]$. This is done to

1. Prohibit attributes in a greater numeric range from dominating those in smaller ranges.
2. Avoid numerical difficulties during the calculation, such as division by large $f(c_{i,j})$.

3.1.5 Grid search

As noted in Section 2.1.3, when using the Radial Basis Function SVM kernel, it is important to choose a suitable penalty for mislabelled examples C and the exponentiation constant γ . Because the optimal values are model-specific, a search algorithm is needed for finding a near-optimal set of values.

The goal is to identify good (C, γ) values so that the classifier can accurately predict unseen testing data, rather than choosing them to maximise prediction accuracy for the training data whose labelling is already known. A common approach for this is to divide the training data into two parts, one for training and the other for testing. This allows the optimised accuracy to more precisely reflect the prediction accuracy on unknown data.

In this work, v -fold cross-validation is used. The training set is divided into v equal-sized subsets, with each subset sequentially tested using a classifier trained on the remaining $v - 1$ subsets. Every subset is predicted once, so the cross-validation accuracy is the percentage of data which are correctly classified.

The GRID-SEARCH algorithm (Algorithm 1) sequentially tries pairs of (C, γ) in a given range, and picks the one with the highest cross-validation accuracy. Exponentially growing sequences were found to work well in practice, as suggested in previous research [23]. The algorithm is then run recursively on a shrinking area, with the number of recursions set by the initial value of n_{step} .

As will be discussed in more detail in the evaluation chapter, grid search considerably improved the emotion detection accuracy rates.

Algorithm 1 Grid search algorithm. The algorithm is run by giving the mislabelling penalty C and exponentiation constant γ ranges initial values. Variable n_{steps} defines how fine-grained the grid search is.

GRID-SEARCH($[C_{low}, C_{high}, C_{step}], [\gamma_{low}, \gamma_{high}, \gamma_{steps}], n_{steps}$)

1. Initialise $P = 0$, $C_{opt} = 0$, $\gamma_{opt} = 0$.
 2. For all $i \in \mathbb{Z}^+ \cup \{0\}$ such that $C_i = (C_{low} + iC_{step}) \leq C_{high}$
 1. For all $j \in \mathbb{Z}^+ \cup \{0\}$ such that $\gamma_j = (\gamma_{low} + j\gamma_{step}) \leq \gamma_{high}$
 1. Divide training model into v equal subsets (for predefined v).
 2. Sequentially classify one subset using a classifier trained on the remaining $v - 1$ subsets using parameters $C = 2^{C_i}$ and $\gamma = 2^{\gamma_j}$.
 3. If $P < (n_{correct}/v)$, set $P = (n_{correct}/v)$, $C_{opt} = C_i$, $\gamma_{opt} = \gamma_j$.
 3. If $n_{steps} \leq 0$, return (C_{opt}, γ_{opt}) .
 4. Else GRID-SEARCH($[C_{opt} - n_{steps}, C_{opt} + n_{steps}, C_{step}/2], [\gamma_{opt} - n_{steps}, \gamma_{opt} + n_{steps}, \gamma_{step}/2], n_{steps} - 1$).
-

Once optimal (C, γ) are determined, final SVM models are computed for all pairwise corpora using the LibSVM [8] library.

3.2 Classification

The classification phase of a supervised machine learning algorithm uses the models computed in the training phase to predict labels for unseen samples. Run-time performance in this phase is critical for providing a real-time system. The system architecture for the real-time classifier is shown in Figure 3.3. Its main components are described below.

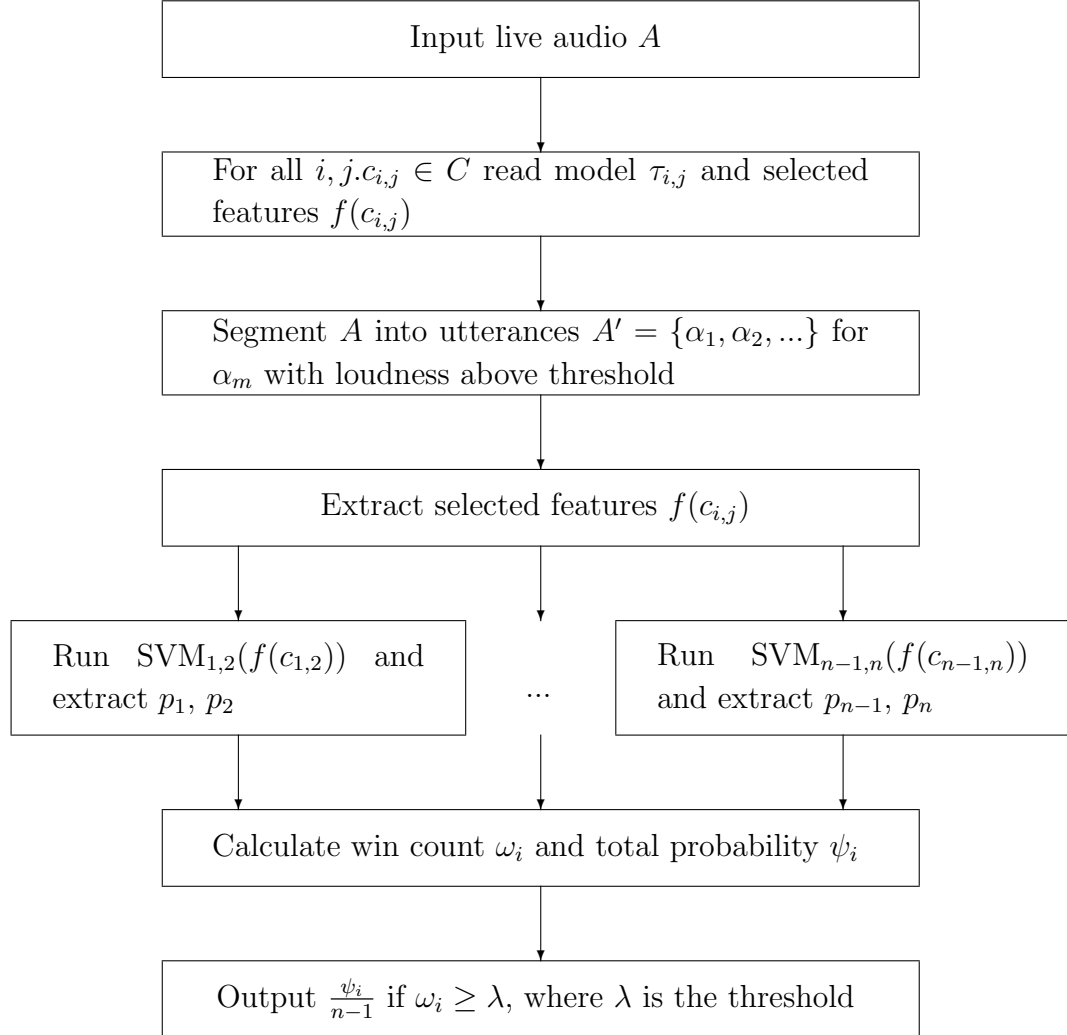


Figure 3.3: The classifier architecture. $SVM_{i,j}$ computes the probabilities p_i and p_j for labels i, j , using features $f(c_{i,j})$.

3.2.1 Segmentation

Before the audio can be classified it needs to be segmented into pieces. Features can then be extracted, mapped into an N -dimensional space and separated by a hyperplane.

One approach to the problem would be to just process every 1 second separately. However, this would create two problems.

1. The single second may only contain silence, but the SVMs will still need to make a binary decision between two classes by mapping the features on one side of the hyperplane.
2. Emotions may not be expressible within a short time interval. Furthermore, some features have a temporal characteristic which will not be extractable if a single segment length is used.

As a result it was necessary to choose the segment length dynamically, approximating to one segment per sentence. The signal energy could be used for differentiating between silence and speech. The choice was between a simple algorithm that uses static thresholds and a more complex algorithm that implements dynamic thresholding. In accordance with the iterative development model, a simple static algorithm was implemented first. By adding complexity in layers, it could at each step be checked that performance sufficient for real-time operation is achieved.

The SEGMENTATION algorithm (Algorithm 2) achieves this by defining three thresholds. First, the silence threshold η defines the threshold for the energy $E = \sum_i^n |s_i|^2 > \eta$, for signals s_i in frame of size n . Second, ρ_{start} sets the number of frames with energy above η that are required until a segment start is detected. Third, ρ_{end} is the number of frames below η until a segment end is detected. From the start to the end of a segment, the features are extracted. At the end the pairwise SVMs are run in parallel.

It turned out that the segmentation results of this algorithm were more than sufficient for detecting pauses in the speech. In practice, separate η could be used for a quiet room and for a noisier environment.

Algorithm 2 Audio segmentation algorithm. η is the silence threshold based on the energy of the signal. ρ_{start} and ρ_{end} specify the thresholds for the number of frames that need to be above and below η for a segment to start and end, respectively.

SEGMENTATION($\eta, \rho_{start}, \rho_{end}$)

1. Initialise $C_{start} = C_{end} = 0$
 2. Repeat
 1. If $\sum_i^n |s_i|^2 > \eta$
 1. Set $C_{end} = 0$
 2. Increment C_{start}
 3. If $C_{start} > \rho_{start}$
 1. Send **start_inference** message to all pairwise SVMs.
 2. Else
 1. Set $C_{start} = 0$
 2. Increment C_{end}
 3. If $C_{end} > \rho_{end}$
 1. Send **end_inference** message to all pairwise SVMs.
-

3.2.2 Support vector machines

Once the audio is segmented and the features are extracted, the $n(n - 1)/2$ pairwise machines can be run in parallel to predict the class for a segment. The hyperplanes separate two emotion classes in an $|f|$ -dimensional space, where f is the set of features being considered. Implementation-wise, this required preparing the features for interfacing with the LibSVM library and then using the results to do further processing. The LibSVM library provides a highly optimised algorithm for solving the Lagrange multiplier optimisation problem in Equation 2.2.

The RUN-SVM algorithm (Algorithm 3) describes this process. First, all but the features selected by the correlation-based feature selection algorithm needed to be disabled. The algorithm then waits for extracted features from openSMILE. Once it receives these, it scales them by the same ratio as used in the training phase. It then predicts probabilities for both labels by calling the LibSVM C++ library, and sends the results paired with the labels to the PAIRWISE-COMBINATOR module.

Algorithm 3 Run-SVM algorithm. τ is the model file, (l_i, l_j) are the pairwise class labels, S is the scaling file and ζ is the feature selection file. These are computed in the training phase.

RUN-SVM($\tau, (l_i, l_j), S, \zeta$)

1. Load τ , (l_i, l_j) , S and ζ from files
 2. Filter away all features except ζ
 3. Repeat
 1. Receive features $f_i \in \zeta$ from openSMILE components
 2. Apply scaling S used in training
 3. Predict probabilities (p_i, p_j) by applying LibSVM on (τ, f)
 4. Send $[(l_i, p_i), (l_j, p_j)]$ to PAIRWISE-COMBINATOR
-

3.2.3 Pairwise classification

When the pairwise SVMs have been run, their results need to be combined so that the label of the segment can be predicted. This glues together the SVMs

running in parallel and the voting algorithm.

The PAIRWISE-COMBINATOR algorithm (Algorithm 4) achieves this by receiving the results from the $\frac{1}{2}n(n-1)$ pairwise SVMs running in parallel. It keeps count of the labels of the winning classes. Once it has received all pairwise classification results for a segment, it runs a pairwise voting algorithm that determines the winning class. The voting is described in the next section.

Algorithm 4 Pairwise combinator algorithm. n is the number of labels. As there are $\frac{1}{2}n(n-1)$ pairwise machines, each label should receive $n-1$ probabilities.

PAIRWISE-COMBINATOR(n)

1. Initialise $P = \{S_1, \dots, S_n\}$ with $\forall i. S_i = \{\}$, $W = \{\omega_1, \dots, \omega_n\}$ with $\forall i. \omega_i = 0$
 2. Repeat
 1. If $\forall i. |S_i| < n-1$
 1. Receive $[(l_i, p_i), (l_j, p_j)]$ from RUN-SVM
 2. Insert p_i into S_i and p_j into S_j
 2. Else
 1. For all i set $\omega_i = \sum_{p \in S_i} g(p)$ where $g(p) = \begin{cases} 1 & \text{for } p \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$
 2. Run PAIRWISE-VOTING(P, W) and set $\forall i. S_i = \{\} \wedge \omega_i = 0$
-

3.2.4 Pairwise fusion mechanism

In order to determine the most probable class, the probabilities of the multiple binary classifiers need to be fused.

Studies in psychology indicate that several emotions can occur simultaneously [14]. Examples of co-occurring emotions include being happy at the same time as being tired, or feeling touched, surprised and excited when hearing good news. This section proposes a fusion method for determining co-occurring emotions. Whereas in traditional single-label classification a sample is associated with a single label l_i from a set of disjoint labels L , multi-label classification associates each sample with a set of labels $L' \subseteq L$. A previous study concluded that the use of complex non-linear fusion methods yielded only marginal benefits

(0.3%) over linear methods when used with SVMs [22]. Therefore, three linear fusion methods are implemented:

1. Majority voting using wins from binary classifiers.
2. Maximum combined probability from binary classifiers.
3. Binary classification wins above a threshold.

In the first method we consider all $n - 1$ SVM outputs per class as votes and select the class with most votes. Assuming that the classes are mutually exclusive, the *a posteriori* probability for feature vector \mathbf{f} is $p_i = P(\mathbf{f} \in \text{class}_i)$. The classifier $\text{SVM}_{i,j}$ computes an estimate $\hat{p}_{i,j}$ of the binary decision probability

$$p_{i,j} = P(\mathbf{f} \in \text{class}_i | \mathbf{f} \in \text{class}_i \cup \text{class}_j)$$

between classes i and j . The final classification decision \hat{D}_{voting} is the class i for which

$$\hat{D}_{\text{voting}} = \arg \max_{1 \leq i \leq n} \sum_{j \neq i} g(\hat{p}_{i,j})$$

where

$$g(p) = \begin{cases} 1 & \text{for } p \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

as computed by Algorithm 4. Ties are solved by declaring the class with higher probability to be the winner.

In the second method, the maximum probability $\psi_i = \sum_{p \in S_i} p$ of the binary SVMs is determined. The winner of decision $\hat{D}_{\text{probability}}$ is i such that

$$\hat{D}_{\text{probability}} = \arg \max_{1 \leq i \leq n} \sum_{j \neq i} \hat{p}_{i,j}.$$

Finally, for detecting co-occurring emotions, the classes are ranked according to the number of wins. The classes with wins above a threshold λ are returned, with the classification decision $\hat{D}_{\text{threshold}}$ being set of classes

$$\hat{D}_{threshold} = \{i | \sum_{j \neq i} g(\hat{p}_{i,j}) \geq \lambda\}.$$

The PAIRWISE-VOTING algorithm (Algorithm 5) combines these three approaches. An experiment comparing the results using the three methods is detailed in the evaluation section. The total wins and probabilities are calculated for each label l_i . The resulting label and its average probability over the $n - 1$ SVMs is sent to the GUI over TCP. Sample C++ code for voting is shown in Appendix A.1.

Algorithm 5 Pairwise voting algorithm. μ is the mean win count and σ is the standard deviation. The threshold definition follows previous research [27].

PAIRWISE-VOTING(P, W)

1. Set win threshold $\lambda = \lfloor (\mu + \sigma)(n - 1) \rfloor$
 2. For each l_i with $i \leq n$ using $p_i \in S_i$, $S_i \in P$ and $\omega_i \in W$
 1. Set the total probability $\psi_i = \sum_{p \in S_i} p$ with $norm(\psi_i) = \frac{\psi_i}{n-1}$
 2. If $\omega_i \geq \lambda$
 1. Send $(l_i, norm(\psi_i))$ to GUI
-

To allow for comparison with previous research [27], the threshold is set to $\lambda = \lfloor (\mu + \sigma)(n - 1) \rfloor$. By the central limit theorem, the distribution of a sum of many independent, identically distributed random variables (RVs) tends towards the normal distribution. By assuming that the SVMs exhibit such RVs, and since for the normal distribution $\mu + \sigma \approx 0.841$, $\lambda = \lfloor 0.841(n - 1) \rfloor$. In particular, for the 9 classes chosen for evaluation, $\lambda = 6$. An example console output for the PAIRWISE-VOTING algorithm is shown in Figure 3.4.

3.2.5 Speech quality assessment

For assessing speech quality, the classifier is retrained using six labels describing speech quality given in Table 2.3. This is a novel application of the classifier that has not been attempted before.

Following the speech classifier requirements set by Schuller et al. [26], the application uses non-acted, non-prompted, realistic data with many speakers, using all

```

tomas@tp: ~/backup — emoDetect — 120x26
unsure: 0.112922 0.223908 0.230791 0.673759 0.878396 0.984226 0.970931 0.985508 (div 8) = 0.622555 wins: 5
maxWins: thinking - 8 maxProbability: thinking - 0.910499

absorbed: 0.057484 0.085010 0.098221 0.197898 0.220630 0.339086 0.534186 0.582317 (div 8) = 0.264354 wins: 2
excited: 0.223919 0.293604 0.380831 0.419890 0.581953 0.779370 0.867084 0.937467 (div 8) = 0.560515 wins: 4
interested: 0.062533 0.113507 0.156481 0.214882 0.249710 0.721134 0.829272 0.942516 (div 8) = 0.411254 wins: 3
joyful: 0.049951 0.267370 0.465814 0.598609 0.776081 0.804238 0.886493 0.889573 (div 8) = 0.592266 wins: 5
opposed: 0.019315 0.080429 0.132916 0.138098 0.170728 0.401391 0.568836 0.660914 (div 8) = 0.271579 wins: 2
stressed: 0.195762 0.199376 0.546699 0.619169 0.750290 0.914990 0.919571 0.972414 (div 8) = 0.639784 wins: 6
sure: 0.010872 0.027586 0.110427 0.181773 0.278866 0.418047 0.431164 0.901779 (div 8) = 0.295064 wins: 1
thinking: 0.380537 0.453381 0.580110 0.802102 0.818227 0.843519 0.950049 0.980685 (div 8) = 0.726066 wins: 6
unsure: 0.417683 0.619463 0.706396 0.732630 0.785118 0.800624 0.861902 0.989128 (div 8) = 0.739118 wins: 7
maxWins: unsure - 7 maxProbability: unsure - 0.739118

absorbed: 0.009036 0.047265 0.190438 0.234540 0.234952 0.334619 0.337664 0.338990 (div 8) = 0.215938 wins: 0
excited: 0.370698 0.521801 0.723838 0.745676 0.765460 0.850628 0.910699 0.997974 (div 8) = 0.735847 wins: 7
interested: 0.254324 0.273853 0.274348 0.297709 0.304996 0.480713 0.606813 0.952735 (div 8) = 0.430686 wins: 2
joyful: 0.083980 0.149372 0.485090 0.540309 0.661010 0.702291 0.787590 0.803608 (div 8) = 0.526656 wins: 5
opposed: 0.002026 0.013795 0.022732 0.070858 0.212410 0.327587 0.726147 0.990964 (div 8) = 0.295815 wins: 2
stressed: 0.089301 0.196392 0.273735 0.367275 0.393187 0.662336 0.672413 0.861100 (div 8) = 0.439467 wins: 3
sure: 0.038349 0.138900 0.148010 0.276162 0.514910 0.519287 0.809562 0.929142 (div 8) = 0.421790 wins: 4
thinking: 0.478199 0.725652 0.726265 0.765048 0.787496 0.851990 0.916020 0.977268 (div 8) = 0.778492 wins: 7
unsure: 0.212504 0.459691 0.629302 0.632725 0.665381 0.695004 0.961651 0.986205 (div 8) = 0.655308 wins: 6
maxWins: thinking - 7 maxProbability: thinking - 0.778492

```

Figure 3.4: Screenshot showing the console debugging output for the fusion of the pairwise machines. The PAIRWISE-COMBINATOR algorithm collects the probabilities of the pairwise SVMs and runs PAIRWISE-VOTING once it has received the outputs from all $\frac{1}{2}n(n-1) = 36$ pairwise machines, with $n = 9$. The probabilities from the pairwise SVMs are shown for the 9 labels. The values on the right of the equal signs are the normalised total probability and the win count for each class. The line below the class-wise results shows the winners when using voting or maximum probability for pairwise fusion.

obtained data. An experienced speech coach was asked to label 124 one-minute-long samples of natural audio from 31 people attending speech coaching sessions. The chosen six labels are the ones that the professional is accustomed to using when assessing the public speaking skills of clients. The samples are labelled on a scale 4–10 for each class. The samples of classes are divided into higher and lower halves according to the score. The upper half represents a positive detection of the class (e.g. *clear*), and the lower half represents a negative detection (e.g. *not clear*).

One binary SVM per class is used to derive a class-wise probability. If a pair-wise approach similar to that in emotion classification had been used, the same samples would have existed in several classes, making separating the classes intractable. As a result, unlike in emotion detection where the most prominent labels describing the speech are selected, for speech quality assessment all classes are detected, each labelled with a probability. This allows users to attempt to maximise all class probabilities, a goal which is more useful for speech coaching.

The Speech Tutor corpus consisted of natural audio as opposed to the acted audio used for emotion detection. This created two new challenges.

Firstly, unambiguous speech qualities are often apparent in only a small portion of a real corpus, and may thus provide too infrequent data for providing a basis for consistent annotation and modelling using fine-grained labels. In this study, a local speech coach was asked to annotate the corpus according to her professional judgement. However, for some labels the corpus did not present enough variation to allow for useful labelling, with all samples being scored 9 or 10 on a scale 4–10.

Secondly, there is varying levels of background noise as the corpus was recorded in multiple locations at varying times, causing some samples to be correlated due to similar noise rather than similar non-verbal speech. This is solved using the noise reduction approach described in Section 3.1.1.

3.2.6 Front-end interface

A simple front-end interface is built to allow results to be visualised by the user and debugged during development. The results are presented graphically using Qt4 together with the Qwt Qt widget library. The resulting user interface is shown in Figure 3.1.

To maximise the flexibility for the use-cases, the communication between the back-end and the front-end is encapsulated within TCP packets. This allows the

recording machine to be separate from viewer. For a higher number of classes, computation can be done on a server with the results presented in real-time on a lightweight device such as a mobile phone.

The main challenge was to separate a TCP server into a separate thread from the drawing, allowing real-time graphing of the results. As expected, the Qwt documentation was also lacking, so time was spent reading its source code. However, the time provided to be well spent, as once Qwt's quirks were understood, the resulting GUI worked smoothly.

Chapter 4

Evaluation

The project is evaluated from two perspectives. Firstly, the system implementation is evaluated in terms of how well it achieves the original overall goals. This is discussed in Section 4.1. Secondly, a quantitative analysis of the class detection accuracies and computation performance of the system is presented in a number of experiments in Section 4.2.

4.1 The system implementation

This section evaluates the training and classification system from the developer's point of view. The system is examined in terms of its accuracy, performance and functionality.

4.1.1 Overall accuracy

The success criteria defined in the project proposal set three main goals:

1. *Core*: Successfully extract a set of features and use them to train and detect emotions from speech with a reasonable accuracy.
2. *Core*: High run-time performance.
3. *Extension*: Successfully apply the emotion detector to provide feedback about public speaking skills.

As will be shown in this chapter, the detection results outperform a recent PhD thesis (Table 4.1) and a number of papers [6, 21, 28]. Moreover, unlike the previous research which runs as a batch job, the detection can be run in real-time. The extension, which presents a novel application of the emotion detector for speech quality assessment, provides useful feedback by detecting the best applicable quality classes.

Type of data	Accuracy (%)
70–30 training/testing split	86 (79)
Training data	99 (81)

Table 4.1: Emotion detection accuracies in percentages. The results in the PhD thesis [27] are shown in parentheses. A random choice would result in 11% accuracy.

4.1.2 Real-time Performance

One requirement for the system was to run in real-time, in contrast to previous work which could only be run as a batch job [27]. This section presents evidence that this goal is achieved.

The average latency in milliseconds of the classification stage is shown in Figure 4.1. It is measured as the time between the detection of the end of a segment and the presentation of the result. As shown in the figure, the system classifies normal sentences (1–15 s) in 0.046–0.110 s, which is a barely noticeable delay for users.

The increase of latency with longer sentences is caused by the feature extractors that need to process more frames with longer audio segments. However, since speakers need to pause to breath, and thus limit the segment length, this increase in latency did not cause problems in practice.

To avoid interfering with concurrently running processes, the resource usage is capped during development. The resulting average resource usage of the training and classification stages is shown in Table 4.2. The very low memory consumption and sub-1/10 CPU usage allows the trainer and classifier to be run on multi-user systems.

Overall, the performance met the real-time user response requirement set in the project plan. The low latency can be attributed to the use of fast feature ex-

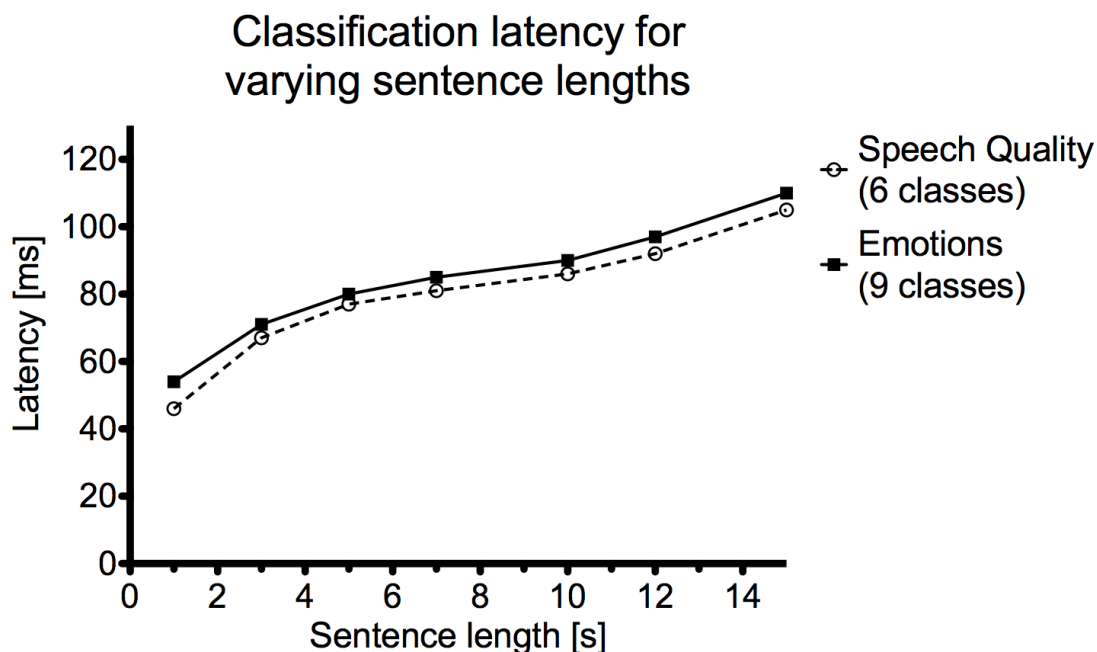


Figure 4.1: Average live classification latency in milliseconds on a 2.66 GHz MacBook Pro with 4 GB RAM.

Type of data	CPU (%)	Memory (%)
Training	11.5	0.1
Classification	10.0	0.9

Table 4.2: Average resource usage on a 2.66 GHz MacBook Pro with 4 GB RAM.

traction and machine learning libraries, and choice of efficient data structures for combining the pairwise SVMs. Moreover, the feature selection stage reduced the feature set size by over 100x on average, significantly lowering the complexity of the optimisation and classification problem for the SVMs.

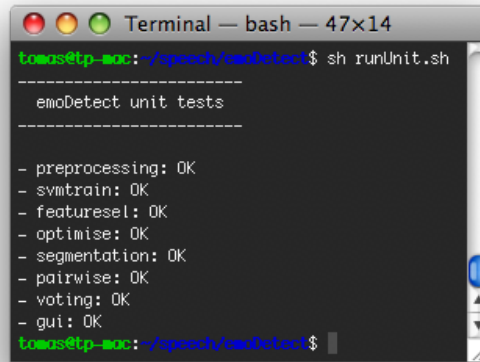
4.1.3 Functionality and testing

Another important area of evaluation is the features implemented and their testing.

All the essential features that the project was set to achieve, namely a fully working emotion detector and optimisation algorithms, are successfully implemented. The extension for speech quality analysis is also achieved. This reflects adequate

planning and preparation before starting the implementation.

As the system consists of a number of linearly connected components, each of which affects the results of the next one, it is crucial to perform thorough and systematic testing of the components. Throughout the development, a number of unit tests were written to test both the individual and combined functionality of the components. A simple unit testing framework was written (Appendix A.3, Figure 4.2) to run the tests. Each test consisted of a set of predefined inputs and outputs and logic for checking that the output equals the predefined output.

A screenshot of a terminal window titled "Terminal — bash — 47x14". The prompt is "tomas@tp-soc: ~/speech/emoDetect\$". The user has entered "sh runUnit.sh". The output shows "emoDetect unit tests" followed by a list of tests, all marked as "OK": preprocessing, svmtrain, featuresel, optimise, segmentation, pairwise, voting, and gui. The prompt returns to "tomas@tp-soc: ~/speech/emoDetect\$".

```
Terminal — bash — 47x14
tomas@tp-soc: ~/speech/emoDetect$ sh runUnit.sh
-----
emoDetect unit tests
-----
- preprocessing: OK
- svmtrain: OK
- featuresel: OK
- optimise: OK
- segmentation: OK
- pairwise: OK
- voting: OK
- gui: OK
tomas@tp-soc: ~/speech/emoDetect$
```

Figure 4.2: Screenshot showing the unit test framework results.

4.2 Classification results

This section evaluates the project in terms of how well speech is classified. A wide range of experiments were conducted, with the five most interesting ones being reported in the subsections below. The purpose of the experiments is twofold. First, it is to investigate the accuracies obtained and how they compare to previous work. Second, it is also to illustrate the wide range of algorithms implemented and how they improved the detection rates.

4.2.1 Experiment 1: Comparison to previous work

This experiment aims to compare the accuracy of the classifier against that presented by Sobol Shikler’s PhD thesis [27] and a number of papers [6, 21, 28]. The

approaches towards classification are similar to Sobol Shikler, but this project runs in real-time and uses a larger full feature set F .

For evaluation, the commonly used 10-fold cross-validation method is employed. The training set is divided into 10 equal-sized subsets. Each subset is sequentially tested using a classifier trained on the remaining 9 subsets. Every subset is classified once, so the cross-validation accuracy is the percentage of data which are correctly classified.

Table 4.3 shows the cross-validation accuracies of the $\binom{9}{2} = 36$ pairwise machines for the 9 categories selected in Section 2.2.4. Each entry in the table represents the 10-fold cross-validation accuracy for one of the SVMs used to differentiate between two classes.

All accuracies are greater than the values obtained in previous research. The results are constantly above 80%, in contrast to the lower bound 60% obtained previously. Overall, the machines achieved remarkably high accuracies in pairwise decisions.

Compared to Dellaert [6], Petrushin [21] and Steidl [28], who achieved accuracies 60–65% (4 classes), 70% (5 classes) and 60% (4 classes) respectively, the results shown in Tables 4.6 and 4.7 (72% and 86% using maximum probability and thresholding for fusion respectively) outperform the earlier papers. This is even though the number of candidate classes for classification in this study is much higher (9 as opposed to 4 or 5). Whereas with 9 classes a baseline random classifier would give $\frac{100}{9} = 11\%$ accuracy on average, with 4 or 5 classes the baseline would be 25% or 20%. The higher the number of candidate classes, the more difficult the classification task becomes.

4.2.2 Experiment 2: Machine learning algorithms

This experiment aims to compare the accuracy of the two machine learning algorithms that were found to achieve highest accuracies. As outlined in Section 2.2.3, a wide range of different classifiers were tried, with SVMs and decision tree-based C4.5 performing best.

C4.5 constructs a decision tree from a set of data by dividing up the data according to the information gain $I(X; Y)$ defined in Section 3.1.3. It recursively splits the tree by the attribute with the highest $I(X; Y)$ in the training, yielding a decision tree that can be reused for classification.

	absorbed	excited	interested	joyful	opposed	stressed	sure	thinking	unsure
absorbed		93 (81)	87 (82)	96 (82)	96 (78)	89 (87)	85 (84)	82 (73)	84 (64)
excited			90 (71)	84 (60)	81 (71)	80 (61)	94 (83)	90 (72)	87 (75)
interested				92 (77)	92 (75)	91 (66)	90 (78)	90 (84)	85 (72)
joyful					86 (71)	85 (61)	99 (83)	95 (72)	92 (75)
opposed						93 (84)	91 (72)	94 (81)	92 (79)
stressed							86 (84)	88 (75)	86 (78)
sure								94 (75)	88 (78)
thinking									90 (89)
unsure									

Table 4.3: Ten-fold cross-validation percentages for the pairwise machines. For comparison, results of previous research [27] are shown in parentheses. The values that improve upon the results of previous research are shown in bold. The average accuracy is 89%, compared to 76% in previous research.

Unlike in previous research [27], the experiment found that SVMs could provide significantly higher performance than from C4.5 for every pairwise machine. This is achieved by using grid search to optimise the SVM parameters. The results are illustrated in Table 4.4, where the SVM results are compared to the results with C4.5. As a consequence, only SVMs were used for implementation.

Another factor that affected the results is feature selection. The square-bracketed values in Table 4.4 show the number of features in each SVM. These are only a small fraction of the original 6669 feature combinations extracted by openSMILE. When training on all 6669 features, accuracies above 60% were rarely obtained.

4.2.3 Experiment 3: Grid search for SVM parameter optimisation

This experiment aims to compare the accuracy obtained with and without using a grid search algorithm (Algorithm 1) for optimising SVM parameters. In previous studies optimisation of the machine learning algorithm has not received as much attention. This study employs a method based on maximising cross-validation accuracy to obtain a considerable improvement in detection accuracy.

As described in Section 3.1.5, the greedy grid search algorithm chooses optimal (C, γ) parameters for each pairwise SVM. It first does a rough search over the pairwise values and then recursively narrows down the search space by searching around the values that produced the highest cross-validation accuracy. This turned out to be a major contributor to the high accuracy of the SVM approach. Previous work has ignored this subtle but clearly important classifier optimisation.

The effect for using grid search with the three pairwise fusion mechanisms is shown in Table 4.5. A significant improvement, between 10% and 38%, is observed. This is as high an improvement as that gained from choosing SVM over C4.5 in Section 4.2.2. As the optimisation maximises the cross-validation accuracy instead of the training data classification accuracy, the optimisation did not result in overfitting of the model.

4.2.4 Experiment 4: Pairwise fusion methods

This experiment demonstrates the results of using the three different algorithms described in Section 3.2.4 for fusing the pairwise SVMs. The confusion matrices

	absorbed	excited	interested	joyful	opposed	stressed	sure	thinking	unsure
absorbed		93 (79) [53]	87 (74) [55]	96 (80) [79]	96 (85) [46]	89 (80) [58]	85 (74) [39]	82 (76) [52]	84 (79) [36]
excited			90 (72) [44]	84 (70) [38]	81 (77) [48]	80 (69) [31]	94 (70) [47]	90 (78) [97]	87 (69) [57]
interested				92 (81) [65]	92 (72) [54]	91 (68) [46]	90 (77) [68]	90 (79) [85]	85 (69) [34]
joyful					86 (77) [53]	85 (70) [54]	99 (75) [75]	95 (84) [103]	92 (73) [62]
opposed						93 (75) [43]	91 (73) [28]	94 (75) [86]	92 (76) [28]
stressed							86 (68) [61]	88 (74) [100]	86 (68) [34]
sure								94 (80) [78]	88 (70) [54]
thinking									90 (74) [72]
unsure									

Table 4.4: Ten-fold cross-validation percentages using grid-searched SVMs and C4.5. The number without bracketing is the result using SVMs. The results using C4.5 are shown in parentheses. The number of features used are in square brackets.

Type of data	Thresholding	Max probability	Max wins
Training data, grid search	99 (81)	86	88
Training data, no grid search	86 (81)	55	50
70–30 training/testing split, grid search	86 (79)	72	70 (70)
70–30 training/testing split, no grid search	76 (79)	47	48 (70)

Table 4.5: Accuracies in percentages for the three different methods to determine a winner, with and without grid search. The results by previous work [27] are shown in parentheses.

for the final emotion detection results for each fusion technique are obtained and shown below.

The confusion matrices for the probability, thresholding and majority voting methods are shown in Table 4.6 (Figure 4.3), Table 4.7 (Figure 4.4) and Table 4.8 (Figure 4.5) respectively. The shaded values show the percentage of correct classifications for each class. The unshaded values show the percentage of classifications in which the ground truth class (column) is mistaken for the inference class (row). For each fusion method, a random choice would result in an overall 11% accuracy. All three fusion methods present average accuracies that are either equal to or higher than achieved previously [27], as is summarised in Table 4.1.

Notably, the average accuracy of the maximum probability fusion technique is higher than that achieved by majority voting (72% vs 70%). However, for some classes the majority voting accuracy is higher (e.g. *stressed* and *interested*). Thus, in future work a slightly higher accuracy could be achieved by combining these methods, using the one giving best results per individual class.

By inspection of the confusion matrices, some classes were clearly better detected than others. The classes *opposed* and *sure* presents the lowest values using any method. This is reflected by the lower number of training data samples (38 and 53 samples, compared to the average of 61) resulting from the categorisation choice to allow comparison to previous research [27]. Similarly, the class with most samples (*joyful*, 94 samples) is most frequently mistaken to be the correct class. In future work classes with equal number of training corpus samples could be used, albeit it will not allow for direct comparison to previous work.

As expected, the thresholding fusion method yields highest detection accuracies

Inferred (down) Actual (right)	absorbed	excited	interested	joyful	opposed	stressed	sure	thinking	unsure
absorbed	74	0	2	0	0	1	2	1	1
excited	0	75	2	6	0	2	6	0	1
interested	4	0	69	0	0	2	2	3	1
joyful	4	10	6	79	16	11	4	3	4
opposed	0	2	0	2	62	1	2	0	0
stressed	4	8	6	3	8	67	9	1	8
sure	0	0	2	2	5	2	63	0	0
thinking	7	0	8	3	0	4	11	86	17
unsure	7	4	6	4	8	8	2	6	68

Table 4.6: Confusion matrix for emotion detection using maximum probability for pairwise fusion. The column headings show the ground truth and the rows show inferences. Average accuracy is 72%.

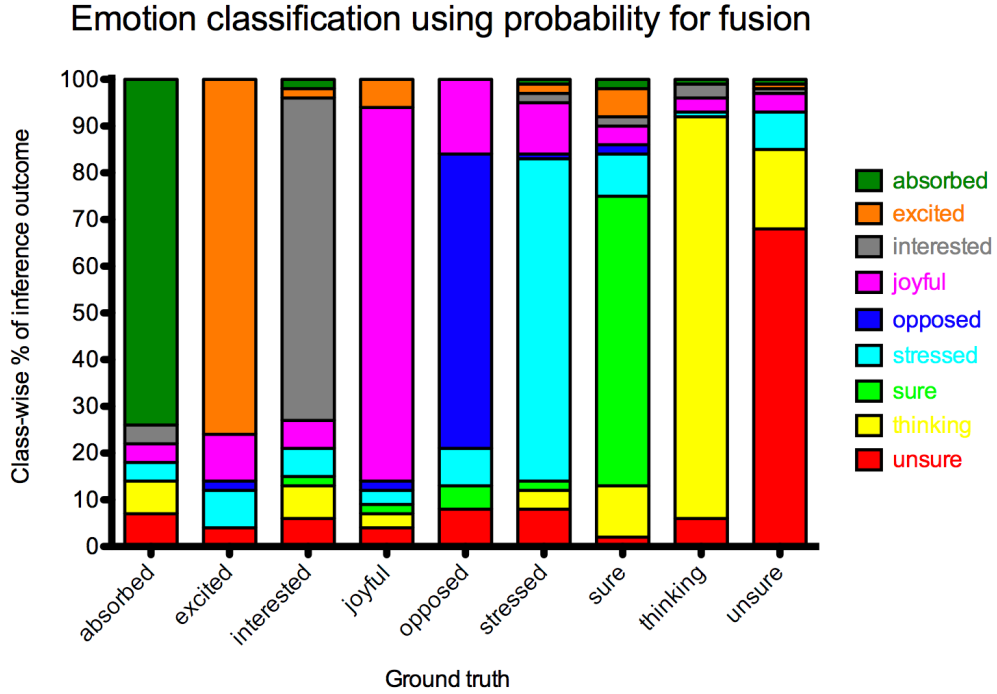


Figure 4.3: Confusion graph for emotion detection using maximum probability for pairwise fusion.

Inferred (down) Actual (right)	absorbed	excited	interested	joyful	opposed	stressed	sure	thinking	unsure
absorbed	93	4	15	0	0	4	12	23	24
excited	15	85	10	29	27	46	24	6	14
interested	22	2	83	14	3	10	11	17	14
joyful	15	35	21	91	41	39	22	23	22
opposed	0	14	6	22	73	11	17	7	8
stressed	15	60	31	56	51	92	31	24	29
sure	11	19	6	4	16	9	74	11	9
thinking	48	15	42	19	24	19	28	93	56
unsure	48	8	52	24	22	31	26	56	91

Table 4.7: Confusion matrix for emotion detection using thresholding for pairwise fusion. The column headings show the ground truth and the rows show inferences. Average accuracy is 86%.

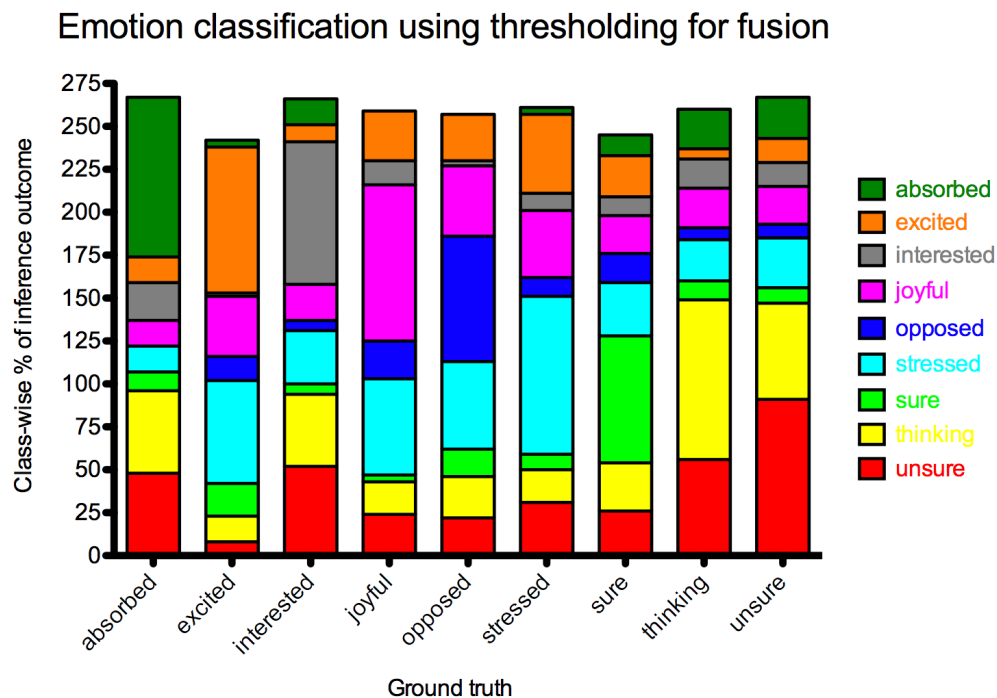


Figure 4.4: Confusion graph for emotion detection using thresholding for pairwise fusion.

Inferred (down) Actual (right)	absorbed	excited	interested	joyful	opposed	stressed	sure	thinking	unsure
absorbed	74	0	0	0	0	1	2	2	3
excited	0	65	3	6	3	2	6	0	1
interested	4	0	73	2	0	2	4	3	0
joyful	4	13	4	76	16	10	4	1	5
opposed	0	2	0	5	59	1	2	1	3
stressed	4	10	4	3	8	71	7	3	8
sure	0	4	2	2	5	2	63	0	0
thinking	7	2	8	3	1	4	10	83	16
unsure	7	4	6	3	8	7	2	7	64

Table 4.8: Confusion matrix for emotion detection using majority voting for pairwise fusion. The column headings show the ground truth and the rows show inferences. Average accuracy is 70%.

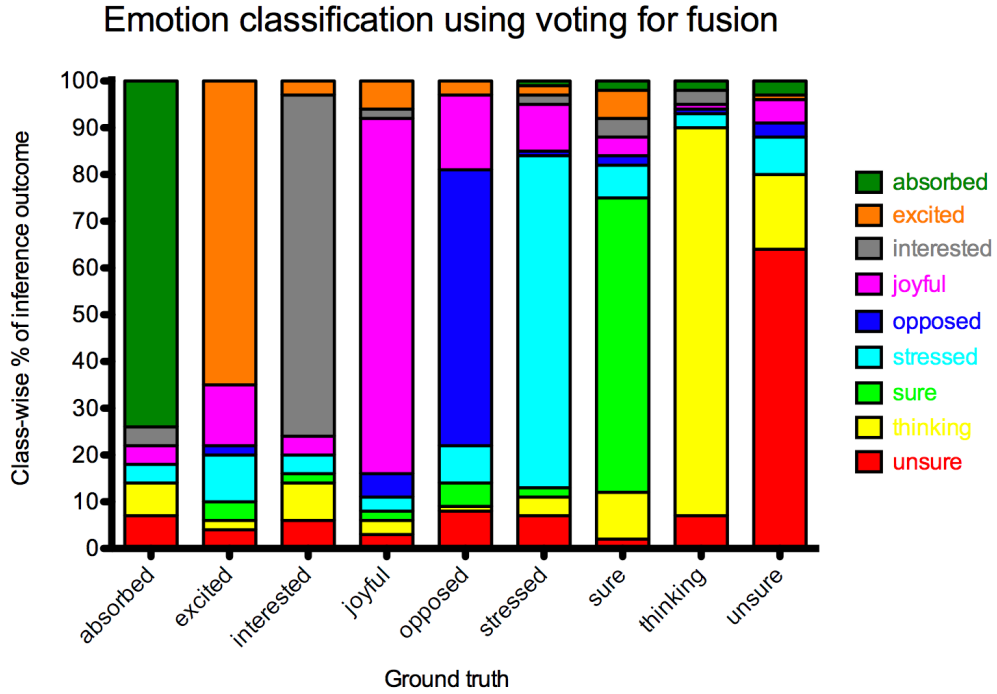


Figure 4.5: Confusion graph for emotion detection using majority voting for pairwise fusion.

since several classes can be selected at a time. This, however, also leads to much higher confusion values which in some cases can be very high. For example, an *excited* voice causes *joyful* to be detected in 35% of cases, compared to detection rate 85% for *excited*. It could be argued that this is because these emotions are similar, and perhaps the samples that are analysed contain both emotions. However, confusions such as 60% detection of *stressed* for samples that are labelled as *excited* may be more difficult to explain using this hypothesis. It is likely that some of the high confusion rates are caused by the uneven distribution of samples between classes.

4.2.5 Experiment 5: Speech quality assessment

This section evaluates the performance of the extension where the training and classification systems described in Sections 3.1 and 3.2 are used to assess the speech quality. The emotion classifier is retrained using six labels describing speech quality. This is a novel application of the classifier that has not been attempted before.

The 124 samples are each labelled with scores for all 6 classes. The samples for each class are halved into the higher and lower parts according to the score. One binary SVM per class is used to derive a probability. This allows the user to try to maximise all class probabilities, which is a useful goal for speech coaching.

The results of speech quality assessment using the modified emotion detector described in Section 3.2.5 are shown in Table 4.9. Again, a significant improvement in classification accuracy is observed after optimising the SVM parameters using grid search, with an average 13% true-positive improvement in accuracy for the 70–30% training/testing split.

All classes can be accurately detected, with a high proportion of true positives and true negatives. The classes *competent* and *dynamic* present slightly lower detection accuracies, perhaps due to the small variation in scores resulting from a small corpus size. Overall, however, the speech quality assessment are high enough (average 79% for a 70–30% split) to provide useful feedback to speakers for all classes.

Class	Training data (no grid)	70–30% split (no grid)	Training data (grid)	70–30% split (grid)	10-fold cross- validation (grid)
clear	79 [83]	80 [70]	95 [98]	90 [76]	80
competent	61 [87]	37 [87]	78 [97]	67 [95]	74
credible	74 [87]	55 [79]	98 [98]	69 [96]	80
dynamic	60 [91]	51 [90]	91 [100]	62 [89]	77
persuasive	87 [71]	85 [58]	94 [62]	87 [60]	82
pleasant	86 [76]	86 [75]	100 [98]	97 [76]	93
Mean	75 [82]	66 [76]	93 [92]	79 [82]	81

Table 4.9: Detection accuracies in percentages for speech quality assessment using the Speech Tutor corpus. Accuracies are true positive rates, with true negatives in square brackets. A 70–30% training/testing split is used to allow direct comparison to emotion detection accuracies. Results are shown both with and without grid search optimisations for choosing SVM parameters.

Chapter 5

Conclusion

This study presents a new framework for emotion detection whose accuracy outperforms a recent PhD thesis [27] and a number of papers [6, 21, 28]. Moreover, it achieves this in real-time, as opposed to previous work which was run as a batch job. The novel application of the system for speech quality assessment also achieves high detection accuracies.

A strong theoretical understanding of machine learning algorithms was the key to success. Small, compact pairwise classifiers that were highly optimised using a grid search algorithm were successfully implemented. Experiment 1 shows that the framework provides high classification accuracies which outperform previous research. Experiment 2 investigates different machine learning algorithms. Classification results show that the highly optimised SVMs produce higher cross-validation accuracies than other algorithms.

The project illustrates a method to optimise the misclassification and exponentiation coefficients (C, γ) in Equations 2.2 and 2.3. Experiment 3 demonstrates the considerable improvement resulting from this optimisation. The fusion of pairwise classifiers is described in Experiment 4. Results show that using maximum probability for fusion achieves highest accuracies for single-label outputs. To achieve real-time performance, it was also essential to choose suitable data structures and algorithms. Section 4.1.2 shows that millisecond-level latency was achieved running on a standard laptop.

This core of the project was completed on schedule as set out in the project proposal.

The extension was also successfully completed on schedule. In particular, the

emotion detector is extended to support automatic speech quality assessment. Experiment 5 shows that a high class detection accuracy is achieved using the Speech Tutor corpus labelled for this project.

In conclusion, this project shows that building a fast and efficient speech emotion detector is a challenging but achievable goal. By combining a thorough theoretical understanding of machine learning beyond the Computer Science Tripos with diligent classifier optimisation, a system that outperforms recent research is attained. The key design principles behind the successful implementation of a large real-time system included choosing efficient data structures and algorithms, and employing suitable software engineering tools. In sum, the project employs an understanding of a wide area of Computer Science to demonstrate that highly accurate speech emotion detection is possible, and that it can be done in real-time.

Bibliography

- [1] H. Altun and G. Polat. *New Frameworks to Boost Feature Selection Algorithms in Emotion Detection for Improved Human-Computer Interaction*. Lecture Notes in Computer Science, 2007.
- [2] S. Baron-Cohen, O. Golan, S. Wheelwright, and J. J. Hill. *Mind Reading: The Interactive Guide to Emotions*. Jessica Kingsley Publishers, University of Cambridge, 2004. ISBN 1 84310 214 5.
- [3] A. Batliner, K. Fisher, R. Huber, J. Spilker, and E. Noth. Desperately seeking emotions or: Actors, wizards, and human beings. In *Proc. of the International Speech Communication Association Workshop on Speech and Emotion*, pages 195–200, 2000.
- [4] M. Berouti, R. Schwartz, and J. Makhoul. Enhancement of speech corrupted by acoustic noise. *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pages 208–211, 1979.
- [5] M. Brookes. VOICEBOX, February 2010. <http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html>.
- [6] F. Dellaert, T. Polzin, and A. Waibel. Recognizing emotion in speech. In *Proceedings of fourth international conference on spoken language processing*, volume 3, pages 1970–1973, 1996.
- [7] F. Eyben, M. Wöllmer, and B. Schuller. openEAR – Introducing the Munich open-source emotion and affect recognition toolkit. In *Proc. 4th International HUMAINE Association Conference on Affective Computing and Intelligent Interaction 2009 (ACII 2009)*, IEEE, Amsterdam, The Netherlands, September 2009.
- [8] R.E. Fan, P.H. Chen, and C.J. Lin. Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

- [9] K. Forbes-Riley and D. Litman. Predicting emotion from spoken dialogue from multiple knowledge sources. In *Proc. of human language technology conference of the north american chapter of the association for computational linguistics*, 2004.
- [10] J. Friedman. Another approach to polychotomous classication. Technical report, Stanford University, Department of Statistics, 1996.
- [11] O. Golan, S. Baron-Cohen, S. Wheelwright, and J. J. Hill. Systemizing empathy: Teaching adults with asperger syndrome and high functioning autism to recognize complex emotions using interactive multimedia. *Development and Psychopathology*, 18:589–615, 2006.
- [12] M. A. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11, 2009.
- [13] M. A. Hall and L. A. Smith. Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. *Florida Artificial Intelligence Symposium*, pages 235–239, 1999.
- [14] J. D. Haynes and G. Rees. Decoding mental states from brain activity in humans. *Nature Reviews Neuroscience*, 7:523–534, 2006.
- [15] S. King. Edinburgh speech tools, February 2010. http://www.cstr.ed.ac.uk/projects/speech_tools/.
- [16] H. T. Lin, C. J. Lin, and R. C. Weng. A note on Platt’s probabilistic outputs for support vector machines. *Machine Learning*, 68:267–276, 2007.
- [17] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. In *Trans. on Knowledge and Data Engineering*, volume 17, pages 491–502, 2005.
- [18] Mathworks. Matlab, February 2010. <http://www.mathworks.com>.
- [19] P. J. Moreno and R. Rifkin. Using the fisher kernel method for web audio classification. In *Proc. of IEEE International Conference on Acoustics Speech and Signal Processing*, volume 4, pages 2417–2420, 2000.
- [20] Nokia. Qt documentation, February 2010. <http://qt.nokia.com/doc/>.
- [21] V. Petrushin. Emotion in speech: Recognition and application to call centers. *Artificial Neural Network Intelligence Engineering*, pages 7–10, 1999.

- [22] S. Pöyhönen, A. Arkkio, P. Jover, and H. Hyötyniemi. Coupling pairwise support vector machines for fault classification. *Control Engineering Practice*, 13(6):759–769, 2005.
- [23] L. Qing-kun and Q. Pei-wen. Model selection for SVM using mutative scale chaos optimization algorithm. *Journal of Shanghai University*, 10:531–534, 2006.
- [24] U. Rathmann. Qt widgets, February 2010. <http://qwt.sourceforge.net/>.
- [25] K. R. Scherer. Vocal affect expression: A review and a model for future research. *Psychological bulletin*, 99:143–165, 1986.
- [26] B. Schuller, S. Steidl, and A. Batliner. The interspeech 2009 emotion challenge. In *Interspeech*, Brighton, UK, 2009.
- [27] T. Sobol Shikler. *Analysis of affective expression in speech*. PhD thesis, Cambridge University, 2007.
- [28] S. Steidl, M. Levit, A. Batliner, E. Noeth, and E. Niemann. Of all things the measure is man – Automatic classification of emotions and interlabeler consistency. In *Proceeding of the IEEE international conference on acoustics, speech, and signal processing*, 2005.
- [29] V. N. Vapnik. The nature of statistical learning theory. *Springer*, 1998.
- [30] G. Yu, S. Mallat, and E. Bacry. Audio denoising by time-frequency block thresholding. *IEEE Transactions on signal processing*, 56:1830–1839, 2008.

Appendix A

Sample source code

A.1 C++ code

The listing below shows some sample code for the PAIRWISE-COMBINATOR and PAIRWISE-VOTING algorithms.

```
int cPairwiseCombinator::processComponentMessage( cComponentMessage *_msg )
{
    if (isMessageType(_msg,"classificationResult")) {
        //get data from message
        double *probEstim = (double*)_msg->custData;
        int nClasses = _msg->intData[0];
        char **classNames = (char**)_msg->custData2;
        userTime1 = (double)_msg->userTime1;
        userTime2 = (double)_msg->userTime2;

        //check if final (numClasses=8 by default)
        bool final = false;
        if(combinedProbEstim.size() > 0) final=true;
        map<string, multiset<double> >::iterator it1; //iterator for
        finality check
        for (it1=combinedProbEstim.begin(); it1!=combinedProbEstim.end()
            && final; it1++) {
            if((*it1).second.size() != (numClasses-1) && (*it1).
                second.size() != numClasses) final = false;
        }

        //create new multiset for probabilities
        multiset<double> multi;

        //create new entries for classes in maps if they don't exist
        for(int i=0; i<nClasses; i++) {
            if(combinedProbEstim.count(classNames[i]) == 0)
                combinedProbEstim.insert( pair<string, multiset<
                    double> >(classNames[i], multi ) );
            if(results.count(classNames[i]) == 0)
```

```

        results.insert( pair<string, pair<int, double> >(
            classNames[i], make_pair(0,0) ) );
    }

    //increment win count for winner
    (results[_msg->msgtext].first)++;

    //iterate over nonzero probability estimates
    if (probEstim != NULL) {
        for (int i=0; i<nClasses; i++) {
            //insert probability estimate into map
            combinedProbEstim[classNames[i]].insert(
                probEstim[i]);
            //calculate total probability
            results[classNames[i]].second += probEstim[i];
        }
    }

    //print results & findmax if this is the final calculation
    if(final) {
        printResults();
        findMax();

        //clear maps after final calculation
        combinedProbEstim.clear();
        results.clear();
    }

    return 1; // message was processed
}
return 0; // if message was not processed
}

void cPairwiseCombinator::printResults()
{
    map<string, multiset<double> >::iterator it1;
    multiset<double>::iterator it2;
    int i=0; //count for class ID
    for (it1=combinedProbEstim.begin(); it1!=combinedProbEstim.end(); it1++)
    {
        string name = (*it1).first;
        char *buf = (char*)name.c_str();
        fprintf(stderr, "%s:", buf);

        for (it2=(*it1).second.begin(); it2!=(*it1).second.end(); it2++)
        {
            fprintf(stderr, " %f", *it2);
        }
        fprintf(stderr, " \n(div %d) = %f \n wins: %d", numClasses, (
            results[name].second)/numClasses, results[name].first);
        fprintf(stderr, "\n");

        i++;
    }
}

void cPairwiseCombinator::findMax()
{
    int maxWins = 0;

```



```

string maxWinsName;
double maxProbability = 0.0;
string maxProbabilityName;

map<string, pair<int, double> >::iterator it3;

//iterate over results map
for (it3=results.begin(); it3!=results.end(); it3++) {
    string name = (*it3).first;

    //find maximum number of wins
    if ((*it3).second.first > maxWins) {
        maxWinsName = name;
        maxWins = (*it3).second.first;
    }
    //find maximum probability
    if ((*it3).second.second > maxProbability) {
        maxProbabilityName = name;
        maxProbability = (*it3).second.second;
    }
}
if(results.size() > 0) {
    const char *maxWinsNameBuf = maxWinsName.c_str();
    const char *maxProbabilityNameBuf = maxProbabilityName.c_str();
    fprintf(stderr, "maxWins: %s - %d - maxProbability: %s - %f",
        maxWinsNameBuf, maxWins, maxProbabilityNameBuf,
        maxProbability/numClasses);
    fprintf(stderr, "\n\n");
}
}

```

A.2 Bash script

The listing below shows the Bash script for constructing pairwise corpora.

```

#!/bin/sh
#
# Script for converting a corpus into a set
# of separate pairwise corpora
#

#source dir
sdir="../audio"

#temporaries
let c=0
b=false

for i in `ls $sdir`; do
    for j in `ls $sdir|awk "!\$1!=\$i\""; do

        #loop through array, see if j is in it
        for e in "${arr[@]"; do
            if [ "$e" = "$j" ]; then

```

```

        b=true
    fi
done

#if j isn't in array (i.e. hasn't been selected before)
#create dir, copy files from srcdir
if [ "$b" = "false" ]; then
    dir=$i-$j;
    mkdir -p $dir
    echo $dir
    cp -r $sdir/$i $sdir/$j $dir
fi

#set boolean back to false
b=false
done

#increment count
((c++))
#store in array
arr[c]=$i

done

```

A.3 Unit tests

The unit testing script written in Bash is shown below.

```

#!/bin/bash
#
# Unit testing script for emoDetect

run() {
    sh unittests/$1.sh
    res=$?
    if [ $res -eq 0 ]; then
        echo - $1: FAIL
    else
        echo - $1: OK
    fi
}

echo "_____"
echo "emoDetect_unit_tests_"
echo "_____"
echo

run preprocessing
run svmtrain
run optimise
run segmentation
run pairwise
run voting
run gui

```

Appendix B

Extracted features

The low-level features extracted by openSMILE [7] are shown in Table B.1. The functionals computed using the low-level features are shown in Table B.2.

Feature Group	Features	#
Signal energy	Root mean square; Logarithmic	2
Fundamental frequency (F0) based on autocorrelation (ACF)	F0 frequency; F0 strength (peak of ACF); F0 quality (zero-crossing rate of ACF w.r.t. F0)	3
Mel-Frequency Cepstral Coefficients	Coefficients 0–15	16
Spectral	Centroid; Roll-off (10%, 25%, 50%, 75%, 90%); Flux; Frequency band energy (0–250Hz, 0–650Hz, 250–650Hz, 1000–4000Hz); Position of maximum; Position of minimum	13
Time signal	Zero-crossing rate; Maximum value; Minimum value; Mean (DC)	4
Linear Predictive Coding	Coefficients 0–11	12
Voice quality	Harmonics to noise ratio; Jitter; Shimmer	3
Pitch by harmonic product spectrum	F0 frequency; Probability of voicing (pitch strength)	2

Table B.1: OpenSMILE’s [7] 55 low-level audio features.

Functionals Group	Functionals	#
Min/max	Max/min value; Relative position of max/min value; Range	5
Mean	Arithmetic mean; Arithmetic mean with absolute values; Quadratic; Maximum value arithmetic mean	4
Non-zero	Arithmetic mean of non-zero values; Percentage of non-zero values	2
Quartiles	20%, 50%, 75% quartiles; inter-quartile range (2-1, 3-2, 3-1)	6
Percentiles	95%, 98% percentiles	
Moments	Variance; Skewness; Kurtosis; Standard deviation	4
Centroid	Centroid (centre of gravity)	1
Threshold crossing rate	Zero-crossing rate; Mean-crossing rate	2
Linear regression	2 coefficients; Linear & quadratic regression error	4
Quadratic regression	3 coefficients; Linear & quadratic regression error	5
Peaks	Number of maxima; Mean distance between peaks; Mean value of peaks; Arithmetic mean of peaks	4
Segments	Number of segments based on delta thresholding	
Time	Length of time that values are above 75% of the total range; Length of time that values are below 25% of the total range; Rise/fall time	5
Discrete Cosine Transform	Coefficients 0-5	6

Table B.2: OpenSMILE's [7] 50 functionals.

Tomas Pfister
Gonville & Caius College
tjp35

Computer Science Tripos Part II Project Proposal

Emotion profiling of speech

19 October 2009

Project Originator: Prof P. Robinson

Resources Required: See attached Project Resource Form

Project Supervisor: Prof P. Robinson

Signature:

Director of Studies: Dr G. Titmus and Prof P. Robinson

Signature:

Overseers: Dr N. Dodgson and Prof R. Anderson

Signatures:

Background and Introduction

In persuasive communication, special attention is required to what non-verbal clues one conveys. Untrained speakers often come across as bland, lifeless and colourless. Precisely measuring and analysing the voice is a difficult task and has in the past been entirely subjective. By incorporating the recent developments in computer science with inspiration from psychology and acoustics, it should be possible to make the analysis more objective.

In previous work on emotion detection from speech, Tal Sobol-Shikler [3], under supervision of Prof P. Robinson, presented a system for batch processing speech samples using pair-wise decision machines. This project will combine some of the ideas in her work with more recent research to build a real-time speech emotion profiling system.

Project Proposal

The purpose of this project is to build a tool that will help in understanding what vocal information is sent. The aim is to show the speaker in real-time what emotions and non-verbal signals are projected to allow them to be adjusted. The core part of the project proposal is to develop methods for the emotional profiling of speech. The extension will be to use a different database to retrain and revalidate the profiler for public speaking tutoring.

The core profiler will be trained and validated using the MindReading emotions database, collected by Professor Simon Baron-Cohen at the Autism Research Centre at the University of Cambridge. It consists of 4411 acted sentences, covering 756 concepts of emotions within 24 emotion groups.

Once a set of vocal features and extraction algorithms have been defined, the profiler will use machine learning techniques to train the system based on the vocal features present in the speech samples of the MindReading database. After the training phase, the profiler segments the speech, extracts the features, classifies the speaker's emotions and presents a summary of the most prevalent emotions in the speech so far. A similar approach will be used for training a virtual public speaking tutor in the extension.

Special resources

Access to the Rainbow Lab for system testing, to the MindReading database of affective speech for training data, and to an additional database for the extension.

Substance and Structure of the Project

The key challenges to be addressed:

1. Theoretical understanding of the background material. The project will involve learning and understanding a considerable amount of mathematics of machine learning not covered in the Computer Science Tripos. This will include investigating applying pair-wise decision machines and voting mechanisms to training the mapping between vocal features and emotional states. It is expected that the learning process will take a significant amount of time.
2. Implementation of a real-time speech emotion detector using the C++ object-oriented language. This will involve designing the system architecture, implementing feature extraction algorithms, and possibly improving the machine learning libraries. The aim is to reuse some of the design ideas by Sobol-Shikler [3] and apply them to a real-time system. Existing implementations will be reused whenever appropriate.
3. Choice of suitable data structures and algorithms. This is essential for achieving real-time performance.

Starting Point

I have done some background reading during the summer and have a basic knowledge of C++. It is expected that I will be able to use some of the open-source code in LibSVM [2] and openEAR [1] projects. Lecture courses that may be useful for the project include Mathematical Methods for CS, Natural Language Processing, Programming in C++, Software Design and Software Engineering.

Evaluation Criteria

The success of the core project will be evaluated based on the performance and features of the emotion profiler. It is expected to successfully extract a set of features and use them to train and detect emotions from speech with a reasonable accuracy. The run-time performance will also be evaluated to investigate whether suitable data structures and algorithms were selected.

In the optional extension, the results will be evaluated by considering whether the tutor successfully applies the profiler from the core part of the project to provide feedback about public speaking skills.

Timetable and Milestones

In the planned timetable the project has been segmented into 10 stages, each lasting about 2 weeks. Notes will be taken in a logbook during all stages to assist with the final write-up.

Stage	Date	Tasks and milestones
1	Oct 19 – Oct 30	Background study. Reading relevant papers.
2	Nov 2 – Nov 13	Planning and preliminary familiarisation. Choice of tools. Writing some toy programs to investigate the chosen tools. Milestone: Thorough understanding of the preparation chapter's material.
3	Nov 16 – Nov 27	Implementation and debugging of the core part of emotion detector.
4	Nov 30 – Dec 11	Finishing off implementation of core. Start working on extension. Milestone: Most of the work in the implementation chapter completed.
5	Jan 4 – Jan 15	Writing up progress report for presentation. Evaluation of emotion detector according to the success criteria. Milestone: Progress report written; first draft of evaluation chapter finished.
6	Jan 18 – Jan 29	Further work on extension. Address any issues arisen from the progress report.
7	Feb 1 – Feb 12	Finish off extension and any uncompleted features. Milestone: All implementation work done.
8	Feb 15 – Feb 26	Writing up.
9	Mar 1 – Mar 12	Writing up. Milestone: First draft sent to supervisor before end of Lent term.
10	Mar 22 – Apr 2	Writing up second draft to supervisor. Milestone: Submission of second draft by Friday 2 April. Final submission by Friday 23 April.

References

- [1] Björn Schuller Florian Eyben, Martin Wöllmer. openEAR – Introducing the Munich open-source emotion and affect recognition toolkit. In *Proc. 4th International HU-MAINE Association Conference on Affective Computing and Intelligent Interaction 2009 (ACII 2009)*, *IEEE*, Amsterdam, The Netherlands, September 2009.
- [2] C.J. Lin R.E. Fan, P.H. Chen. Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, 6:1889–1918, 2005.
- [3] Tal Sobol-Shikler. *Analysis of affective expression in speech*. PhD thesis, University of Cambridge, Cambridge, UK, 2007.