Optimization Techniques for Speech Emotion Recognition

por Julia (Yulia) Sidorova

Departament de Traducció i Ciències
del Llenguatge
TESI DOCTORAL UPF / 2009
Director de la tesis: Prof. PhD Toni Badia

December 28, 2009

To the memory of my grandparents,
Katerina Fadeeva and Anatoli Konkin.

0.1 Acknowledgements

As a PhD student I have been working on interesting problems and visiting excellent research institutions. These have been wonderful four years, and I am grateful to my advisers and colleagues, senior and eminent as well as junior and totally disreputable yet none the less very inspiring. I thank Prof. Robert M. Gray from Stanford University for his advice in many research and administrative problems and for being my host adviser during my visit to Stanford University. I particularly enjoyed my brief stay with the Lab of Human-Machine Communication at Technical University of Munich (chair of Prof. Gerhard Rigoll) and I am grateful for many suggestions and for further research ideas they generously gave me. I thank Prof. Richard Olshen from Stanford University for his encouragement and comments that definitely made my work better style. I thank my scientific supervisor Prof. Toni Badia. I am grateful to my two host advisers, Prof. Elisabeth André from University of Augsburg and Prof. Dietrich Klakow from the University of the Saarland. I thank Prof. Elisa Barney from Boise State University for her useful comments.

I acknowledge the financial support by the Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR) for the FI grant, also for the two travel grants to make the research stays at the University of the Saarland (BE-DRG 2007) and University of Augsburg (BE-DRG 2008), and to the University Pompeu Fabra together with Stanford University for the funds to cover my Stanford visit (ajut estada DTCL and Prof. Gray's research funds as Lucent Technology professor).

Thanks to my family: mama, Sergei, Alena, Pusa, Sery, and my friend Bene, who joined me in my journey through emotional patterns.

0.2 Abstract

- Hay tres aspectos innovadores. Primero, un algoritmo novedoso para calcular el contenido emocional de un enunciado, con un diseño mixto que emplea aprendizaje estadstico e información sintáctica. Segundo, una extensión para selección de rasgos que permite adaptar los pesos y así aumentar la flexibilidad del sistema. Tercero, una propuesta para incorporar rasgos de alto nivel al sistema. Dichos rasgos, combinados con los rasgos de bajo nivel, permiten mejorar el rendimiento del sistema.

- The first contribution of this thesis is a speech emotion recognition system called the *ESEDA* capable of recognizing emotions in different languages.

  The second contribution is the classifier *TGI+*. First objects are modeled by means of a syntactic method and then, with a statistical method the mappings of samples are classified, not their feature vectors. The *TGI+* outperforms the state of the art top performer on a benchmark data set of acted emotions.

  The third contribution is high-level features, which are distances from a feature vector to the tree automata accepting class i, for all i in the set of class labels. The set of low-level features and the set of high-level features are concatenated and the resulting set is submitted to the feature selection procedure. Then the classification step is done in the usual way. Testing on a benchmark dataset of authentic emotions showed that this classification strategy outperforms the state of the art top performer.

0.3 Prologue

The aim of a speech emotion recognizer is to produce an estimate of the emotional state of the speaker given a speech fragment as an input. In other words we seek a solution for the tricky problem: given a speech fragment how to know what the speaker is feeling, even if she did not intend us to know that.

How would it be possible to construct such recognizers? Intuitively, when emotion is experienced, there are physiological changes, for example faster heart rate, higher blood pressure, faster breath rate, tension of certain muscles, and so forth. Some of these physiological changes affect speech production organs and they are in a state different from normal, therefore the speech signal comes out of the mouth distorted as compared to emotionally neutral. Different emotions trigger different physiological changes – one feels differently when bored compared to when scared. Since the changes are typical and differ from emotion to emotion, the speech waves produced under the effects of different emotions are distorted in predictably different ways. Our intuition tells us the following procedure might lead to a solution of the problem. We can record speech in different emotional states, measure acoustic parameters of the wave, form feature vectors from these measurements, and then we hope that pattern recognition techniques would do their job well, the way they recognize different iris flowers based on the shape of petals and colour. Instead of shape and colour we have parameters of intensity, pitch, formants and so on – the acoustic parameters that are well studied in speech recognition and synthesis.

Speech emotion recognition is a young interdisciplinary research field[1]. For the first time the above described methodology was given a try not earlier than a decade ago. The first systems worked in lab conditions and were quite different from what can be used in real life applications. Since then mathematicians, engineers, psychologists and various speech experts united their efforts in designing working SER systems, and significant progress has been made. Some old challenges have been successfully faced, many others remain. New applications keep appearing and in their turn pose new open problems and challenges. This year at the biggest annual speech conference *INTERSPEECH 2009*, several leading researchers of the field started a centralised event called the *INTERSPEECH Emotion Challenge*, aimed at providing an assessment settings and comparing different SER systems. They also challenged the SER community with the Hilbert-like list of problems:

- How to achieve high and robust performances? (the open system challenge),
- New high-level preferably perceptually adequate features are sought after (the open feature challenge),
- How to craft classifiers for SER and go beyond the main-stream libraries for classification? (the classifier challenge)

In this work I give my answers to these questions.

[1]Henceforth *speech emotion recognition* is abbreviated to SER.

Contents

0.1 Acknowledgements II
0.2 Abstract III
0.3 Prologue IV

1 Introduction **1**
1.1 Hypotheses 3
1.1.1 *ESEDA*: classical speech emotion recogniser and module for error prevention 3
1.1.2 The *TGI+* classifier 4
1.1.3 The high-level features based on distances to tree automata . 5
1.2 Objectives 5

2 State of the Art **7**
2.1 Why SER is important and difficult 7
2.2 Classifiers 9
2.2.1 Preliminaries 9
2.2.2 Combined Classification Strategy from OCR 11
2.2.3 Learning Decision Trees 19
2.3 Databases of Emotional Speech 22
2.3.1 Berlin Emotional Database: EMO-DB 23
2.3.2 Interface databases 23
2.3.3 FAU Aibo database 23
2.4 Considerations for SER system design 24
2.4.1 Considerations for Feature Extraction module 24
2.4.2 Considerations for Feature Selection 25
2.4.3 Considerations for Classification Module 26
2.4.4 Multilingual SER 26

3 *ESEDA* system **29**
3.1 Basic System Architecture 29
3.1.1 Testing for the basic recognizer 33
3.2 Module of Error Analysis and Prevention 36
3.2.1 Description of the additional block 36
3.2.2 Example 39
3.2.3 Testing for the module of error prevention 39
3.3 Conclusions 40

4 *TGI+* classifier **41**
4.1 Departure point 41
4.2 *TGI+* algorithm 42
4.2.1 Informal description of the algorithm 42
4.2.2 Formal definition of the algorithm 44
4.2.3 Tree Grammar Inference Algorithm 46
4.2.4 Edit Distance Calculation Algorithm 47
4.3 Extension of the general scheme: TGI+.2 49
4.4 Experiments 50
4.5 Main property of *TGI+* 50
4.6 Discussion 52
4.6.1 Correctness of algorithm construction 52
4.6.2 Selection of C4.5 as a base classifier in *TGI+* 53
4.7 Conclusions 53

5 Distance-to-Automaton Features **55**
5.1 Low-level features 58
5.2 Distance to tree automata features 58
5.2.1 Calculation of Distance-to-Automaton Features 59
5.3 Experiments 60
5.4 Results and Discussion 60
5.4.1 Conclusions 61

6 Results **63**
6.1 The *ESEDA* system and learning from classification errors 63
6.2 *TGI+* classifier 63
6.3 High-level features based on distances to tree automata 64

7 Future Work **65**
7.1 Medical Application 65
7.1.1 Task Formulation 65
7.1.2 *TGI+ cognitive* 66
7.1.3 Data 67
7.1.4 Hypothesis 67
7.1.5 Usability 67
7.2 Future Research 67
7.2.1 Feature Selection Embedded in Tree Structure 67
7.2.2 Towards a more cognitive algebra for $TGI+_c$ 68

Appendices **68**

A Three classifiers **69**
A.1 Multilayer Perceptron 69
A.2 Support Vector Machines 74
A.3 $RIPPER_k$ 75

B Low level features **77**

C Publications 83

List of Figures

1.1 Interconnections of different sections of the thesis. 2

2.1 The Aibo robot. [image taken from www.inf.ed.ac.uk/postgraduate/msc.html 8
2.2 The canonical model of the classifier. 10
2.3 An example for the $q - tree$ algorithm. 12

3.1 The scheme for the classification decomposition. 37
3.2 The flowchart for the *ESEDA*. 38

4.1 The *TGI+* steps. 43
4.2 String comparison on strings (A) and vectors (B). 48
4.3 The training/testing protocol for the *TGI+*. 51

5.1 Calculation of the distance-to-automaton features. 56
5.2 The classification scheme with early fusion. 57

A.1 The computational model of the neuron. 70
A.2 Multilayer perceptron. 71

List of Tables

3.1 The accuracy on English, Slovenian, French and Spanish. 33
3.2 Basic Recognition for French. 33
3.3 Basic Recognition for English. 34
3.4 The confusion matrix for Slovenian. 34
3.5 Basic Recognition for Spanish. 34
3.6 The confusion matrix for multilingual SER. 34
3.7 Precision, Recall, and F-measure for multilingual emotion recognition. 35
3.8 Recognition with the MLP on the EMO-DB. 35
3.9 Precision, Recall and F-measure for the Aibo corpus. 35
3.10 The confusion matrix for the Aibo corpus. 36
3.11 The Minority Class Problem treatment. 40
3.12 The improvements due to error-prevention. 40

4.1 Different values of $k = \frac{w_n}{w_s}$. 50
4.2 The C4.5 on the EMO-DB. 51
4.3 The MLP on the EMO-DB. 52
4.4 The *TGI+* on the EMO-DB. 52

5.1 Accuracies on the low-level and the fused vectors. 61

Chapter 1

Introduction

This chapter provides a reading guide over this thesis and formulates the objectives and the hypotheses. In this thesis I claim three contributions:

- a speech emotion recogniser capable of recognizing acted and authentic emotions in various languages,
- a classification method for SER, and
- high-level features for SER.

For each of the above listed contributions, there is a paragraph in Section 1.1 *Objectives* describing my idea and my departure point in the state of the art research. A reading map in Figure 1.1 shows interconnections between sections and chapters, where the sections with contributions are coloured green.

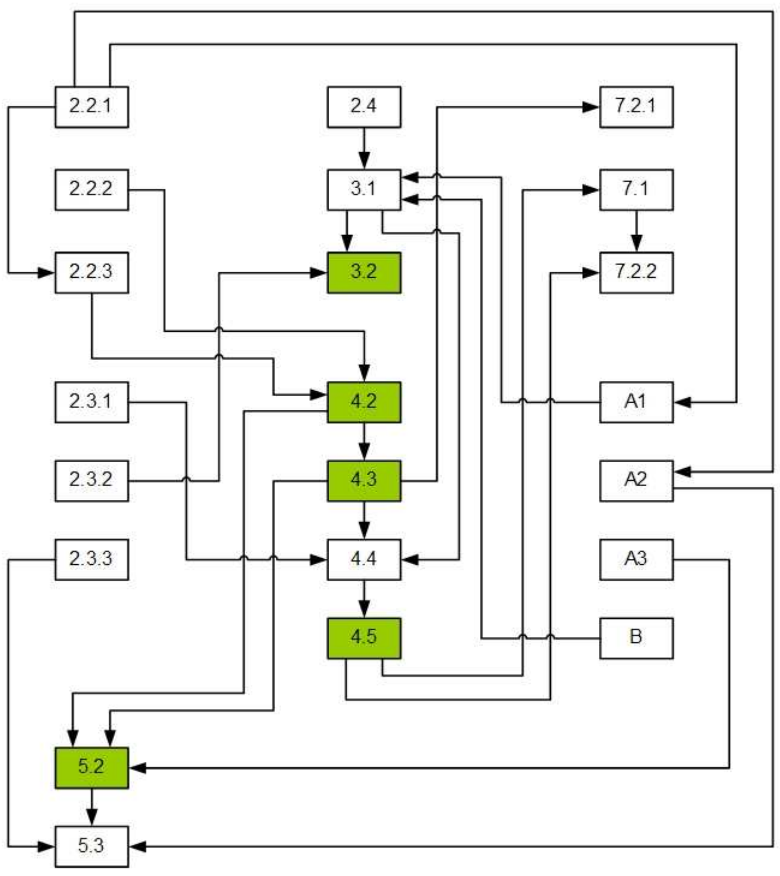
Chapter 2 is a literature review of relevant topics. In Section 2.1 I answer the question *Why SER is important and difficult?* and summarize SER applications. Section 2.2 contains a selection of relevant topics on classification including a complete account of the work in optical character recognition that gave me the general idea for the proposed classification method. In Section 2.3, I describe the databases I use. Section 2.4 gives considerations for the design of different modules of a SER recognizer.

Chapter 3 describes the *ESEDA* system[1] that I developed for this thesis project. Section 3.1 *Basic System Architecture* explains the design of the three modules: feature extraction, feature selection and classification. In Section 3.1.1 I report the testing results for the basic recogniser on different databases in mono- and multilingual modes. Section 3.2 *Module of Error Analysis and Prevention* describes my idea of doing classification decomposition based on the analysis of the confusion matrix. The idea is explained in Section 3.2.1 *Description of the additional block*, then in Section 3.2.2 *Example* I provide an example of how it works and finally the testing results are reported in Section 3.2.3. In Section 3.3 I draw conclusions with respect to the first contribution of this thesis.

In Chapter 4, the *TGI+* classifier [61] is explained. In Section 4.1 I refresh the idea of a combined classification strategy from optical character recognition, which was my starting point. In Section 4.2 I informally introduce and formally define

[1]The abbreviation stands for *E*nhanced *S*peech *E*motion *D*etection and *A*nalysis.

Figure 1.1: Interconnections of different sections of the thesis.



the proposed classification method. In Section 4.3 I explain my further extension of the *TGI+*. In Section 4.4, I report the experimental results, which I then discuss in Section 4.6 and on the basis of which in Section 4.7 I conclude that the *TGI+* is a classifier with a righteous place among other classifiers that are recommendable for SER.

In Chapter 5, the high-level features are proposed [62]. In Section 5.2 the proposed algorithm for high-level feature construction is explained. In Section 5.3 I describe the data set and experiments to test the idea. Section 5.4 is discussion and conclusions for the third contribution of this thesis.

In Chapter 6 I draw conclusions for this thesis.

In Chapter 7 I discuss the potential impact of my contributions, both in applications and research.

In Appendix A I include the explanations of the multilayer perceptron, the support vector machines, and the $RIPPER_k$. These are the top performers on the databases I work with and therefore serve as baselines for the proposed classification methods.

Appendix B provides a full account of the features that the feature extraction module extracts.

Appendix C is the list of my papers that cover my contributions defended in this thesis.

1.1 Hypotheses

The starting point of my research was construction of the basic emotion recogniser capable of recognizing acted and authentic emotions in various languages. Its performances will serve as a baseline to validate the theory proposed in this thesis.

1.1.1 *ESEDA*: classical speech emotion recogniser and module for error prevention

The first contribution concerns practical work and did not imply the development of any new theory.

After having constructed the basic speech emotion recogniser, I analysed the accuracy and realised that there is room for improvement. Certainly the desired improvement can be obtained in more than one way. For example, a direct solution is a revision of every module of the basic recognizer: feature extraction, feature selection or classification. That for example could be extracting more signal features, or trying all the available feature selection algorithms on the validation set and choosing the one that leads to the best accuracy, and so forth. The weakness of this solution is that it can turn out to be database-dependent, since generally there is no way to be sure which are the best features or the best classifier a priori before having seen the type of speech data: e.g. speech of a German child interacting with a toy is unlikely to show the same trends as Spanish data from a call centre. A data-independent solution would be of more use – a *wrapper* machine learning technique without reprogramming the basic recognizer. From the rich palette of machine learning methods, I take classification decomposition[2]. A decomposition splits a complete multiclass problem into a set of smaller classification problems. Decompositions allow for learning more accurate concepts due to simpler classification boundaries in subtasks and the feature selection procedure done individually for each classification step [12]. When doing classification decomposition, the central choice is the order of combinations of smaller classification steps, called the classification path. My idea is to derive the classification path from the confusion matrix and, after uncovering the reasons for errors, design a module that prevents the system from making such errors in the future.

Another fact that is necessary to take into account at the training stage is that not all emotions are equally frequent in data sets from real-life applications. A class with few samples is called the minority class. The rarity of samples with this class value in the training data does not mean that the emotion is unimportant. On the contrary it may signify an exceptional situation and therefore require accurate detection, while the classifier can be biased towards a more frequent class.

To counter the low separability of some classes in the feature space and the minority class problem, the error-prevention module implements the following strategy:

1. The class of special interest is identified (denote it class I), for which the recognition rates are to be improved. For example it can be the worst recognised class or a class of special interest for some application. From the confusion matrix of the standard classification step it is deduced with which other class the class of interest is most frequently confused (denote it class J). The original classification step is then divided into two new steps: the first multiclassification step, where the new class labels are the old ones, except that there is a joint label K for samples from the classes I and J, and the second binary classification step, where instances of class K are classified into I or J.

2. If the minority class problem is present and hampers the classification accuracy, cost-sensitive training is used, more specifically, every minority class sample in the database is duplicated.

My intuition is that the above formulated wrapper procedure derived from the confusion matrix will improve the recognition accuracy of the basic speech emotion recognizer. In Section 3.2.3 *Testing for the module of error-prevention* I describe the experiments and draw conclusions in Section 3.3.

1.1.2 The *TGI+* classifier

In the optical character recognition literature [58] it was reported that a combination of statistical and syntactic pattern recognition techniques led to significant improvements in accuracy as compared to "mono-" methods and a new classification method was proposed that combined tree grammar inference and entropy decision trees in a coherent learning strategy.

My *first hypothesis* is that the above mentioned approach from optical character recognition is adaptable to speech emotion recognition and will lead to accurate recognition results.

[2]The approach is sometimes reffered to as hierarchical or tree structured classification.

I propose a classification method [61], the general idea of which comes from optical character recognition. The syntactic part implements tree grammar inference, and the statistical part implements an entropy decision tree classifier. First, the objects are modeled by means of a syntactic method, that is samples are mapped into their representations. A representation of a sample is a set of numeric values, signifying to which degree a given sample resembles the averaged pattern of each of the recognition classes. Then, the mappings of samples are classified, not their initial feature vectors, with a statistical method. The new domain required a revision of every algorithm involved in the syntactic phase. I called the classifier *TGI+*, which stands for Tree Grammar Inference and the plus is for the statistical learning enhancement.

My *second hypothesis* is that the combined scheme can be beneficially extended with a built-in feature selection procedure, which would result in weights put on nodes corresponding to the selected features in the tree representation. My idea is to apply some standard feature selection procedure and then according to its results add various edit costs to penalise more important features with higher edit costs for being outside the interval which the tree automata learned at the inference stage.

1.1.3 The high-level features based on distances to tree automata

As was formulated in the list of challenges for the community at the INTERSPEECH-2009 conference, currently one of the central SER open problems is the search for novel features, especially perceptually adequate and high-level ones.

The classifier from the previous section can be viewed as the C4.5 run on the high-level distance-to-automaton features. My hypothesis is that these high-level features can be useful when early-fused with the low level features from which they were calculated.

Thus I propose the high-level features, which are distances from a feature vector to the tree automata accepting class i, for all i in the set of class labels. The automata are trained to operate on feature vectors through a previously described grammar inference procedure. The set of low-level features and the set of high-level features are concatenated and their union is submitted to the feature selection procedure. Then the classification step is done in the usual way.

1.2 Objectives

Thus, the objectives to be achieved for this thesis are:
with respect to the system construction

- build a robust system to be exposed to acted and authentic emotions in different languages;

with respect to the proposed classifier

- adapt the new combined classifier from optical character recognition to the new domain of speech emotion, which will take revision of some its algorithmic components;

- extend the initial algorithm with a built-in feature selection which will result in different weights put on different nodes in the tree representation.

with respect to the high-level features

- propose new high-level features for SER and show that their fusion with the initial features is beneficial in terms of recognition rates.

Chapter 2

State of the Art

This chapter provides a review of relevant topics. In Section 2.1 I answer the question *Why SER is important and difficult?* and summarize some SER applications. The focus of Section 2.2 is relevant issues in pattern recognition. In Section 2.3, the databases I use are described. Section 2.4 gives considerations concerning the design of different modules of a SER recognizer.

2.1 Why SER is important and difficult

SER is *important* because it is an intriguing research field about human cognition. In the present days cognitive sciences are gaining in importance. The XX century with its major mathematical breakthroughs gave us the techniques and allowed for the luxury to turn our attention to more humanistic problems:

- how to be more attentive to the emotional needs of others;
- how to teach the machines and the sick, who for some reason are incapable of conveying or understanding emotions, to become competent in fully-human communication.

Better understanding between people replaces conflict with cooperation. Morally and financially our society is in need for such *affective* technology. Despite only a decade of history, SER has already been integrated into many useful applications.

In *smart call-centers*, SER helps to detect potential problems that arise from an unsatisfactory course of interaction. A frustrated customer is typically offered the assistance of human operators or some reconciliation strategy [14], [8], and [9].

In *intelligent spoken tutoring systems*, detecting and adapting to student's emotions is considered to be an important strategy for closing the performance gap between human and computer tutors [1]. Studies in educational psychology point out that emotions can impact a student's performance and learning.

In *spoken dialogue research*, it is beneficial to enable the systems not only to recognize the content encoded in a user's response, but also to extract information about the emotional state of the user by analyzing how these responses were spoken.

In *human-robotic interfaces*, robots can be taught to interact with humans and recognize human emotions. Robotic pets, for example, should be able to understand

Figure 2.1: The Aibo robot. [image taken from www.inf.ed.ac.uk/postgraduate/msc.html

not only spoken commands, but also other information, such as the emotional and health status of its human commander and modify their actions accordingly. For example, [28] constructed a robot capable of detecting and moderating tension. The Aibo robot has the potential to let even dogs benefit from modern technological advances (Figure 2.1).

Reasons why *SER is difficult* include:

- noisy data;

Noise is defined in very general terms [15]: any property of the pattern which is not due to the true underlying model, but instead to randomness in the world or sensors.

- emotional expressions are discrete;

Unlike expressive imitations of emotions by actors, everyday emotions are hard to detect by humans and computers alike.

- diversity of patterns;

It is hard to build systems that would work well without retraining when exposed to new tasks than those for which they were trained. For example, different languages have different emotional patterns, and in distinct circumstances people express emotions differently: at work during project meetings as contrasted to at home when talking to the family.

2.2 Classifiers

In the Section 2.2.1 *Preliminaries* I introduce the general framework of classification. In Section 2.2.2 a hybrid classification solution is explained, which is a starting point of the work on the *TGI+* I describe in Chapter 4. Section 2.2.3 introduces the general framework of classification trees with one such particular algorithm C4.5 detaily explained in Section 2.2.3. The C4.5 is given this attention because it is a building block of the proposed *TGI+* classifier.

Additionally in Appendix A I include the explanations of the multilayer perceptron, the support vector machines, and the $RIPPER_k$. These are the top performers on the databases I work with and therefore serve as baselines for the classification methods I propose.

2.2.1 Preliminaries

A classifier assigns class labels to objects. Objects are described by a set of measurements called attributes or features. In SER objects are the speech fragments. Features are signal statistics extracted from the utterance. Classes correspond to a set of emotions.

Let there be c possible classes in the problem, with labels from:

$$\Sigma = \{w_1, w_2, ..., w_c\}.$$

Feature values for a given object form an n-dimensional vector:

$$\mathbf{x} = [x_1, ..., x_n].$$

The real space $\Re^n$ is called the feature space, with every axis corresponding to a particular feature. The data set is denoted as,

$$\mathbf{Z} = \{z_1, ..., z_N\}, \text{ where } z_j \in \Re^n.$$

Let $l(z_j)$ denote the class label of z_j, and $l(z_j) \in \Sigma$, where $j = 1, ..., N$. A classifier D is a function:

$$D : \Re^n \longrightarrow \Sigma.$$

In the canonical model of the classifier Figure 2.2 a set of c discriminant functions is considered:

$$G = \{g_1(x), ..., g_c(x)\}, \text{ where } g_i : \Re^n \longrightarrow \Re, \text{ for all } i = 1, ..., c.$$

Every function yields a score for the representative class, and x is assigned the label of the highest scoring class. The maximum membership rule can be stated as

$$D(x) = w_j \Leftrightarrow j = argmax_i g_i(x). \quad (2.1)$$

Ties are resolved randomly, that is x is assigned randomly to one of the tied classes. The discriminant functions divide the feature space $\Re^n$ into c, not necessarily compact, classification regions, $R_1, ..., R_c$:

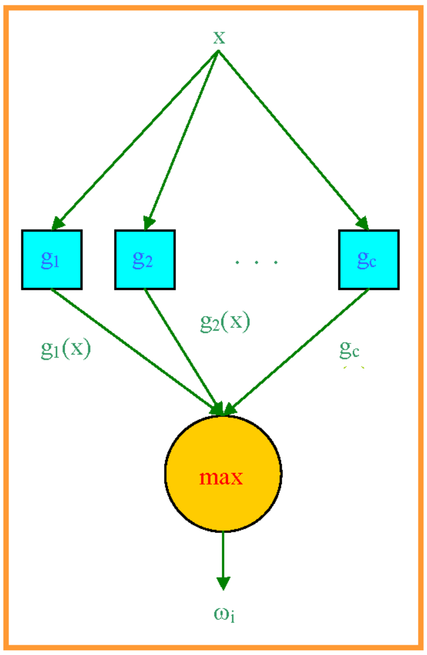


Figure 2.2: The canonical model of the classifier.

$$R_c = \{x \mid x \in \Re^n, g_c(x) = max\ g_k(x),\ k = 1, ..., c\},\ i = 1, ..., c.$$

The points for which the i^{th} discriminant function has the highest score constitute the decision region for class w_i. By the maximum membership rule (Equation 2.1), all points in decision region R_i are assigned to class w_i. The decision regions are defined by classifier D, or, equivalently, by discriminant functions G. The classification boundaries of the decision regions contain the points for which two or more discriminant functions return the same value. A point on the boundary can be assigned to any of the adjacent classes. The classes w_i and w_j are said to be overlapping if a decision region R_i contains data points from the labelled set **Z** with true class labels w_j, with $i \neq j$. If classes overlap given a particular partition of the feature space, they also can be non-overlapping if the feature space were to be partitioned differently.

To know exactly how accurate D is, it must be run on all possible input objects. Since this is not possible, the counting estimator $Error(D)$ is used to characterize accuracy. Assume a labelled data set Z_{test} with N_{test} objects in it is available for testing the accuracy of a classifier D. After running D on Z_{test}, calculate $Error(D)$:

$$Error(D) = \frac{N_{error}}{N_{test}}\ ,$$

where N_{error} is the number of misclassifications made by D, and N_{test} is the total number of objects in the test data set.

Another question is which subsets of **Z** it takes for training and testing purposes. In this thesis, I invariably use *cross-validation (π- or rotation method)*: The data set

$\mathbf{Z}$ is randomly divided into K subsets of size $\frac{N}{K}$, where the integer K is preferably a factor of N. One subset is then used to test the performance of D trained on the union of the remaining $(K-1)$ subsets. This procedure is repeated K times and the average error over all K trials is taken as $Error(D)$.

To know how the misclassifications are distributed across the classes, confusion matrices are built for Z_{test}. The entry a_{ij} of a confusion matrix denotes the number of elements from Z_{test} whose true class is w_i, and which are classified by D as belonging to the class w_j.

2.2.2 Combined Classification Strategy from OCR

In this section I describe a classification method originally devised for optical character recognition (OCR) of hand-written isolated digits from 0 to 9 [58], which was the departure point in my work on the classification method for SER. The general idea is to first model the objects by a tree and then use other pattern recognition techniques: k-neighbour, clustering, etc. [73], [74], [75], [44]. A particular implementation of this general idea for OCR in question is organized in three stages:

1. Through a grammar inference technique the system learns a set of tree automata (one per digit).

2. The system obtains a set of edit distances of every digit to every tree automaton.

3. According to a set of rules based on distances the system learns a decision tree from the last set of distances that will classify the digit.

In the following paragraphs I explain these stages in greater detail.

Learning Tree Automata

There are three algorithms involved in learning tree automata:

1. an algorithm to obtain tree-representations from physical objects;

2. an algorithm to obtain a distance between a tree and a tree automaton;

3. a tree grammar inference algorithm.

Converting physical objects to suitable representation

Figure 2.3 shows a scanned handwritten digit 2. Under the approach being explained, digits must be converted into their tree representations with the *q-tree* [24]. A *q-tree* is constructed by drawing a square window around the digit and splitting the window into four windows of the same size recursively up to a predefined depth. In Figure 2.3 the digit has been placed in the recursively split window of depth 3. Once the system obtains the window of the digit, it assigns a label to the window of the smallest size. The label is chosen depending on the scale of gray: black, white or gray. So, every smallest window is represented by a label from the three letter alphabet $\{a, b, c\}$. The relations between windows can be represented by a tree using the up-down and left-to-right scanning of the q-tree.

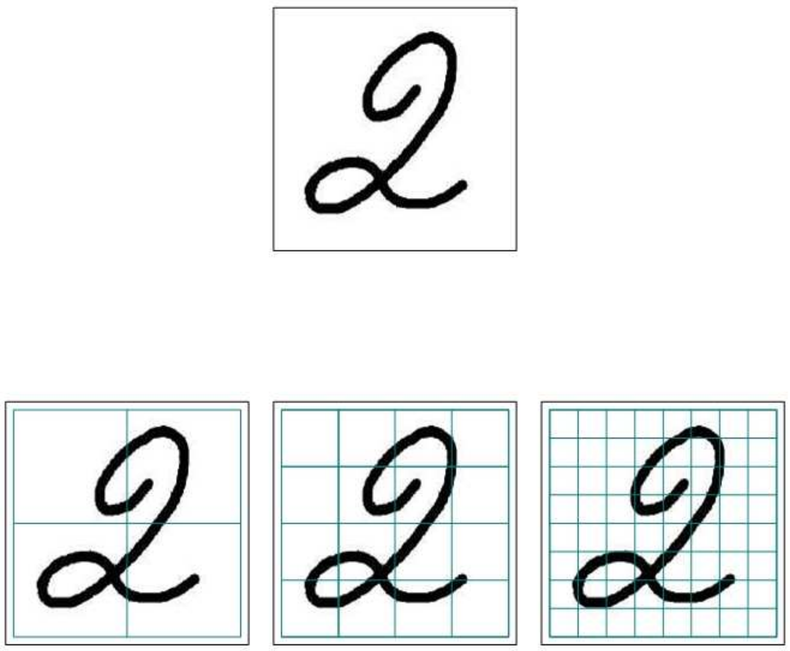


Figure 2.3: An example for the $q - tree$ algorithm.

After having converted hand-written digits into suitable representations, grammar inference is used to construct a tree automaton from every set of trees representing the same digit. Grammar inference [51] is an inductive approach to represent recognition classes as grammars in the sense of a term-rewriting system and learn such grammars from examples available for the recognition classes.

An algorithm from [34] is used to obtain the automata from examples.

In this section I provide a description of this polynomial time algorithm to calculate a distance between a tree and a tree automaton [34]. Below I include a few necessary definitions, and for a broader picture of Tree Automata theory and applications, the reader is referred to [13].

Assume we are given an alphabet V, and a set of natural numbers N.

A ranked alphabet is defined as the association of V with a finite relation r in $(V \times N)$.

$$V_n := \{\sigma \in V \mid (\sigma, n) \in r\}. \tag{2.2}$$

V^T is the set of finite trees whose nodes are labelled with symbols in V.

A *tree* is inductively defined as follows:

$$V_0 \subseteq V^T; \tag{2.3}$$

$$\sigma(t_1, \ldots, t_n) \in V^T : \tag{2.4}$$

for $\forall t_1, \ldots, t_n \in V^T$, and $\sigma \in V_n$.

Given a tree t, the set of *subtrees* of t is denoted as $Sub(t)$:

$$Sub(a) = \emptyset, \tag{2.5}$$

for $\forall a \in V_0$;

$$Sub(\sigma(t_1, \ldots, t_n)) = \{t_1, \ldots, t_n\} \bigcup_{i=1..n} Sub(t_i), \tag{2.6}$$

for $\forall\ t_1, \ldots\ , t_n \subseteq V^T$, and $\sigma \in V_n$.

The *root* of the tree, $root(t)$, is defined as:

$$root(a) = a, \tag{2.7}$$

for $\forall a \in V_0$;

$$root(\sigma(t_1, ..., t_n)) = \sigma, \tag{2.8}$$

for $\forall t_1, ..., t_n \in V^T$, and $\sigma \in V_n$.

The *depth* of the tree, $depth(t)$, is defined as:

$$depth(a) = 0; \tag{2.9}$$

$$depth(\sigma(t_1, ..., t_n)) = 1 + max\{depth(t_i)\} \tag{2.10}$$

for $i = 1, ..., n$, and $\forall t_i \in V^T$, where $1 \leq i \leq n$, and $\sigma \in V_n$.

Subtrees for which $depth(t) \geq 1$ are called *successors* of t and denoted H^t:

$$H^\sigma(t_1, ..., t_n) = \langle root(t_1), ..., root(t_n) \rangle : \tag{2.11}$$

for $\forall t_1, ..., t_n \in V^t$, and $\sigma \in V_n$.

Let the *size of the tree t*, denoted by $|t|$, be defined inductively as

$$|a| = 1, \tag{2.12}$$

for $\forall a \in V_0$.

$$|\sigma(t_1, ..., t_n)| = 1 + \sum_{i=1...n} |t_i| : \tag{2.13}$$

for $\forall t_n \in V^T$, and $\sigma \subseteq V_n$.

Given a tree t, let $leaves(t)$ be the set defined as follows:

$$leaves(a) = \{a\}, \tag{2.14}$$

for $\forall a \in V_0$;

$$leaves(\sigma(t_1, \ldots, t_n)) = \bigcup_{i=1..n} leaves(t_i), \tag{2.15}$$

for $\forall t_1, \ldots, t_n \in V^T$, and $\sigma \in V_n$.

Σ denotes the set of $a \in V_0$ and is called the *leaf alphabet*.

Let the *deterministic tree automaton*[1] be defined as a tuple:

$$A = (Q, V, \delta, F), \tag{2.16}$$

where:
Q is finite set of states;
V is a ranked alphabet with $Q \cap V = \emptyset$;
$F \subseteq Q$ is a set of final sates;
$\delta = (0, \ldots, m)$ is a finite set of transitions defined as follows:
δ_n: $(V_n \times (Q \cup V_0)^n \rightarrow Q)$, where $n = 1, \ldots, m$.
$\delta_0(a) = a$, for $\forall a \in V_0$.

The relation δ can be extended to operate on trees:

$$\delta : V^T \rightarrow Q \cup V_0, \tag{2.17}$$

$$\delta : (\sigma(t_1, \ldots, t_n)) = \delta_n(\sigma(\delta(t_1), \ldots, \delta(t_n))), \tag{2.18}$$

if $n \geq 1$,

$$\delta_0(a) = a. \tag{2.19}$$

A given tree $t \in V^T$ is accepted by A, if $\delta(t) \in F$.

Given the state $q \in Q$, we define the *ancestors* of the state q:

$$Ant(q) = \{\langle p_1, ..., p_n\rangle | p_i \in (Q \cup V_0) \wedge \delta_n(\sigma, p_1, ..., p_n) = q\}. \tag{2.20}$$

Let σ_i^t be the i^{th} node of tree t in post order enumeration, which means that on the same level the nodes are ordered from left to right and the levels are passed in bottom-up manner. H_i^t is the string formed by the successors of σ_i in t.

Edit costs between trees are defined as follows:

Insertion costs are defined as:

$$B(a) = 1, \tag{2.21}$$

for $\forall a \in \Sigma$;

$$B(\sigma(t_1, ..., t_k)) = 1 + \sum_{\forall j} B(t_j). \tag{2.22}$$

Deletion costs are defined as:

$$I(a) = 1; \tag{2.23}$$

$$I(\sigma(t_1, ..., t_k)) = 1 + \sum_{\forall j} I(t_j). \tag{2.24}$$

Substitution costs are defined as:

$$S(a, a) = 0 \tag{2.25}$$

[1]further in the text abbreviated to TA

and

$$S(a,b) = 1 \tag{2.26}$$

for $\forall a, b \in \Sigma$.

$$S(\sigma(t_1, ..., t_k), a) = B(\sigma(t_1, ..., t_k)) + I(a); \tag{2.27}$$

$$S(a, \sigma(t_1, ..., t_k)) = B(a) + I(\sigma(t_1, ..., t_k)). \tag{2.28}$$

The algorithm explores the tree and for $\forall \sigma_i \in t$ calculates the cost to reduce the tree S_i^t to every state of the automaton. Given a node of a tree σ_i:

$$S_i^t = \sigma(t_1, ..., t_n) \tag{2.29}$$

and a state $q \in Q$, the strings involved in comparison are:

- the ancestors of each state of the automaton $Ant(q)$ and
- the sequences
$$H_i^t = \langle h_1, ..., h_n \rangle, \tag{2.30}$$
where $h_i \in \{t_i\} \times Q$, if $Depth(t_i) \geq 1$, $h_i = (t_i, t_i)$.

$\hat{H}_i^t$ extends the notion of H_i^t to a chain of states instead of tree nodes. This measure is stored in a matrix D_A indexed by the states and the tree nodes. To carry out this calculation it is necessary to extend the edit operations to take into account every possibility. Instead of what was defined in Equations 2.23 - 2.24 for *delete*:

$$I(q) = Min\{I(t) | \delta(t) = q\}, \tag{2.31}$$

for $\forall q \in Q$. Instead of what what was defined for *insert* above in Equations 2.21 - 2.22

$$B(t, q) = B(t), \tag{2.32}$$

for $\forall q \in Q$, and $\forall t \in V^T$. For substitution instead of Equations 2.25 - 2.29,

$$S((a, a), b) = S(a, b), \tag{2.33}$$

for $\forall a, b \in V_0$.

$$S((a, a), q) = B(a) + I(q), \tag{2.34}$$

for $\forall q \in Q$, and $\forall a \in V_0$.

$$S((t, q), p) = B(t) + I(p). \tag{2.35}$$

The method visits the tree nodes in post-order and at each node carries out distance calculations.

Given a generic string edit distance calculation algorithm D_c, e.g. [69], instead of D_c we have $\hat{D}_c$:

$$\hat{D}_c : (V^T) \times (Q \cup V^T))^* \times (Q \cup V_0) \longrightarrow N. \tag{2.36}$$

The algorithm maintains the same algorithmic scheme, but using the above defined edit operations.

We can establish the distance between a tree t

$$t = \sigma(t_1, t_2, ..., t_k) \tag{2.37}$$

and a state q. If $depth(t) = 1$, then

$$D(t, q) = Min\{\hat{D}_c(\hat{H}^t, x) \mid x \in Ant(q)\}. \tag{2.38}$$

Otherwise,

$$D(t, q) = Min\{\hat{D}_c(\langle t_1, q_{i1}\rangle, ..., \langle t_k, q_{ik}\rangle, x) + \sum_{j=1}^{k} D(t_j, q_{ij})\}, \tag{2.39}$$

where $x \in Ant(q)$, and $\langle(t_1, q_{i1}), ..., (t_k, q_{ik})\rangle \in \hat{H}^t$.

After the edit operations between a tree t and a state q have been defined, let the distance from a tree t to a TA A be the operations with minimum cost necessary to allow the TA to accept the tree:

$$D(t, A) = Min\{D(t, q|q \in F)\} \tag{2.40}$$

Under the dynamic programming scheme in post-order enumeration when a node σ_i

$$S_i^t = \sigma(t_1, t_2, ..., t_k) \tag{2.41}$$

is going to be analysed, every distance between S_i^t and the state of the automaton has already been calculated and stored in D_A.

A scheme of the algorithm:

```
Input: A finite tree automaton $A$ and a tree $t$.

Output: Distance from the tree $t$ to the automaton $A$.

Method: [Initialization]

$\forall \sigma_i \in t\{$

$\forall q \in Q\{$

$D_A[\sigma_i, q] = \infty$

$\}$

$\}$

[ in post order enumeration]
```

```
$\forall \sigma_i \in t\{$

$\forall q \in Q\{$

$D_A[\sigma_i, q] = Min(D_A[\sigma_i, q], Min_{\forall x \in Ant(q)}(\hat{D}_c(S_i^t, x))$

$\}$

$\}$

End Method.
```

3) Once the distance is defined, the learning method is an
error-correcting grammar inference technique [35].

The sequence of transitions applied to accept a tree is called the *path*. *The minimum cost path* Δ_t is a sequence of transitions that should be applied to obtain the minimum distance from t to A under the scheme described above. The set of edit operations with a minimum cost may not be unique, but this does not affect the inference algorithm.

For a subtree of t:

$$\tau_i = \sigma(\tau_1, ..., \tau_{ik}) \tag{2.42}$$

let $d(\tau_i) \in \Delta$ be the transition, which is applied to reduce τ_i:

$$\delta(\sigma, u_{i1}, \ldots, u_{ik}) = q_i, \tag{2.43}$$

where $u_{ij} = \tau_{ij}$, if $\tau_{ij} \in V_0$, otherwise $u_{ij} = q$, in case $d(\tau_{ij}) \equiv \delta(\sigma(z_1, \ldots, z_n)$.

Let the sum of every edit operation cost applied in Δ be called the *accepting cost*:

$$\Delta = \Delta^P \cup \Delta^N. \tag{2.44}$$

The accepting cost consists of two non-overlapping parts: the transitions that add to the error cost, Δ^N, and the transitions that do not add to the error cost, Δ^P.

Error-correcting tree language algorithm

Input: A set of samples $\{\zeta = t_1, \ldots, t_n\}$.

Output: A tree automaton $A = (Q, V, \delta, F)$, which accepts at least the sample set.

```
Method: [Initial automaton, which accepts $t_1$]
$Q = Sub(t_1) \cup \{t_1\}$
$V = leaves(t_1) \cup \{\sigma\}$
$F = \{t_1\}$
$\delta(a) = a, \forall a \in leaves(t_i)$

if $[u_1, \ldots, u_p \in Q]\sigma(u_1, \ldots, u_n) \in Q$
```

```
**then**
$\delta \supseteq \delta(\sigma, u_1, \ldots, u_p) = \sigma(u_1, \ldots, u_p)$
}
```

```
    [Processing other trees in the set $\zeta$]
    $\forall t \in \zeta : i : 2, ..., n\{$
    $A = Expand(t_i, A)$
    }

    EndMethod
```

Expand(t_i, A) method.

Input:
A tree $t = \sigma(\tau_1, ..., \tau_n)$.

A tree automaton $A = (Q, V, \delta, \{q_f\})$

Output: A tree automaton that accepts at least $\{t\} \cup L(A)$.

Variables:
$Red[1, \ldots, \mid t \mid]$ array of $Q \cup leaves(t)$
$A' = (Q', V', \delta', F')$

Method:

```
$Q' = Q$
$V' = V \cup leaves(t)$
$\delta' = \delta$
$F' = F$
Obtain the minimum cost path $\triangle_t$
for $\forall z \in Sub(t)\{$
$d_z = \delta(\sigma, u_{z1}, \ldots, u_{zp}) = q_z$
**if** $(z \in leaves(t))$
**then** $Red[z] = z$
**else** $Red[z] = q_z$
}
[in post order]
$\forall z = \sigma(z_1, \ldots, z_p) \in Sub(t)\{$
**if** $(d_z \in \Delta^N)$
**then** $\{Q' = Q' \cup \{q_N\}$ } [add a new state]
$\delta' \supseteq \delta(\sigma, Red[z_1], \ldots, Red[z_p]) = q_N$
**elsif** { $\exists d_{zj} \in \Delta_t^N$
$\delta' \supseteq \delta(\sigma, Red[z_1], \dots, Red[z_p]) = Red[z]$
}
}

    [Accept transition]
$\delta' \supseteq \delta(\sigma, Red[\tau_1], \dots, Red[\tau_n]) = q_f$
Return $A'$
EndMethod.
```

2.2.3 Learning Decision Trees

The above described learning process ends with every sample represented by a feature vector with ten features, which are distances to the ten tree automata for digits from 0 to 9. Then the C4.5 algorithm from the family of entropy decision tree classifiers is used to first create a decision tree and then to classify new samples based on its distances to the ten tree automata. The C4.5 is explained in the next section.

To conclude, the protocol for the optical character recognition was as follows:

1. obtain $q - tree$ representation of every digit in the data set;
2. divide the set into two disjoint subsets (Set 1 and Set 2), and with the error-correcting inference algorithm from Set 1 learn a tree automaton for every digit;
3. calculate the distance between every digit and every tree automaton;
4. from Set 1 ∪ Set 2 perform the learning and testing phase for decision trees.

C4.5

The C4.5 is a building block for the proposed classification strategy I described in Chapter 4. The C4.5 belongs to the family of algorithms that employ a top-down greedy search through the space of possible decision trees. A decision tree is a representation of a finite set of *if-then-else* rules. An alternative to the C4.5 is a generic tree growing framework [5] called CART.

The construction of the tree splits a given training set hierarchically until either all the objects within the same region have the same label or the subregion is pure enough. Consider a c-class problem with $\Omega = \{w_1, w_2, ..., w_c\}$. Let P_j be the probability for class w_j at node t, estimated as the proportion of points from the respective class from the data set which reached node t. The *impurity* of the distribution of the class labels at t can be measured with the entropy function:

$$i(t) = -\sum_{j=1}^{c} P_j log P_j. \tag{2.45}$$

Impurity has its maximum when the classes are uniformly distributed: $i(t) = \log c$. For a pure region with only one class label, $i(t) = 0$, that is takes its minimum.

Denote the set of classes: $\{w_1, ..., w_n\}$ by Ω, and the training set of records by T. The information needed to identify the class of an element of T:

$$Info(T) = -i(t) = I(P). \tag{2.46}$$

P is the probability distribution of the partition $(w_1, w_2, ..., w_k)$:

$$P = (\frac{w_1}{|T|}, \frac{w_2}{|T|}, ..., \frac{w_k}{|T|}). \tag{2.47}$$

If we partition T into sets $\{T_1, T_2, ..., T_n\}$ on the basis of some feature X:

$$Info(X,T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} Info(T_i). \tag{2.48}$$

The notion of *Gain* is used to rank attributes and to build decision trees where at each node the attribute is located with greatest gain among the attributes not yet considered on the path from the root:

$$Gain(X,T) = Info(T) - Info(X,T). \tag{2.49}$$

Equation 2.49 represents the difference between the information needed to identify an element T and the information needed to identify an element T after the value of attribute X has been defined, i.e. the gain in information due to the attribute X.

The C4.5 is the extension of the ID3 algorithm.

```
function ID3 ($R$: a set of features, $w$: a class label, $S$: the training set) return
a DT

    begin

    /* First special cases */

    If S is empty, return a single node with value Failure.

    If S consists of records all with the same value for the
    class value, return a single node for that value.
/* Now regular cases */

    Let D in R be the attribute with the largest $Gain(D, S)$;

    Let $\{d_j | j = 1, 2, ..m\}$ be the value of attribute D;

    Let $\{S_j | j = 1, 2, ..m\}$ be the subsets of S consisting
    respectively of records with value $d_j$ for attribute $D_j$;

    Return a tree with root labelled D and arcs labelled $d_1, d_2, ..., d_m$
    going respectively to the trees
    **ID3**$(R - \{D\}, w, S_1)$, **ID3**$(R - \{D\}, w, S_2)$, ..., **ID3**$(R - \{D\}, w, S_m)$;
    end ID3.
```

Having constructed an almost perfect decision tree we have to prune it to prevent overfitting. Pruning strategies can be different. Below we explain two of them.

Reduced Error Pruning (REP) is the simplest pruning method. It uses an additional training set, called the "pruning set", unseen during the growing stage.

1. A simple error check in the bottom-up manner is calculated for all non-leaf nodes.

2. Replace the node with a leaf and label it to the majority class, then calculate the error of the new tree on the pruning set:
   - If the error is smaller than the error of the whole tree on the pruning set, we replace the node with the leaf.
   - Otherwise, keep the subtree.

Pessimistic Error Pruning: the same data set is used both for growing and pruning the tree. Denote n the number of data points that reached node t and $e(t)$ the number of errors if t was replaced by a leaf. Let T_t be the subtree rooted at t, L_t be the set of leaves of T_t, be $l'(T_t)$ the number of errors of T_t with a complexity correction:

$$l'(T_t) = \sum_{l \in L_t} e(l) + \frac{|L_t|}{2}. \tag{2.50}$$

The node t is replaced by a leaf if

$$e(t) \leq e'(T_t) + \sqrt{\frac{e'(T_t[n - e'(T_t)])}{n}} - \frac{1}{2}. \tag{2.51}$$

Tree complexity is expressed as the number of leaves of the subtree. Thus a balance is sought between the error and the complexity.

The notion of *Gain* from Equation 2.49 tends to favour attributes that have a large number of values. For example, if we have an attribute D that has a distinct value for each record, then $Info(D, T) = 0$, and thus $Gain(D, T)$ is maximal. To counter this the *GainRatio* impurity was proposed:

$$GainRatio(D, T) = \frac{Gain(D, T)}{SplitInfo(D, T)} = \Delta_{iM} = \frac{\Delta_i}{-\sum_{i=1}^{M} P_i log P_i}. \tag{2.52}$$

$SplitInfo(D, T)$ is the information due to the split of T on the basis of the value of the class value D.

$$SplitInfo(D, T) = I(\frac{T_1}{T}, \frac{T_2}{T}, ..., \frac{T_m}{T}) \tag{2.53}$$

where $\{T_1, T_2, ...T_m\}$ is partition of T induced by the value of D.

On the way of evolution from the ID3 to the C4.5, a number of modifications were added:

- Missing values.
  - While building a decision tree, we can deal with training sets that have records with unknown attribute values by evaluating the *Gain*, or the *GainRatio* by considering only the records where that attribute is defined;
  - While using a decision tree, we can classify unknown attribute values by estimating the probability of the various possible results.

- Continuous values. To determine the best split point for the feature X we sort the values of X for all the points in the current node and check Δ_{iM} for a split point between every two consecutive values of X. The highest Δ_{iM} signifies the best split point.

- Pruning the decision tree. Pruning the decision tree is done by replacing a whole subtree by a leaf node. The replacement takes place if a decision tree established that the expected error rate in the subtree is greater than in the single leaf.

2.3 Databases of Emotional Speech

In the literature the issue of authenticity of speech material has enjoyed a vivid research interest. I can single out three types of databases used in SER:

- Type 1 is acted emotional speech with human labelling. They are obtained by asking an actor to speak with a predefined emotion, e.g. as in DES [16] or in EMO-DB [7].

- Type 2 is authentic emotional speech with human labelling. These databases come from real-life applications for example call-centres: [32], [1], [55], [64], [41], etc.

- Type 3 is elicited emotional speech with self-report instead of labelling, e.g. [29], where emotions are provoked and self-report is used for labelling control. In this case, no manual labelling is needed.

Different types of databases are suitable for different purposes. There are objections against the use of acted emotions. It was shown that acted and spontaneous samples differ in the view of features and accuracies [68]. In [70] some experiments focusing on the production and perception of real and acted emotional speech supported the opinion that acted emotional speech is not felt when spoken, and is perceived more strongly than real emotional speech. Yet Type 1 can still be quite adequate testing data as for example in [10]. It is suitable for a novel method which first requires proof of the concept, rather than construction of a real-life application for the industry. In Section 7.1 of Chapter 7 *Future Work* I describe the case where acted emotions are indispensable for creation of a real life application. Real emotions create additional and partially predicable difficulties, for example the database is typically highly imbalanced as in the case of the Aibo corpus, and a solution is resampling [57]. A less trivial difficulty is when the emotional states are fuzzy and close to one another, as in the Smartkom corpus [53]. The solution to counter some of the difficulties that authentic emotions pose have been found, yet some remain to be open problems. Type 3 is preferable, when adequate reliability on annotating emotions is difficult to achieve, particularly when annotators are asked to judge infrequent emotions like *confusion* and *surprise* rather than more typical like *anger* or *boredom.*

There are six databases used in the present study: five of them are of Type 1 (the Berlin Emotional Database and the four Interface databases of English, Slovenian, French and Spanish) and one database is of Type 2: the Aibo corpus with authentic emotions in German. Below brief descriptions of these data sets are provided.

2.3.1 Berlin Emotional Database: EMO-DB

The Berlin Emotional database (abbreviated to EMO-DB) [7] is a publicly available benchmark database of acted emotional speech. Ten German sentences of emotionally undefined content were acted in seven emotions by ten professional actors.

Number and sex of speakers: five male and five female.

Recognition classes: *anger*, *joy*, *disgust*, *fear*, *sadness*, *surprise* and *neutral*, i.e. the set of emotions from the MPEG-4 standard.

Size: 488 phrases.

Labelling: Out of initial 700 phrases, throughout perception tests by twenty human listeners 488 phrases were chosen as more than 60% natural and more than 80% clearly assignable.

Recording characteristics are 16 bit, 16 kHz, under studio noise conditions.

2.3.2 Interface databases

The Interface databases [23] contain acted emotional speech in French, English, Slovenian and Spanish.

Number and sex of speakers: two male actors per each database.

Recognition classes: *anger*, *disgust*, *fear*, *joy*, *surprise*, *sadness* and *neutral* as well as variations of *neutral*: regular neutral for French, slow-soft and fast-loud neutral styles for French, Slovenian and English databases.

Size: randomly chosen subsets of 3711, 3805, and 4030 utterances for English, Slovenian and French, respectively.

Labelling and unit of analysis: The corpora contain isolated words, short (five to eight words), medium-length (thirteen words) and long (fourteen to eighteen) sentences that are context-independent. The sentences are both interrogative and affirmative.

Recording characteristics are 16 bit, 16 kHz, under studio noise conditions.

2.3.3 FAU Aibo database

The *FAU Aibo* database [65] contains recordings of spontaneous speech of German children interacting with the Sony's pet robot Aibo, which is shown in Figure 2.1. The children were led to believe that Aibo was responding to their commands, whereas in truth the robot was controlled by a human operator. The wizard caused

Aibo to perform a predefined sequence of actions. Sometimes Aibo behaved disobediently thereby provoking emotional reactions.

Number and sex of speakers: The speech was recordered from 26 children: 13 boys and 13 girls.

Recognition classes: There are five recognition classes: *Anger*, *Emphatic*, *Neutral*, *Positive*, and *Rest*.

Size: For this study a subset of 2970 samples from the Aibo corpus, namely the TRAIN set from the EMO challenge was available.

Labelling and unit of analysis: Here the labelling and the segmentation strategies decided on by the data set holders are summarized. The recordings were segmented automatically into *turns* using a pause threshold of one second. Five labellers listened to the turns and annotated each word as neutral or belonging to one of ten other classes. Since many utterances were only short commands and rather long pauses can occur between words due to Aibo's reaction time, the emotional state of the child can also change within turns. Hence, the data was validated on the word level. The words were labelled according to majority voting: if three or more labellers agreed, the label was attributed to the word. The results of the labelling were the following: 101 words were labelled as *joyful*, 0 as *surprised*, 2528 as *emphatic*, 3 as *helpless*, 225 as *touchy* (irritated), 84 as *angry*, 1260 as *motherese*, 11 as *bored*, 310 as *reprimanding*, 3 is *rest* (non-neutral but not belonging to other categories), 39169 as neutral and other 4707 did not received any majority vote. In total there are 48401 words. The final labels are *Anger* (subsuming *angry*, *touchy* and *reprimanding*), *Emphatic*, *Neutral*, *Positive* (subsuming *motherese* and *joyful*) and *Rest*.

The creators of the *Aibo* corpus carried out classification experiments on a subset of the *Aibo* corpus [65] (Table 7.22, p. 178) and arrived to the conclusion that the best unit of analysis is neither the word nor the turn, but some intermediate chunk, which is the best compromise between the length of a unit of analysis and and the homogeneity of emotional states within one unit. Hence manually defined chunks based on syntactic-prosodic criteria [65] (Section 5.3.5) were used. A heuristic approach similar to the one applied in [65] (Section 5.3.8) was used to map the labels of the five labels on the word level for a whole chunk.

Recording characteristics: Speech was transmitted with a high quality wireless head set and recordered with a DAT-recorder. The database is 16 bit, 16 kHz.

2.4 Considerations for SER system design

In this section, the modules of the basic SER system are revisited, – feature extraction, feature selection, and classification.

2.4.1 Considerations for Feature Extraction module

Throughout this thesis the term *feature extraction* is used in the SER tradition, that is it means extracting numeric values of acoustic parameters from the speech signal, not as the construction of high-level features, as in the pattern recognition literature.

Feature vectors can be long- or short-time ones. Long-time features are estimated over the entire utterance length, while short-time features are determined in a smaller time window, usually 20 to 100 msec. The contemporary research approach favours long-time features since long-time features identify emotions better than short-time ones [31], [56]. On the other hand, the argument in favour of short-time features [43] is that global statistics can be misleading because it captures other phenomena. For example, interrogative sentences usually imply a wider pitch contour than affirmative ones, thus the pitch standard deviation in the interrogative phrase is usually larger, yet this is a reflex of the syntax of the sentence only and not the case of speaking emotionally as larger value of pitch std usually indicates.

In phonetics, acoustic parameters are traditionally divided into two classes: prosodic and segmental. To my best knowledge, there is no rigorous definition of prosody. For a working definition of prosodic and segmental I take the one from affective speech synthesis research, where prosody is defined as pitch, intensity and duration, and segmental parameters as spectrum and voice quality. The voice quality features extracted in SER systems are: formant means and band widths, harmonic to noise ratio, MFCC coefficients, and spectrum connected features for example FFT, spectral roll-of-point and flux. In the literature prosodic features are invariably included in the feature vector, while segmental features are often left out. For SER it makes sense to extract both types of features, since it has been shown that some vocally transmitted emotions are better described and recognized via prosodic parameters, while others are better identified with segmental parameters. Languages differ in the way how they encode speech emotions. For example, in Spanish *sadness* and *surprise* are expressed by means of prosodic features, while *happiness* and *cold anger* are communicated by segmental features [38]. The authors of [2] conducted experiments through recognition and synthesis modifying pitch with a PSOLA-like algorithm. They confirmed that in Spanish, *surprise* is a prosodic emotion, *anger* is segmental, while *sadness* and *happiness* are a mix. On the contrary, in Japanese *anger*, *surprise* and *sorrow* are prosodic, while *joy* and *fear* are segmental [40].

There is plenty of active research in the search for the golden set of features. This includes work in defining new signal features [36], [41]. As well as *constructing* new features from the known low-level features. Such constructed features can be used alone or be early- or late-fused with the initial feature set [49]. As for constructed features, evolutionary programming techniques [27] were applied [55] and I contributed to this line offering the distance-to-automaton features, which are described in Chapter 5. Also non-signal information can be included in the feature vector, for example adding information about dialogue structure [1], [32], [9], speaker personality and tracking lexical keys [54] were proposed.

2.4.2 Considerations for Feature Selection

The feature selection procedure gives a number of simultaneous improvements:

- it eliminates irrelevant features that hinder the recognition rates;
- it lowers the input dimensionality and therefore improves generalization;
- it saves computational time.

There have been examples of successful SER experiments that do not have an automatic stage of feature selection, especially in the early literature. Sometimes people extract the features that were chosen in other studies or they chose a classifier, which is robust against noisy features, or had enough knowledge about the domain to guess the right set.

As far as the automatic feature selection is concerned, there are two fundamentally different approaches. One is to make an independent assessment based on general characteristics of the data and the other one is to evaluate the subset using the machine learning algorithm that will be ultimately employed for learning. The first is called the *filter* method, because the attribute set is filtered to produce the most promising subset before learning starts. The second is the *wrapper* method, because the learning algorithm is wrapped into the selection procedure.

2.4.3 Considerations for Classification Module

As for the choice of the classifier, as it is stated in the *no free lunch theorem* [72] there is no uniform *a priori* answer to the question which classifier constitutes the best choice. The criterion for selecting the classifier should be related to the task, in order to take into account the regularities of the problem, and the geometry of the input feature space. Some classifiers are more efficient with certain type of class distributions, and some are better at dealing with many irrelevant features or with structured feature sets. A straightforward strategy to choose a classifier is to test all the classifiers available on the validation set and take the top performer. In the literature many classifiers have been tried for SER. The support vector machines and the neural networks are the most popular choices. For the data in this study the support vector machines and the multilayer perceptron proved their reputation of the top performers: on the Berlin EMO-DB and on the Aibo corpus respectively (Section 5.3).

2.4.4 Multilingual SER

In [52] the hypothesis of *universality versus language variability* of the emotion effects on vocal production was studied. The idea was as follows: if emotion effects on the voice were universal, little or no adaptation would be necessary when switching between recognition in different languages, whereas culturally or linguistically relative emotional effects would require special adaptations for specific languages or countries. The results of perception experiments support a mild version of the universality hypothesis. The experiments in question deal with the languages of three families: Germanic (Dutch and English), Romance (Italian, French and Spanish) and non Indo-European languages spoken in Indonesia. According to the experiment, belonging to a language family correlates with the emotion coding too, that is being similar as languages (in grammar and basic vocabulary) implies similarity in speech emotion coding by means of pitch, energy, time-related parameters and so on. In practice this partial universality is large enough to allow for direct multilingual emotion recognition, at least on a mixture of Indo-European languages.

Another interesting result was reported in [26], where it was shown that basic emotions in English, Chinese and Japanese are separable in the coordinates of pitch expectancy and variance.

Chapter 3

ESEDA system

The first objective of this thesis as I put it in Chapter 1 is

- to build a robust system to be exposed to acted and authentic emotions in various languages.

Here I present my *ESEDA* system. *ESEDA* stands for **E**nhanced **S**peech **E**motion **D**etection and **A**nalysis. It is based on a standard supervised machine learning procedure and is enhanced with an additional module of classification error analysis and prevention. The error prevention module explores the confusion matrix obtained on the validation set and decides which of the available actions to undertake in order to avoid some types of errors in the future. Despite its theoretical simplicity the inclusion of this module leads to good improvements in accuracy and is therefore practically useful.

The structure of this chapter is as follows. Section 3.1 covers the basic part of the system's architecture and its testing. In Section 3.2 I explain the ideas underlying the error-prevention module, give an example of how it works on onde of the data sets, and finally provide its testing results. Section 3.3 is discussion and conclusions.

3.1 Basic System Architecture

Having carefully weighed all the facts from Section 2.4 [60], I made my choices with respect to the design of the *ESEDA* system. The standard part of the system is comprised of the three modules:

- feature extraction,
- feature selection, and
- classification.

Their performance serves as a baseline to judge the usefulness of the module for error prevention.

Feature Extraction

The feature extraction module extracts 116 global statistical features, both prosodic and segmental. The acoustic parameters are pitch, intensity, formants and harmonicity. A complete list of the features can be found in Appendix B *Low-level features.*

I started extracting statistical functions from the acoustic parameters that were reported to be beneficial in the SER literature. Where I was not sure whether a particular feature would be redundant or useful, I still added it. This is an affordable redundancy because:
– feature selection will filter out the irrelevant features;
– some features become relevant in combination with other features;
– it is known that many of the features proposed in the psychoacoustic literature turned out to be not-very-useful for SER. I decided to allow for a bigger diversity in the *ESEDA* for the sake of possible new applications that have to do with human impressions. I stopped having extracted a little more than 100 features because:

1. Enough features have been extracted already to allow for reasonably accurate SER.

2. Open-source powerful feature extraction software was on its way. Such systems are not solely SER-oriented and thus offer a bigger variety of features than a SER designer would think of. For the work described in Chapter 5 *Distance To Automata Features* I used the powerful feature extraction system SMILE for music [17], made available for the public use this year. Another advantage of the open source software is the resulting transparency of the system.

New open-source powerful solutions keep appearing and with time older blocks can be substituted with them. Therefore, with respect to the basic recognizer, instead of polishing individual modules, I decided that further efforts to improve robustness of the *ESEDA* should lie in devising wrapper approaches, like the module for error prevention explained in Section 3.2.

Feature Selection

The feature selection procedure implements correlation-based feature subset selection [18], which is one of the filter type. It eliminates redundant and irrelevant features extracted by the previous module by selecting a subset of attributes that individually correlate well with the class but have little intercorrelation. The correlation between two nominal attributes are measured by the *symmetric uncertainty*:

$$U(A, B) = \frac{i(A) + i(B) - i(A, B)}{i(A) + i(B)}; \tag{3.1}$$

where i is the entropy function[1] from Equation 2.45. The symmetric uncertainty lies between 0 and 1. Correlation based feature selection determines the goodness

[1]The joint entropy of A and B $i(A, B)$ is calculated from the joint probabilities.

of an attribute set through

$$\sum_j \frac{U(A_j, C)}{\sqrt{\sum_i \sum_j U(A_i A_j)}}, \tag{3.2}$$

where C is the class value and the indices i and j range over all attributes in the feature vector.

The resulting vector depends on the language[2]:

- For *French* it had nine features: mean of pitch analysis based on cross-correlation method, mean of intensity, mean of harmonicity Cc, value at 1500 Hz as a function of frequency of long-term average spectrum, max of long-term average spectrum, frequency of minimum of the power spectral density, min of pitch Cc, mean absolute value of pitch SPINET, mean absolute slope of pitch SPINET.

- For *Interface English* the feature vector had 22 features: max of pitch Cc, std of pitch Cc, column distance harmonicity Gne, mean of intensity, mean of harmonicity analysis based on auto-correlation method (abbreviated to Acc), min of long term average spectrum, max of long term average spectrum, mean of long term average spectrum, bin width of long term average spectrum, min of long term average spectrum pitch corrected, max of long term average spectrum pitch corrected, quantile of pitch Acc, std of pitch Acc, min of pitch Cc, max of pitch Cc, std of pitch Cc, jitter Dpd, min of pitch SPINET, max of pitch SPINET, quantile of pitch SPINET, max of pitch Shs, mean absolute slope of pitch Shs.

- For *Interface Slovenian* the feature vector had 13 features: column distance harmonicity Gne, max of intensity, std of intensity, mean of harmonicity Cc, bin width of long term average spectrum, value at 1500 Hz as a function of frequency of long-term average spectrum, min of long-term average spectrum, mean of long-term average spectrum, std of pitch Acc, jitter Local, mean of absolute slope of pitch SPINET, min of pitch Shs, std of pitch Shs.

- For *Interface Spanish* the feature vector had 31 features: min of pitch Cc, max of pitch Cc, std of pitch Cc, mean of intensity, min of harmonicity Acc, min of harmonicity Acc, min of long-term average spectrum, frequency of min of long-term average spectrum, max of long-term average spectrum, mean of long-term average spectrum, value in bin of long-term average spectrum pitch corrected, min of long-term average spectrum pitch corrected, max of long-term average spectrum pitch corrected, local pitch height of long-term average spectrum pitch corrected, height1 of long-term average spectrum pitch corrected, min of pitch Cc, max of pitch Acc, mean of pitch Acc, std of pitch Acc, mean absolute slope of pitch Acc, slope with octave jumps pitch Acc, quantile of pitch Cc, mean of pitch Cc, std of pitch Cc, slope with octave jumps of pitch Cc, point

[2]The abbreviations used and the algorithms behind a particular statistics are covered in Appendix B *Low-level Features*.

pitch process: jitter rap, min of pitch SPINET, slope with octave jumps pitch SPINET, linear fit of pitch SPINET, max of pitch Shs quantile of pitch Shs.

- For *EMO-DB German* the feature vector had 27 features: mean of pitch Cc, std of pitch Cc, mean of intensity, max of intensity, min of format LPC, max of formant LPC, min of harmonicity Acc, max of harmonicity Acc, value at 1500 Hz as a function of frequency of long-term average spectrum, min of long-term average spectrum, frequency of min of long-term average spectrum, max of long-term average spectrum, bin width of long-term average spectrum pitch corrected, min of long-term average spectrum pitch corrected, max of long-term average spectrum pitch corrected, mean of pitch Acc, std of pitch Acc, mean absolute slope of pitch Acc, quantile of pitch Cc, std of pitch Cc, jitter Rap pitch Point Process, jitter Dpd, std of pitch SPINET, slope with octave jumps pitch SPINET, max of pitch Shs, quantile of pitch Shs, mean absolute slope of pitch Shs.

- I did not use the feature extraction module from the *ESEDA* for the work with *Aibo German*. Instead I used the powerful feature extraction system *SMILE* for music from [17][3].

- For merged *Interface* data sets, the feature vector had 21 features: maximum of pitch CC, std of pitch Cc, column distance of harmonicity Gne, mean of intensity, mean of harmonicity Cc, max of harmonicity Acc, min of long-term average spectrum, frequency of min of long-term average spectrum, max of long-term average spectrum, bin width of long-term average spectrum pitch corrected, max of long-term average spectrum pitch corrected, min of pitch Acc, mean of pitch Acc, std of pitch Acc, slope with octave jumps of pitch Cc, jitter Dpd, min of pitch SPINET, mean absolute slope of pitch SPINET, quantile of pitch Shs, std of pitch Shs.

Classification

The classification module takes as input a feature vector created by the feature selection module, and applies the multilayer perceptron classifier [71] or the support vector machine [67], in order to assign a class label to it. As for multilingual classification, I merged the *Interface* corpora, did feature selection, trained and tested the classifier as if the data were from a monolingual data set.

[3]My reasons were as follows:

1. I wanted to submit the TA features for an open competition of high-level features at the INTERSPEECH challenge, and the requirement was that the high-level features should be constructed from the low-level features provided by the organizers;

2. *SMILE* is a powerful feature extraction system aimed for various signal processing applications and has just become publicly available. I took the earliest opportunity to benefit from it in my research.

	English	Slovenian	French	Spanish
Neutral	95%	92%	76%	93%
Anger	75%	86%	70%	66%
Joy	62%	48%	83%	74%
Fear	78%	72%	53%	81%
Surprise	76%	51%	63%	79%
Disgust	49%	61%	94%	70%
Sadness	76%	78%	72%	79%
Averaged	73%	71%	81%	80%

Table 3.1: The accuracy on English, Slovenian, French and Spanish.

	Neu	An	Dis	Fe	Jo	Su	Sa	%
Neu	**297**	61	18	1	1	9	2	76
An	41	**218**	16	0	0	38	0	70
Dis	22	16	**660**	0	0	6	1	94
Fe	0	0	0	**372**	70	112	146	53
Jo	0	0	0	66	**577**	30	25	83
Su	15	17	3	83	44	**330**	33	63
Sa	0	0	0	140	33	26	**501**	72

Table 3.2: Basic Recognition for French.

3.1.1 Testing for the basic recognizer

For the testing protocol, 10-fold cross-validation was used. The recognition accuracy for the *Interface* data sets are listed in Table 3.1. The confusion matrices for monolingual validation are presented in Tables 3.2, 3.3, 3.4, and 3.5 for French, English, Slovenian and Spanish respectively.

As follows from the matrix for **the French language** in Table 3.2, the obtained accuracy is 73%. Accuracies for the individual classes are: 76% for *neutral*, 70% for *angry*, 94% for *disgusted*, 53% for *fear*, 83% for *joy*, 63% for *surprise*, and 72% for *sad*. On average the accuracy are good, with the exception of *fear*, which is often confused with *surprise* and *sad*, and *surprise*, which is often confused with *fear*. Symmetry[4] in the error pattern is not infrequent and will appear further.

As follows from the matrix for **the English language** in Table 3.3, the average accuracy is 81%. The accuracy for individual classes is: 95% for *neutral*, 75% for *angry*, 49% for *disgusted*, 78% for *fear*, 62% for *joy*, 76% for *surprise*, and 76% for *sad*. The least well recognized classes are *joy* (it is often confused with *neutral* and *surprise*) and *disgust* (it is often confused with *neutral* and *anger*).

As follows from the matrix for **the Slovenian language** in Table 3.4, the obtained accuracy is 71%. We have 92% for *neutral*, 81% for *angry*, 74% for *fear*, 62% for *joy*, and 57% for *surprise*. The least well recognized classes are *joy* (it is often confused with *surprise* and *disgust*) and *surprise* (it is often confused with *disgust*).

[4]I use the underlined font to highlight symmetry.

	Neu	An	Jo	Fe	Su	Di	Sa	%
Ne	**1752**	7	33	8	4	46	2	95
An	33	**279**	19	1	12	27	1	75
Jo	61	20	**231**	13	32	13	1	62
Fe	13	0	14	**289**	1	10	45	78
Su	2	6	27	0	**141**	9	1	76
Dis	117	30	20	5	16	**181**	3	49
Sa	9	0	1	34	0	1	**141**	76

Table 3.3: Basic Recognition for English.

	Neu	An	Jo	Fe	Su	Di	Sa	%
Ne	**1058**	90	0	0	1	0	0	93
An	105	**639**	0	0	1	0		66
Dis	0	1	**465**	8	84	47	153	70
Fe	0	1	4	**270**	42	51	7	81
Joy	0	0	96	41	**371**	232	35	74
Sur	0	0	31	67	227	**388**	47	79
Sa	0	1	92	4	36	30	**594**	79

Table 3.4: The confusion matrix for Slovenian.

	Neu	An	Jo	Fe	Su	Di	Sa	%
Ne	**1534**	34	53	9	13	0	15	93
An	36	**512**	40	14	70	49	1	71
Dis	84	54	**508**	38	17	12	18	70
Fe	18	13	31	**591**	12	38	31	81
Joy	19	73	12	16	**538**	73	0	74
Sur	9	51	10	37	44	**576**	1	79
Sa	38	1	18	25	0	0	**311**	79

Table 3.5: Basic Recognition for Spanish.

	Neu	An	Jo	Fe	Su	Di	Sa	%
Ne	**5286**	348	166	79	110	76	99	86
An	496	**1688**	73	61	178	197	18	62
Dis	460	106	**1066**	88	93	84	97	53
Fe	113	8	58	**1756**	159	215	244	69
Joy	275	232	92	183	**1688**	449	28	57
Sur	102	108	30	186	402	**1476**	41	63
Sa	203	17	80	263	38	51	**999**	61

Table 3.6: The confusion matrix for multilingual SER.

class	precision	recall	F-measure
neutral	0.76	0.86	0.81
anger	0.67	0.62	0.65
disgust	0.68	0.53	0.60
fear	0.67	0.69	0.68
joy	0.63	0.57	0.60
surprise	0.58	0.63	0.60
sad	0.66	0.61	0.63

Table 3.7: Precision, Recall, and F-measure for multilingual emotion recognition.

class	precision	recall	F-measure
fear	0.82	0.74	0.77
disgust	0.72	0.74	0.73
happiness	0.52	0.49	0.51
boredom	0.73	0.75	0.74
neutral	0.71	0.78	0.75
sadness	0.88	0.94	0.91
anger	0.75	0.76	0.75

Table 3.8: Recognition with the MLP on the EMO-DB.

As follows from Table 3.8, the accuracy for acted German (*Berlin EMO-DB*) averaged over all classes is 74%.

The question of authenticity of emotions has been a hot topic in the literature for the last four years. I had to prove that the *ESEDA* can classify authentic speech emotions as well as try out my ideas beyond the lab conditions. Table 3.10, reports the testing results on authentic German (the *Aibo* corpus). As follows from the confusion matrix, the accuracy averaged over all classes is 61%.

Table 3.6 provides the confusion matrix for multilingual SER on a mixed dataset of English, Slovenian, French and Spanish. The averaged accuracy is 69.5%. Precision, recall and the F-measure are provided in Table 3.7. The least well recognized classes are *joy* (is mostly confused with *surprise*), *surprise* (is mostly confused with *joy*) and *sad* (is mostly confused with *fear*). Again, there is symmetry (highlighted with the underlined font) is present in the error pattern: *joy* and *surprise* are con-

class	precision	recall	F-measure
emphatic	0.55	0.31	0.40
anger	0.65	0.15	0.24
neutral	0.62	0.95	0.75
rest	0	0	0
positive	1	0.01	0.02

Table 3.9: Precision, Recall and F-measure for the Aibo corpus.

	Emphatic	Anger	Neutral	Rest	Positive	%
Emphatic	**189**	13	411	0	0	31
Anger	57	**35**	148	0	0	15
Neutral	74	3	**1597**	0	0	95
Rest	18	1	202	**0**	0	0
Positive	3	2	215	0	**2**	1

Table 3.10: The confusion matrix for the Aibo corpus.

fused with one another. One more time, a suitable treatment would be to treat *joy* and *surprise* as one class first, recognize everything else and then recognize between *joy* and *surprise* in a more appropriate feature space defined by a separate feature selection step made for this binary classification task. Summing up the observed properties of the error matrices, I can conclude that:

1. Errors often imply a symmetrical pattern: if class X is typically taken for class Z, then it often is the case that the reverse is also true: Z is taken for X. This makes classification decomposition a naturally fitting solution.

2. *Neutral* seems always to be the best captured emotion, while *disgust* is least well recognized with the exception of one language. I think this caused by habit: the *neutral* affect is the most frequent, while *disgust* is extremely infrequent in real life. People can hardy recall being disgusted and therefore can not play this emotion in a sufficiently convincing way.

3.2 Module of Error Analysis and Prevention

3.2.1 Description of the additional block

After having constructed the basic speech emotion recogniser I analysed the accuracy and realised that there is room for improvement. Certainly the desired improvement can be obtained in more than one way. For example, a direct solution is the revision of every module of the basic recognizer: feature extraction, feature selection or classification. That for example could be extracting more signal features, or trying all the available feature selection algorithms on the validation set and choosing the one that leads to the best accuracy, and so forth. The drawback is that it can turn out to be database-dependent, since we can not be sure which are the best features or the best classifier *a priori* before having seen the type of speech data: for example speech of a German child interacting with a toy is unlikely to show the same trends as Spanish data from a call centre. A data-independent solution would be of more use: a *wrapper* machine learning technique without reprogramming the basic recognizer. To this end, from the rich palette of machine learning methods, I take classification decomposition.

A decomposition splits a complete multiclass problem into a set of smaller classification problems. Decompositions allow for learning more accurate concepts due to simpler classification boundaries in subtasks and the feature selection procedure

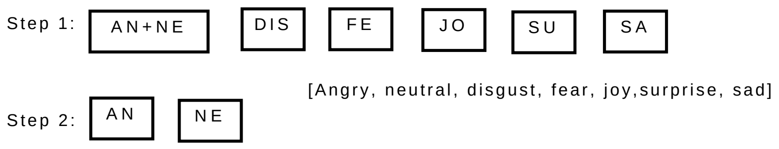


Figure 3.1: The scheme for the classification decomposition.

done individually for each classification step [12]. When doing classification decomposition, the central choice is the order of combination of smaller classification steps, called the classification path. My idea is to derive the classification path from the confusion matrix and, uncovering the reasons for errors, design a module that prevents the system from making such errors in the future:

1. The class of special interest is identified (denote it class I), for which the recognition rates are to be improved. For example it can be the worst recognised class or a class of special interest for some application. From the confusion matrix of the standard classification step it is deduced with which other class the class of interest is most frequently confused (denote it class J). The original classification step is then divided into two new steps: the first multiclassification step, where the new class labels are the old ones, except that there is a joint label K for classes I and J, and the second binary classification step, where instances of class K are classified into I or J.

   Analogously for to the case of one class of special interest I, for the case when there are m ($m \geq 2$) classes of special interest (or classes recognized with the accuracy below the desired threshold): $\{I_1, ..., I_m\}$. From the confusion matrix of the standard classification step, it is deduced with with other class $\{J_1, ..., J_m\}$ $\{I_1, ..., I_m\}$ are pairwisely confused: $\{(I_1, J_1), ..., (I_m, J_m)\}$. Tires are broken randomly. The original classification step is then divided into $(m + 1)$ new steps: the first misclassification step, where the new labels are the old ones, except that there is a joint labels $K_1, ..., K_m$ for K pairs of (I_p, J_p) for $p = 1, ..., m$. And k second classification steps where instances of classes K_p are classified into I_p or J_p.

2. If the minority class problem is present and hampers the classification accuracy, cost-sensitive training is employed, more specifically, every minority class sample in the database is duplicated.

The first step of this procedure is an ordered decomposition approach to a multiclass classification problem. Usually every sub-classification problem in decomposition has a simpler classification boundary and feature selection is performed separately for individual classification steps. The classification path of this experiment is shown in Figure 3.1.

The minority class detection operates as follows:

- If some class is not well recognised (that is $\leq 70\%$), check if this is a minority class (that is ≤ 500 samples in the training set);

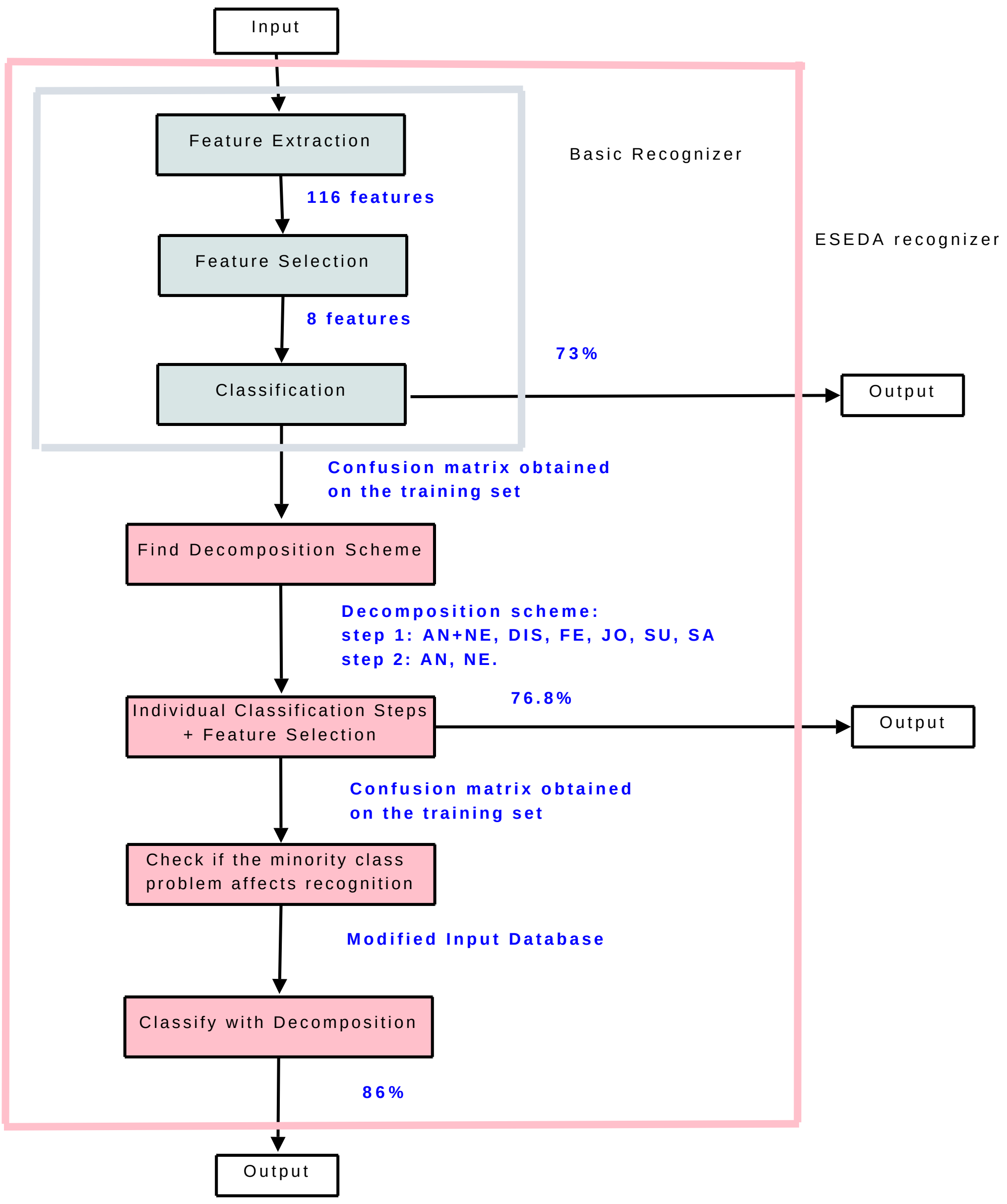

Input
Feature Extraction
116 features
Feature Selection
8 features
Classification
Basic Recognizer
ESEDA recognizer
73%
Output
Confusion matrix obtained on the training set
Find Decomposition Scheme
Decomposition scheme:
step 1: AN+NE, DIS, FE, JO, SU, SA
step 2: AN, NE.
Individual Classification Steps + Feature Selection
76.8%
Output
Confusion matrix obtained on the training set
Check if the minority class problem affects recognition
Modified Input Database
Classify with Decomposition
86%
Output

Figure 3.2: The flowchart for the *ESEDA*.

- If this is a minority class, duplicate each sample of this class in the training set.

3.2.2 Example

A flow chart of the *ESEDA* recognizer is given in Figure 3.2. To try the proposed block of error-prevention, I took the French data set. The additional block proceeds as follows:

1. The confusion matrix is obtained, see Table 3.2;

2. *Anger* was taken as a class of special interest as required in a number of applications. For example, in call centres anger detection is needed for the off-line control of how well conflict dialogues are resolved, etc. Alternatively, *fear* could have been taken as least well recognized[5].

3. From the confusion matrix obtained it was deduced that *anger* is mostly confused with *neutral.* Therefore the classification was done in two steps: among the new classes (the new labels are the old labels, except that there is a joint label for *anger* and *neutral*), and then an extra classification step is added to classify between *anger* and *neutral.* The classification path is depicted in Figure 3.1.

4. Table 3.11 gives a step-wise explanation of how the detection of minority class problem worked on the data, where the possible "paths" of the minority class problem are coloured in red. As long as the logical conditions are met, the font is red. The black font signifies that the data in the column was checked, some logical condition had not been met, and therefore the detection of the minority class problem stopped. For example, in Table 3.11 take the row starting with "Fear": 53% is consistent with the condition *The accuracy for a given class is less than 50%*, so the minority class problem detection can continue. Yet the second condition is not met: the number of samples is 700, which is greater than 500. The detection of the minority class problem stops and the treatment of duplicating every sample is not applied for the *fear* row.

   The minority class problem was detected for *angry*, therefore its every sample was duplicated in the training set.

3.2.3 Testing for the module of error prevention

Table 3.12 shows the consecutive improvements due to the actions based on error-analysis. The improvement of 3,5% after the classification decomposition is expected: decompositions allow for learning more accurate concepts, because usually smaller problems have simpler classification boundaries and feature selection was performed separately for the two classification steps.

[5]At the time I was asked to give a presentation on anger detection for a commercial application in *Telefonica ID*, so I thought *anger* would make the example more interesting.

Emotion	Accuracy	Number of samples	MCP treatment?
Neutral	76%	389 samples	No
Anger	**70%**	**313 samples**	**Yes**
Disgust	94%	705 samples	No
Fear	**53%**	700 samples	No
Joy	83%	689 samples	No
Surprise	63%	525 samples	No
Sad	72%	700 samples	No

Table 3.11: The Minority Class Problem treatment.

	baseline	+ extra classification step	+ cost-sensit. training
An	70%	84%	99.5%
Ne	76%	95%	93%
All classes	73.3%	76.8%	86%

Table 3.12: The improvements due to error-prevention.

The cost-sensitive training brought 9.2% more. As the recognition rates improve, the false alarm rate increases only by 2%, that is the accuracy for the neutral class drops from 95% to 93%. In [57] resampling is done as the part of data cleansing before training without checking any conditions. It can be done in either way: as deterministic data cleansing or in a more economical way, as do it, only after having checked whether a given operation is optimal. In both cases, balancing improved the accuracy.

3.3 Conclusions

The *ESEDA* [59], [62] is based on the supervised pattern recognition cycle. The classical part of the system is comprised of the three modules: feature extraction, feature selection, and classification. Its performance served as a baseline to validate the new theory. The *ESEDA* was tested on acted and real emotions, and on five languages. The testing results allow me to conclude that the system is ready to be integrated into real-life applications.

To enhance the classical design, the *ESEDA* has an exclusive block of error prevention. The underlying idea is to analyse the confusion matrix on the validation set and design a module that prevents the system from making these errors on new material. Despite its simplicity, the module led to notable improvements in accuracy.

Chapter 4

TGI+ classifier

The second contribution of this thesis is a new algorithm of a mixed design with syntactic [6] and statistical learning [25], the general idea for which comes from optical character recognition [58]. The syntactic part implements tree grammar inference, and the statistical part implements C4.5 [48][1]. First I model the objects by means of a syntactic method, that is the samples are mapped into their representations. A representation of a sample is a set of seven numeric values, signifying to which degree it resembles the averaged pattern of each of the seven classes. Then, the mappings of samples are classified, not their feature vectors, with a statistical method. I called the classifier *TGI+*, which stands for Tree Grammar Inference[2] and the plus is for the statistical learning enhancement. I evaluated the *TGI+* against a state of the art classifier. To choose the competitor for the *TGI+*, I ran all the *weka* classifiers [71] on a benchmark data set. The multilayer perceptron turned out to be the top performer. Experimental results showed that the *TGI+* outperforms the multilayer perceptron by a statistically significant difference in accuracies of 4.68%, which allows us to conclude that the *TGI+* is a legitimate classification method.

In this chapter, I explain the *TGI+* classifier [61]. In Section 4.1 I refresh the idea of a combined classification strategy from optical character recognition, which was my starting point. In Section 4.2 *TGI+ algorithm* I informally introduce and formally define the proposed classification method. In Section 4.3 I explain my further extension of the *TGI+*. In Section 4.4, I report the experimental results, which I then discuss in Section 4.6 and on the basis of which in Section 4.7 I conclude that the *TGI+* is a classifier with a rightful place among other classifiers that are recommended for SER.

4.1 Departure point

The general idea comes from optical character recognition and was explained in greater detail in Section 2.2.2 of Chapter 2. The highlights of the method proposed for optical recognition of handwritten digits were:

1. obtain a tree representation of every sample in the data set;

[1]The C4.5 algorithm is also explained in Section 2.2.3 C4.5 of Chapter 2 *State of the Art*

[2]Grammar inference is a process of learning a grammar (in the sense of a term-rewriting system) from examples.

2. divide the data set into two disjoint subsets: Set 1 and Set 2. From Set 1 with an error-correcting inference algorithm learn a tree automaton for each recognition class;
3. calculate edit distance between every digit and every tree automaton;
4. from Set 1 and Set 2 perform the learning and testing phase for decision trees.

The original implementation of this idea [58] had to be modified in order to suit the new application: emotional speech in place of hand written digits.

4.2 *TGI+* algorithm

In this section I informally introduce and formally define the proposed classifier.

4.2.1 Informal description of the algorithm

The *TGI+.1* is comprised of the four major steps, which I summarize below and explain in greater detail in further sections. Figure 4.1 graphically depicts the procedure.

Step 1: In order to perform tree grammar inference, the samples are represented with tree structures. Each utterance from the data set is represented by a tree graph, whose skeletons are described by the grammar below. The organization of the nodes in the tree reflects a traditional view on speech, as for example by the phonetician, expressed as a set of rules. S denotes a start symbol of the formal grammar in the sense of a term-rewriting system:

{$S \longrightarrow$ ProsodicFeatures SegmentalFeatures;

ProsodicFeatures $\longrightarrow$ Pitch Intensity Energy;

SegmentalFeatures $\longrightarrow$ Jitter Shimmer Formants Harmonicity;

Pitch $\longrightarrow$ Min Max Quantile Mean Std MeanAbsoluteSlope;

etc.
}

The *etc.* stands for further terminating productions, that is the productions, which have low-level features on their right hand side. All trees have 116 leaves, each corresponding to one of the 116 features from the sample feature vector. The trees of the same class are put into one set. In the dataset there are the following seven recognition classes: *fear*, *disgust*, *happiness*, *boredom*, *neutral*, *sadness* and *anger*.

Step 2: Apply tree grammar inference to learn seven automata each accepting a different type of emotional utterance. Divide the training set into two subsets: T_1 is 39% of training data and T_2 is the rest of training data. The splitting point at 39%

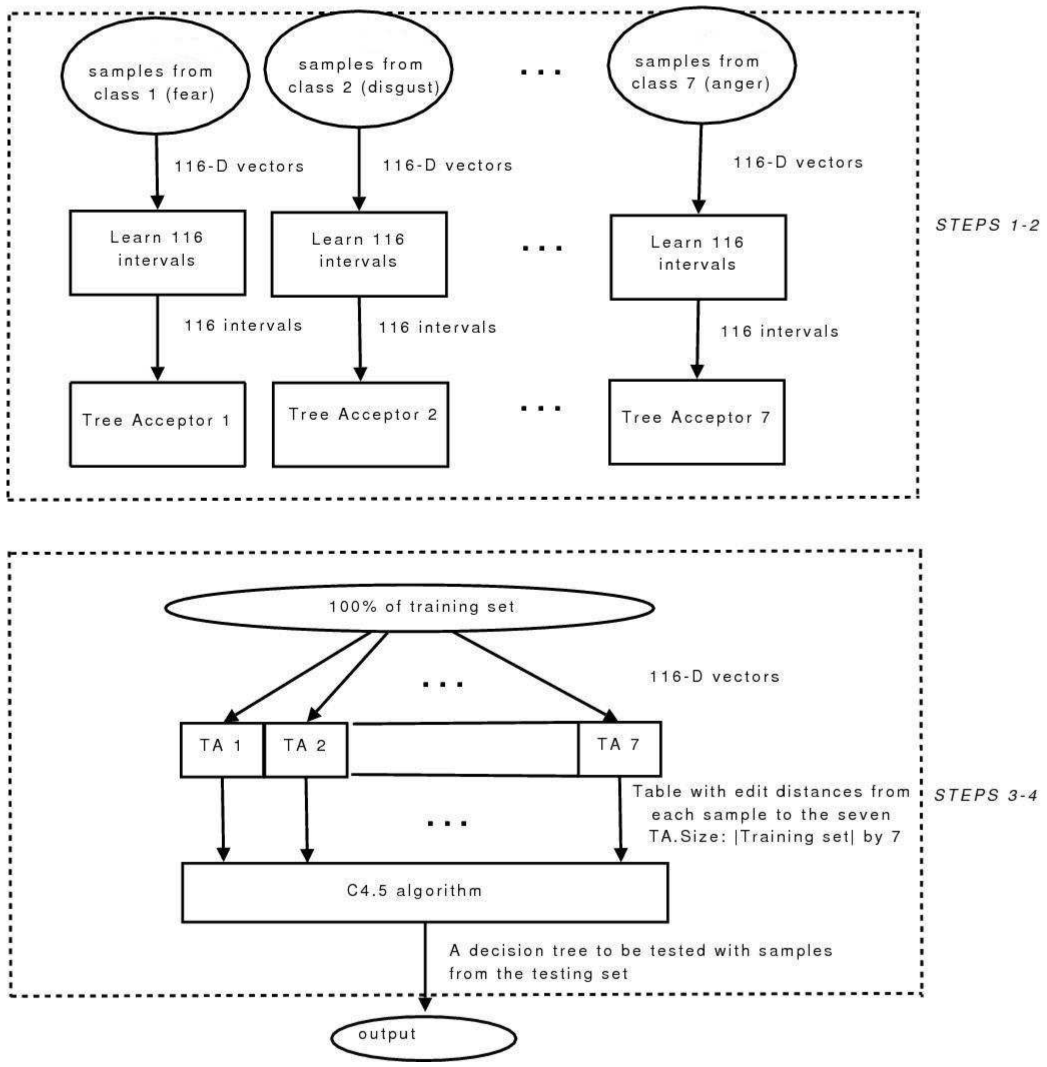

samples from class 1 (fear)
samples from class 2 (disgust)
samples from class 7 (anger)
116-D vectors
Learn 116 intervals
116 intervals
Tree Acceptor 1
Tree Acceptor 2
Tree Acceptor 7
STEPS 1-2
100% of training set
116-D vectors
TA 1
TA 2
TA 7
Table with edit distances from each sample to the seven TA.Size: |Training set| by 7
STEPS 3-4
C4.5 algorithm
A decision tree to be tested with samples from the testing set
output

Figure 4.1: The *TGI+* steps.

was chosen experimentally. The result of this step is seven automata, one for each of the seven emotions to be recognised.

Step 3: Calculate edit distances between obtained tree automata and trees in the training set. Edit distances are calculated between each automaton obtained at step two and each tree representing utterances from the training set ($T_1 \cup T_2$). The calculated edit distances are put into a matrix of size: (number of samples in the training set) $\times$ (number of recognition classes). The table has a characteristic look: the first 39% of the rows are guaranteed to have 0 for a distance to some of the tree automata, while generally this is not true for the rest of the rows. The explanation for this is that the first 39% of the training set were used to train the automata and, clearly, the distance from a sample used in the inference procedure to the automaton, the grammar of which was being learnt, is 0.

Step 4: Run C4.5 over the matrix to create a decision tree. The C4.5 algorithm is run over this matrix in order to create a decision tree, classifying each utterance into one of the seven emotions according to edit distances between a given utterance and the seven tree automata. The accuracies obtained from testing this decision tree are the accuracies of the *TGI+.1.*

A new input sample is fed to the automata in the form of a 116 feature vector. First the *TGI+* calculates distances from a sample to the seven tree automata (the automata learnt 116 feature intervals at the inference step). Then the *TGI+* uses the decision tree to classify the sample (the decision tree was learnt with the C4.5 from the table, where distances to seven automata to all the training samples had been put).

As compared to OCR algorithms, Steps 2 and 3 will require adaptations to suit a new application and in the section below I explain my solution.

4.2.2 Formal definition of the algorithm

For the reader's convenience, I repeat some of the definitions for tree grammar inference, which have been already introduced in Section 2.2.2 *Combined Classification Strategy from OCR* of Chapter 2 with minor alterations to serve our cause.

Assume we are given an alphabet V, and a set of natural numbers N. *A ranked alphabet* is defined as the association of V with a finite relation r in $(V \times N)$:

$$V_n := \{\sigma \in V \mid (\sigma, n) \in r\}. \tag{4.1}$$

V^T is the set of finite trees whose nodes are labelled with symbols in V.

A *tree* is inductively defined as follows:

$$V_0 \subseteq V^T; \tag{4.2}$$

$$\sigma(t_1, \ldots, t_n) \in V^T, \tag{4.3}$$

for $\forall t_1, \ldots, t_n \in V^T$, and $\sigma \in V_n$.

Given a tree t, the set of *subtrees* of t is denoted as $Sub(t)$:

$$Sub(a) = \emptyset, \tag{4.4}$$

for $\forall a \in V_0$;

$$Sub(\sigma(t_1, \ldots, t_n)) = \{t_1, \ldots, t_n\} \bigcup_{i=1..n} Sub(t_i), \tag{4.5}$$

for $\forall$ t_1, ... , $t_n \subseteq V^T$, and $\sigma \in V_n$.

The *root* of the tree, $root(t)$, is defined as:

$$root(a) = a, \tag{4.6}$$

for $\forall a \in V_0$;

$$root(\sigma(t_1, ..., t_n)) = \sigma, \tag{4.7}$$

for $\forall t_1, ..., t_n \in V^T$, and $\sigma \in V_n$.

The *depth* of tree, $depth(t)$, is defined as:

$$depth(a) = 0; \tag{4.8}$$

$$depth(\sigma(t_1, ..., t_n)) = 1 + max\{depth(t_i)\} \tag{4.9}$$

for $i = 1, ..., n$, and $\forall t_i \in V^T$, where $1 \leq i \leq n$, and $\sigma \in V_n$.

Subtrees for which $depth(t) \geq 1$ are called *successors* of t and denoted H^t:

$$H^\sigma(t_1, ..., t_n) = \langle root(t_1), ..., root(t_n) \rangle, \tag{4.10}$$

for $\forall t_1, ..., t_n \in V^t$, and $\sigma \in V_n$.

Let the *size of the tree* t, denoted by $|t|$, be defined inductively as follows

$$|a| = 1, \tag{4.11}$$

for $\forall a \in V_0$.

$$|\sigma(t_1, ..., t_n)| = 1 + \sum_{i=1...n} |t_i|, \tag{4.12}$$

for $\forall t_n \in V^T$, and $\sigma \subseteq V_n$.

Given a tree t, let $leaves(t)$ be the set defined as follows:

$$leaves(a) = \{a\}, \tag{4.13}$$

for $\forall a \in V_0$;

$$leaves(\sigma(t_1, \ldots, t_n)) = \bigcup_{i=1..n} leaves(t_i), \tag{4.14}$$

for $\forall t_1, \ldots, t_n \in V^T$, and $\sigma \in V_n$.

Σ denotes the set of $a \in V_0$ and is called the *leaf alphabet.* Instead of a finite alphabet of symbols, we allow a to take real values in R: $V_0 \subseteq R$.

Let the *deterministic tree automaton* (abbreviated to TA) be defined as a tuple

$$A = (Q, V, \delta, F), \tag{4.15}$$

where:
Q is finite set of states;
V is a ranked alphabet with $Q \cap V = \emptyset$;
$F \subseteq Q$ is a set of final sates;
$\delta = (0, \dots, m)$ is a finite set of transitions defined as follows:
δ_n: $(V_n \times (Q \cup V_0)^n \to Q)$, where $n = 1, \dots, m$.
$\delta_0(a) = a$, for $\forall a \in V_0$.

The relation δ can be extended to operate on trees:

$$\delta : V^T \to Q \cup V_0, \tag{4.16}$$

$$\delta : (\sigma(t_1, \dots, t_n)) = \delta_n(\sigma(\delta(t_1), \dots, \delta(t_n)), \tag{4.17}$$

if $n \geq 1$,

$$\delta_0(a) = a. \tag{4.18}$$

A given tree $t \in V^T$ is accepted by A, if $\delta(t) \in F$.
Given the state $q \in Q$, we define the *ancestors* of the state q:

$$Ant(q) = \{\langle p_1, ..., p_n \rangle | p_i \in (Q \cup V_0) \wedge \delta_n(\sigma, p_1, ..., p_n) = q\}. \tag{4.19}$$

Let σ_i^t be the i^{th} node of tree t in post order enumeration, i.e. on the same level nodes are ordered from left to right and the levels are passed in the bottom-up manner. H_i^t is the string formed by the successors of σ_i in t.

4.2.3 Tree Grammar Inference Algorithm

```
Input: A set of samples $\zeta = \{t_1, t_2, ..., t_n\}$.

  Output: $A = (Q, V, \delta, F)$, which accepts at least the sample set.

  Method: [Learn algebra from the first tree]
  $Q = Sub(t_1) \cup \{t_1\}$

  $V = leaves(t_1) \cup \{\sigma\}$

  $F = \{t_1\}$

  If $[u_1, u_2, ..., u_p \in Q] \wedge [\sigma(u_1, u_2, ..., u_p) \in Q]$

  then $\{\delta \supseteq (\sigma, u_1, u_2, ..., u_p) = \sigma(u_1, u_2, ..., u_p)$

  }
  [ From $\zeta$, learn intervals for leaves]

  [ Loop over leaves in post-order enumeration ]
```

```
    ∀$i$: 1, ..., 116 {
    $getI(i)$;

    $\delta(a_i) \supseteq ([I_{Li}, I_{Ri}], q_0)$

    }

    Method getI(i)

    Input: $\zeta = \{t_1, ..., t_n\}$

    $i$: number of the leaf in post-order enumeration

    Output: $[I_{Li}, I_{Ri}]$

    Method:

    $\forall t_j \in \zeta\{$

    $V_{0i} \supseteq a_i$;

    }

    $\forall i\{$

    $I_{L,i} = argmin\{$ elements of $V_{0i}\}$;

    $I_{R,i} = argmax\{$ elements of $V_{0i}\}$;

    Return $[I_{Li}, I_{Ri}]$
}
```

4.2.4 Edit Distance Calculation Algorithm

The original algorithm for OCR has a string distance calculation [69] plugged into the algorithm of calculation of the distance between the automaton and the tree. This algorithm uses the resemblance of substrings to measure the resemblance of resulting strings and allows "shifts", see Figure 4.2 (A.). In the case of the *TGI+*, the grammar from Step 1 outputs a vector-representation, where the position of a symbol is important and comparisons like in Figure 4.2 (B.) do not make sense. Instead of D_C, the distance on strings, I define the distance on vectors D_V. Let x denote value of some feature X, for the interval $[I_L, I_R]$, with $I_L \neq I_R$, D_V is the distance between the interval and the value x:

$$D_V = 0, \tag{4.20}$$

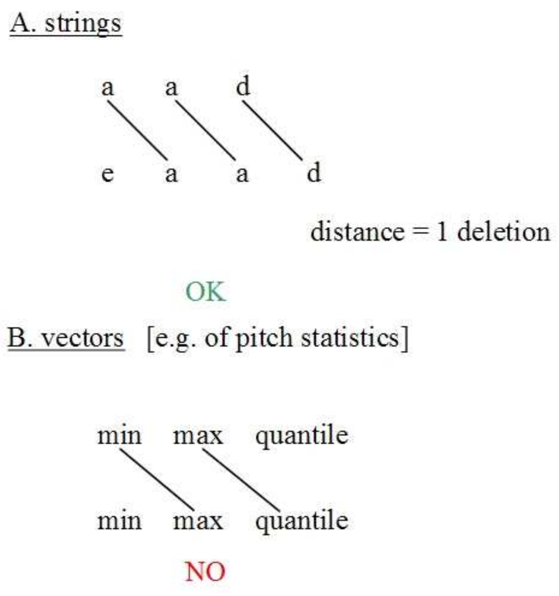


Figure 4.2: String comparison on strings (A) and vectors (B).

in case $x \in [I_L, I_R]$.

$$D_V = \frac{I_L - x}{I_R - I_L}, \tag{4.21}$$

in case $x \leq I_L$.

$$D_V = \frac{x - I_R}{I_R - I_L}, \tag{4.22}$$

in case $x \geq I_R$. We set $D_V = 0$, in case $I_L = I_R$.

The algorithm proceeds as follows:

1. For every leaf it calculates how far it is from the numeric interval, which the automaton has for this feature. The distance is calculated through Equation 4.20 – 4.22, for $\forall a$; $V_0 \subseteq V^T$. Denote by i be the position of the leaf node in post order enumeration.

$$D(a, q_{0i}) = D_V(a, I_i) \tag{4.23}$$

2. The cost for every upper node is the sum of costs of the node's ancestors. For $\forall \sigma(t_1, ..., t_n) \in V^T$, $\forall t_1, ..., t_n \in V^T, \sigma \in V_n$.

$$D(\sigma, A) = D_V(t_1) + ... + D_V(t_n) \tag{4.24}$$

3. The cost of the tree is the cost to accept the tree's root node.

$$D(t, A) = D_A[\sigma_{root}, q_f] \tag{4.25}$$

In other words, the costs are calculated in the bottom-up manner and send them up the tree.

```
Input: A finite tree automaton $A$. A tree $t$.

Output: Distance from the tree $t$ to the automaton $A$.

Method: [explore the tree in post-oder]

[as opposed to the solution for OCR,
here there is no table to do dynamic programming,
since there is one state for one node]

$\forall \sigma_i \in t\{$

$D_A[\sigma_i, q] = \sum_{\forall x \in Ant(q)} D_V(S_i^t, x)$
}

$D(t, A) = D_A[\sigma_{root}, q_f]$
Return $D(t, A)$
End Method.
```

4.3 Extension of the general scheme: TGI+.2

My further extension of the four steps has to do with Step 3. In the *TGI+*.1 all edit costs are equated to 1. In other words, if a feature value fits the interval a tree automaton has learned for it, the acceptance cost of the sample is not altered. Whenever a feature value is outside the interval the automaton has learnt for it, the acceptance cost of the sample processed is equally augmented regardless of which feature is being processed. In the *TGI+*.2 some edit costs have a coefficient greater than 1 (namely the constant $k = 1.5$ was chosen via experimentation). In other words, more important features are penalised with higher costs for being outside their interval. The set of these more important features is determined exclusively for every class (*anger*, *neutral*, etc.) through a feature selection procedure. The feature selection procedure implements a correlation based feature selection (the algorithm for which was explained in Section 2.4.2 *Feature Selection* of Chapter 2)[3].

Let w_s and w_n denote the weight for the selected and non-selected features respectively. When choosing values for (w_s, w_n), the solution lies somewhere between the two extremes

- either treat selected and non-selected features equally,
- or put a lot more of weight on selected features.

I recast the problem of choosing the best pair (w_n, w_s) into the problem of finding the value of the parameter k:

$$k = \frac{w_n}{w_s}$$

[3]Thus, the feature selection algorithm is the same for the *TGI+* and its competitor, the multi-layer perceptron.

w_n	w_s	validation accuracy
0	1	47.00%
0	2,3,...	47.19%
1	1	79.21%
1	1.5	81.09%
1	2	78.65%
1	2.5	78.84%
1	3	80.52%

Table 4.1: Different values of $k = \frac{w_n}{w_s}$.

on the interval $[0, 1]$. Table 4.1 shows the empirical assessment of this ratio. As follows from the Table, the pair $(1; 1.5)$ for (w_n, w_s) yields the best result, and w_n is set to 1 and w_s is set to 1.5.

4.4 Experiments

The proposed algorithm should outperform other weka's classifiers on a benchmark data set. Having run all the *weka* classifiers on the benchmark data set *EMO-DB* (which I described in Section 2.3.1 of Chapter 2) I already know that the multilayer perceptron is the top performer. Thus it becomes the competitor for the *TGI+*.

I carried out the experiments on a benchmark data set for acted emotional speech *EMO-DB*, which described in Section 2.3.1 of Chapter 2.

As for the testing protocol, 10-fold cross-validation was used. Figure 4.3 depicts the division of data into the training and testing sets for different folds: each time one tenth part is reserved for testing, the 39% of the training set is used for syntactic training and the whole training set (39% and remaining 61%) is used for statistical training.

Recall, precision and F-measure per class are given in Tables 4.2, 4.3 and 4.4 for the C4.5, the multilayer perceptron and the *TGI+*, respectively. The overall accuracy of the multilayer perceptron, the state of the art recogniser, is 73.9% and the overall accuracy of the *TGI+* is 78.58%, which is a 4.68% ± 3.45% in favour of the *TGI+*.

The *TGI+* has been evaluated against the C4.5 to find out which is the contribution of moving from the feature vector representation of samples to the distance-to-automaton one. The C4.5 performs with 52.9% of accuracy, which is 25.68% less than the *TGI+*.

4.5 Main property of *TGI+*

In this section I will discuss the main property of the *TGI+* and a consequent further application for the method.

Unlike many other machine learning schemes, the TGI+ is a human-readable classification method. It consits of two phases:

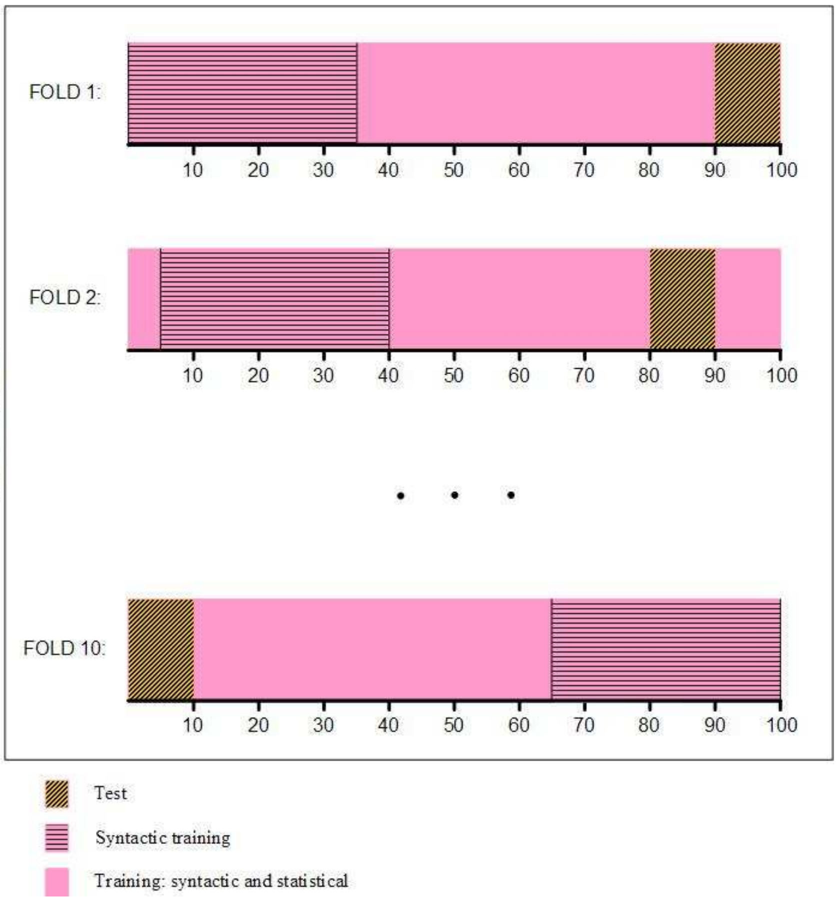


Figure 4.3: The training/testing protocol for the *TGI+*.

class	precision	recall	F-measure
fear	0.49	0.44	0.46
disgust	0.26	0.24	0.26
happiness	0.35	0.36	0.35
boredom	0.49	0.55	0.52
neutral	0.51	0.46	0.49
sadness	0.71	0.82	0.76
anger	0.69	0.70	0.70

Table 4.2: The C4.5 on the EMO-DB.

class	precision	recall	F-measure
fear	0.82	0.74	0.77
disgust	0.72	0.74	0.73
happiness	0.52	0.49	0.51
boredom	0.73	0.75	0.74
neutral	0.71	0.78	0.75
sadness	0.88	0.94	0.91
anger	0.75	0.76	0.75

Table 4.3: The MLP on the EMO-DB.

class	precision	recall	F-measure
fear	0.66	0.66	0.66
disgust	0.60	0.60	0.60
happiness	0.86	0.73	0.81
boredom	0.81	0.72	0.77
neutral	0.64	0.79	0.71
sadness	0.83	0.83	0.83
anger	0.89	0.93	0.91

Table 4.4: The *TGI+* on the EMO-DB.

1. the human-readable modeling with features based on the idea of simple distance to a prototypical expression of the emotion;
2. the human-readable decision tree based on the assessment of how close the expressed emotion is to the sets of rules describing each emotion.

Unlike other methods, the *TGI+* can describe in simple terms what in the speech pattern prevents it from being recognized as this or that emotion. One of the applications following from this property lies in the clinical context to assess the capability of the patient with disorders that affect their ability to express speech emotion.

4.6 Discussion

4.6.1 Correctness of algorithm construction

While constructing the *TGI+*, it is of critical importance that the following condition holds:

- The accuracy of the *TGI+* is better than that of tree automata and the C4.5.

If this condition holds, then the *TGI+* is well constructed, that is its parts, TA and C4.5, can classify, but less efficiently than their combination. I tested the *TGI+*, tree automata and the C4.5 on the same *EMO-DB* database under the same experimental settings.

With respect to tree automata, if the sample x is accepted by more than one automaton, then ties are resolved randomly:

$$D(x) = rand(A_1, ..., A_k), \tag{4.26}$$

where $x \in L(A_1) \wedge ... \wedge x \in L(A_k)$. The tree automata performed with 43% of accuracy, which is 35.58% worse than the accuracy of the *TGI+*. The C4.5 performed with 53% of accuracy, which is 25.7% worse than the *TGI+*. Therefore the condition is met and I can state that I arrived to a meaningful combination of the methods from different pattern recognition paradigms.

4.6.2 Selection of C4.5 as a base classifier in *TGI+*

I considered the possibility of having the MLP in place of the C4.5. The accuracies dramatically went down and abandoned this alternative. Different features go well with different classifiers, and in the literature their combinations are often determined via experimentation.

For decision trees feature construction has long been considered a powerful tool for increasing both accuracy and understanding of structure, particularly in high-dimensional problems [5]. Feature Construction is the application of a set of constructive operators to a set of existing features resulting in construction of new features. And this is what the *TGI+* does: it first constructs new features from the low-level features and then uses a decision tree for classification.

4.7 Conclusions

I have adapted a combined classification approach coming from the optical character recognition research to the task of speech emotion recognition, that required devising new algorithms for the grammar inference stage. The syntactic part implements a tree grammar inference algorithm. The statistical part implements the entropy decision tree classifier C4.5. I have further extended the idea with an built-in feature selection procedure. I tested the classifier on a benchmark data set. The proposed classifier outperformed a state of the art classifier, the multilayer perceptron, with a statistically significant difference in accuracies of 4.68% and the baseline of the C4.5, by 26.58%.

The main property of the *TGI+* is human-readability of the classification process, which is of a potential application for example in the clinical context as an assessment and training tool for patients with impaired capabilities to express speech emotions.

Chapter 5

Distance-to-Automaton Features

Currently one of the central open problems in SER is the search for novel features, especially perceptually adequate and high-level ones. There are two ways of doing so: defining new signal features, as for example in [36] and [41] or *constructing* high-level features from the existing signal low-level features.

Feature Construction has long been considered a powerful tool for increasing both accuracy and understanding of structure, particularly in high-dimensional problems [5]. Feature Construction [33], [47] is the application of a set of constructive operators to a set of existing features resulting in construction of new features. Examples of such constructive operators include checking for the equality conditions $\{=, \neq\}$, the arithmetic operators $\{+, -, \times, /\}$, the array operators $\{max(S), min(S), average(S)\}$ as well as other more sophisticated operators were proposed, for example $count(S, C)$ [3] that counts the number of features in the feature vector S satisfying some condition C. As for constructed features in SER, evolutionary programming techniques were applied in [55].

In this chapter, new constructed features for SER are proposed. In order to understand the principles of their construction, let us recall the principles of the *TGI+* described in the previous chapter. First the objects are modeled by means of a syntactic method, that is the samples are mapped into their representations. A representation of a sample is a set of the measurements signifying to which degree the sample resembles the averaged pattern of each recognition class. Then, the mappings are classified with a statistical method. These mappings can be used as the constructive features to be early-fused with the low-level features from which they were calculated. Figure 5.1 depicts the process of feature calculation. Thus, I propose the high-level features, which are the distances from a feature vector to the tree automaton accepting class i, for all i in the set of the class labels. I propose to concatenate the set of the low-level features and the set of the high-level features, submit the resulting set to the feature selection procedure and then to do the classification step in the usual way. Figure 5.2 illustrates this scenario of the classification with early-fusion.

The structure of this chapter is as follows: Section 5.1 provides a description of the set of the low-level features, used for this experiment, in Section 5.2 I explain my idea of the high-level distance to automaton features in greater detail and following it construct such features. In Section 5.3 I describe the data set and the experimental

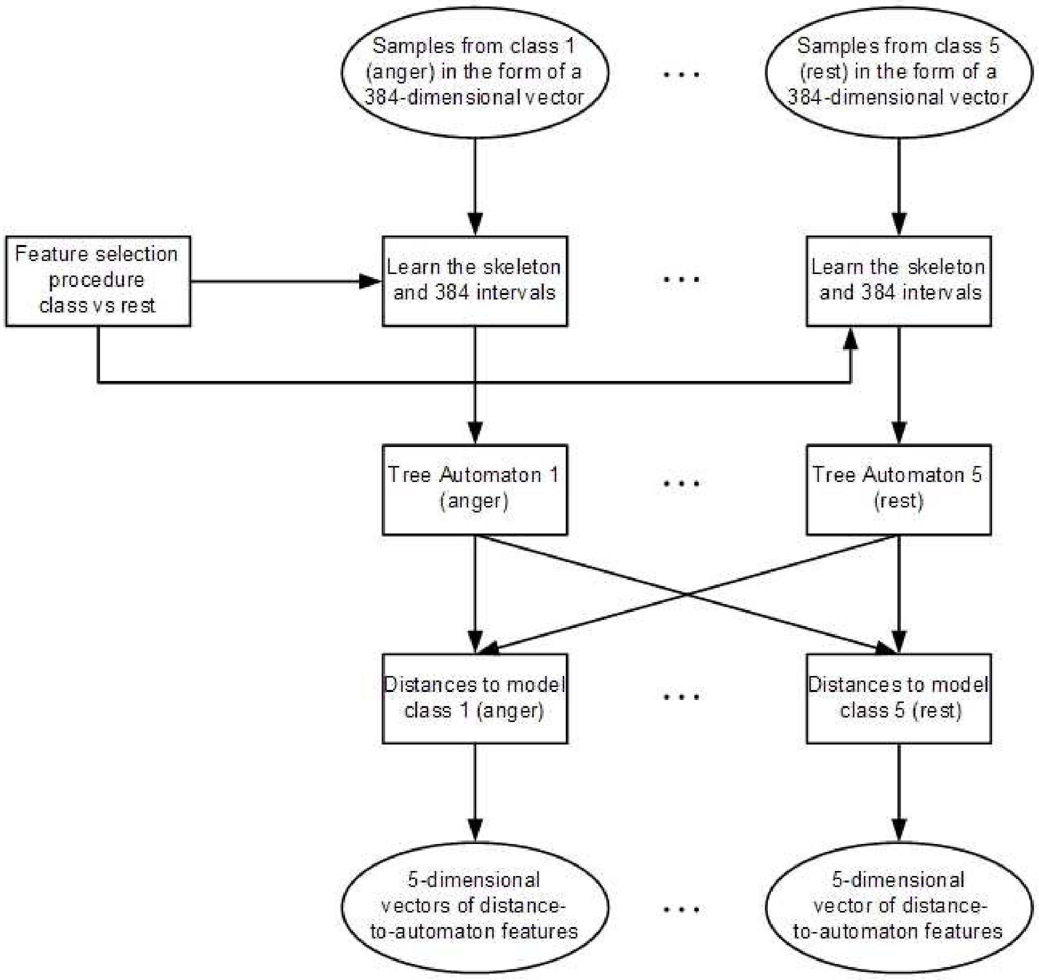

Samples from class 1 (anger) in the form of a 384-dimensional vector
...
Samples from class 5 (rest) in the form of a 384-dimensional vector
Feature selection procedure class vs rest
Learn the skeleton and 384 intervals
...
Learn the skeleton and 384 intervals
Tree Automaton 1 (anger)
...
Tree Automaton 5 (rest)
Distances to model class 1 (anger)
...
Distances to model class 5 (rest)
5-dimensional vectors of distance-to-automaton features
...
5-dimensional vector of distance-to-automaton features

Figure 5.1: Calculation of the distance-to-automaton features.

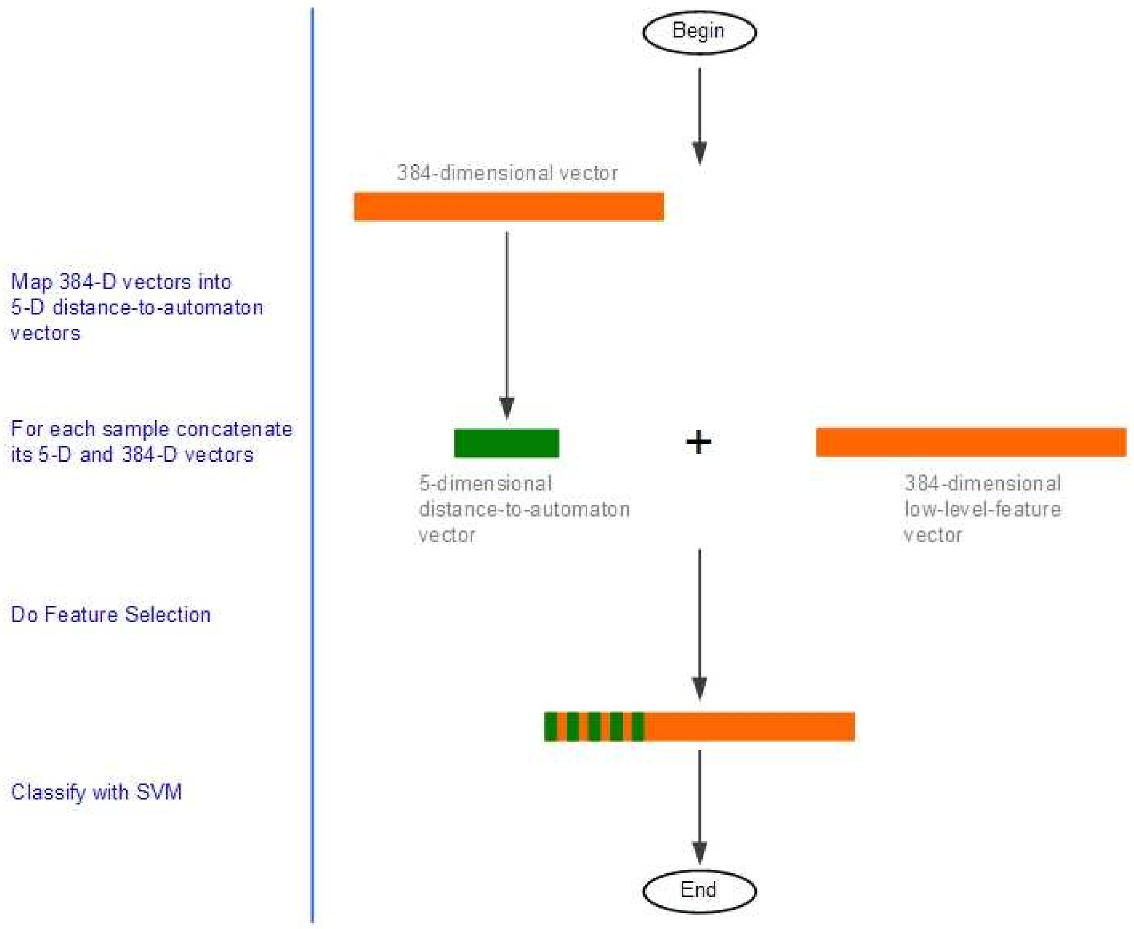

Begin
384-dimensional vector
Map 384-D vectors into 5-D distance-to-automaton vectors
For each sample concatenate its 5-D and 384-D vectors
+
5-dimensional distance-to-automaton vector
384-dimensional low-level-feature vector
Do Feature Selection
Classify with SVM
End

Figure 5.2: The classification scheme with early fusion.

settings. In Section 5.4 conclusions are drawn.

5.1 Low-level features

The openSMILE feature extraction system [17] was used to extract speech statistics from the Aibo corpus. Although this switch from the *ESEDA* feature extraction module to openSMILE was a condition to use Aibo, it was also beneficial, since openSMILE was meant for applications in both speech and music and thus allows for a greater divercity of low-level features than *ESEDA*.

There were 16 low-level descriptors:

- zero-crossing rate from the time signal,
- root mean square frame energy,
- pitch frequency normalized to 500Hz,
- harmonics-to-noise ratio by autocorrelation function,
- $1-12$ mel-frequency cepstral coefficients in full accordance to HTK-based computation.

To each of these the delta coefficients were additionally computed. Next for each sample the following 12 functionals were calculated:

- mean,
- standard deviation,
- kurtosis,
- skewness,
- minimum value,
- maximum value,
- relative position and range as well as two linear regression coefficients with their mean square error.

Thus, the resulting feature vector per sample contains $12 \times 2 \times 12 = 384$ attributes.

5.2 Distance to tree automata features

The high-level features are distances from a feature vector to the tree automaton accepting class i, for all i in the set of class labels. There are as many such features per sample as there are recognition classes, i.e. five in the case of Aibo. The scheme for calculation of the high-level features is depicted in Figure 5.1. After having calculated the five distance-to-automaton features for each sample:

1. the set of low-level features and the set of high-level features are concatenated;
2. their union is submitted to a feature selection procedure;
3. the classification step is done in the usual way.

Figure 5.2 illustrates the scheme of this classification with early fusion.

5.2.1 Calculation of Distance-to-Automaton Features

In this section, following the ideas above the distance-to-automaton features will be constructed.

Step 1: In order to perform tree grammar inference, samples are represented by tree structures.

For all classes the skeletons of all tree structures have the same shape determined by the grammar below. S denotes a start symbol of the formal grammar (in the sense of a term-rewriting system):

$\{\ S \longrightarrow ProsodicFeatures\ SegmentalFeatures;$

$ProsodicFeatures \longrightarrow Pitch\ Pitch_{\Delta}\ ZeroCrossingRate\ ZeroCrossingRate_{\Delta}\ Energy\ Energy_{\Delta};$

$SegmentalFeatures \longrightarrow$
$HarmonicToNoiseRatio\ HarmonicToNoiseRatio_{\Delta}\ MFCC\ MFCC_{\Delta};$

$Pitch \longrightarrow Min\ Max\ MaxPosit\ MinPosit\ Range\ Mean\ Std\ Skewness\ Kurtosis\ LinRegO\ LinRegS$
$LinRegMSE;$

$Pitch_{\Delta} \longrightarrow ...$
$\}$

The "..." signifies that I continue structuring the space of the low-level features. All the trees have 384 leaves, each corresponding to one of the 384 features from the feature vector.

Step 2: Apply tree grammar inference to learn the five automata, each accepting a different recognition class. This step implements the algorithm proposed in Section 4.2.3 of this thesis.

Although the automata for different recognition classes have the same skeletal shape, they differ with respect to weights on the branches. The set of the more important features are determined exclusively for every class (*anger*, *neutral*, etc.) through a feature selection procedure. The feature selection procedure implements correlation based feature selection, the algorithm for which was explained in Section 2.4.2 *Feature Selection* of Chapter 2.

The leaves of the tree structure corresponding to a concrete emotional sample are the numeric values from the sample's feature vector. The leaf nodes of the tree automata are the numeric intervals a particular feature takes for a given class in the training set. These are learned through grammar inference.

The numeric intervals for the leaf nodes were learned from 30% of the training set. The percentage of the training set that should be fed to the grammar inference algorithm was taken the same as in the previous chapter, where it was chosen via experimentation.

Step 3: Calculate the distance-to-automaton features. This step implements the algorithm that was proposed in Section 4.2.4 of this thesis.

While calculating the edit costs between automaton and samples, if all the numeric leaves are within the intervals the automaton has for them, then the edit distance from the automaton to the sample is 0. Otherwise, the cost of the modifications in the grammar of the acceptor in order to accept the sample is calculated. The rules for feature-selected nodes are more expensive to modify. More specifically, $w_s = 1.5$ and $w_n = 1$. The pair of weight values were taken the same as in the previous chapter, where they were chosen via experimentation.

5.3 Experiments

The data set was the training set of the Aibo corpus (as was provided for the INTERSPEECH challenge [57]), and Section 2.3.3 can be referred to for the corpus description. There were five recognition classes: *Anger*, *Emphatic*, *Neutral*, *Positive*, and *Rest*.

In order to see to which type of the classifier the type of the proposed features suits best, I took several conceptually different classifiers: the rule-based classifier $RIPPER_k$[1] [11], the neural network MLP [71], and the geometric classifier SVM [67]. A description of these classifiers can be found in Appendix A.

The baseline to beat is the performance of these three classifiers on the low-level features, selected by the feature selection procedure. The experiments were carried out in 10-fold cross-validation.

For each sample, five distance-to-automaton features were calculated, then the set of the original features was concatenated with the set of the newly calculated distance-to-automaton features, and their union was fed to a feature selection procedure. As a result, two distance-to-automaton features and 45 other low-level features were selected. The 48-dimensional vectors became the input for the three classifiers.

5.4 Results and Discussion

The results of the contrastive testing, with and without distance-to-automaton features, are presented in Table 5.1. The column *baseline* reports accuracies for the multilayer perceptron, the support vector Machine, and the $RIPPER_k$ [11] working on the low-level feature vectors. The column *with TA features* reports accuracies for the same classifiers working on the vectors containing both, the proposed distance-to-automaton features and the low-level features. The inclusion of the high-level features into the feature vector led to the unstable improvements in accuracy: from

[1] $RIPPER_k$ stands for Repeated Incremental Pruning to Produce Error Reduction.

Classifier	baseline	with TA features	improvement
MLP	56.57%	57.67%	1.1.%
SVM	**61.35**%	61.48%	0.13%
$RIPPER_k$	59.53%	**68.61**%	9.08%

Table 5.1: Accuracies on the low-level and the fused vectors.

notable 9% for the rule based classifier $RIPPER_k$ to a negligibly small improvement (1% and 0.13%) on the multilayer perceptron and the support vector machine. The support vector machine was the top performer on the low-level feature set. Yet, when the $RIPPER_k$ operates on the fused feature set, it outperforms the support vector machine by 7%. The $RIPPER_k$ is a decision tree classifier which was shown to be competitive with the C4.5 with respect to error-rates but much more efficient on noisy big data sets [11]. Thus, the classification of the fused set of low-level- and distance-to-automaton features with the $RIPPER_k$ is the evolution of the *TGI+* idea and its adaptation to face real-life data.

My intuition is that the distance-to-automaton features should be perceptually adequate, since they are a general descriptive measure of how far a given sample is from the class prototype, where the class prototype is modeled with a tree automaton.

5.4.1 Conclusions

In this chapter I proposed the high-level features, which are the distances from a feature vector to the tree automaton accepting class i, for all i in the set of the class labels. The automata are trained to operate on feature vectors through the grammar inference procedure proposed in the previous chapter. There are as many tree automaton features as there are recognition classes. I proposed to early-fuse the set of the low-level features and the set of the high-level features, submit the resulting set to a feature selection procedure and then to do the classification step in the usual way.

According to the experimental results, inclusion of the proposed features in the feature set leads to an improvement in accuracies, at least when the proposed features are used together with a rule-based classifier. The proposed classification scheme of the $RIPPER_k$ run on the fused set of the low-level and the distance-to-automaton features outperformed the state of the art top performer (the SVM) with a statistically significant difference in accuracies of 7%.

The classification of the fused set of low-level- and distance-to-automaton features with the $RIPPER_k$ is the evolution of the *TGI+* idea and its adaptation to face big and noisy real-life data sets.

Chapter 6

Results

In this chapter I draw conclusions with respect to my contributions:

1. the *ESEDA* tool for enhanced speech emotion recognition and analysis;
2. the *TGI+* classifier;
3. the high-level distance-to-automaton features.

6.1 The *ESEDA* system and learning from classification errors

The *ESEDA* [59], [62] is based on the supervised pattern recognition cycle. The classical part of the system is comprised of the three modules: feature extraction, feature selection, and classification. Its performance served as a baseline to validate the new theory developed in this thesis. The *ESEDA* was tested on acted and real emotions, and on five languages. The testing results allow me to conclude that the system is ready to be integrated into real-life applications.

To enhance the classical design, the *ESEDA* has an exclusive block of error prevention. The underlying idea is to analyse the confusion matrix on the validation set and design a module that prevents the system from making these errors on new material. Despite its simplicity, the module led to notable improvements in accuracy.

6.2 *TGI+* classifier

I have proposed a classifier for SER, called the *TGI+*[1], [61] of a mixed design with syntactic and statistical learning. The syntactic part implemented tree grammar inference, and the statistical part implements an entropy decision tree classifier. First, by means of a syntactic method the samples are mapped into their representations. A representation of a sample is the set of seven numeric values, signifying to which degree a given sample resembles the averaged pattern of each of the seven classes. Second, the mappings of samples, not the initial feature vectors, are classified with a

[1]TGI stands for Tree Grammar Inference and the plus is for the statistical learning enhancement.

statistical method. I have also extended the grammar inference part with a feature selection procedure to penalise more important features with higher edit costs for being outside the interval, which the tree automata learned at the inference stage.

The *TGI+* has been evaluated against the weka's top performer for the benchmark data set EMO-DB, which turned out to be the multilayer perceptron. The *TGI+* outperformed the multilayer perceptron by 4.68%.

6.3 High-level features based on distances to tree automata

I have proposed the high-level features, which are the distances from a feature vector to the tree automaton accepting class i, for all i in the set of the class labels. The automata are trained to operate on feature vectors through the grammar inference procedure. There are as many tree automaton features as there are recognition classes. I proposed to early-fuse the set of the low-level features and the set of the high-level features, submit the resulting set to a feature selection procedure and then to do the classification step in the usual way.

According to the experimental results, inclusion of the proposed features in the feature set leads to an improvement in accuracies, at least when the proposed features are used together with a rule-based classifier. The proposed classification scheme of the $RIPPER_k$ run on the fused set of the low-level and the distance-to-automaton features outperformed the state of the art top performer (the SVM) with a statistically significant difference in accuracies of 7%.

The classification of the fused set of low-level- and distance-to-automaton features with the $RIPPER_k$ is the evolution of the *TGI+* idea and its adaptation to face big and noisy real-life data sets.

Chapter 7

Future Work

In this chapter I discuss the potential impacts of my contributions in both research and applications.

7.1 Medical Application

Emotional prosody is affected in a few language pathologies, including the *foreign accent syndrome*. A potential application of SER systems, which would be of use for speech pathology specialists is automatic assessment of acted speech emotion in the clinical context to evaluate emotional prosody skills of the patient. According to the practitioners it is difficult to use naive judges to assess emotional prosody. Ideally, a SER system would provide objective measurements of the patient's progress.

7.1.1 Task Formulation

The immediate problem one runs into using a state of the art SER system is that computers recognize emotions not in the same way than humans do and classification algorithms were designed for the computer with the built-in Turing computation, which is not the same as the human brain processing. As was reported in the SER literature, computers can recognize emotions better than human judges, but the purpose of the application is to help the patient learn how to express emotions to be understood by other people rather than to have her emotions accurately recognized by the computer. Also with respect to features, it was repeatedly shown in the psycholinguistic literature that cognitively adequate features are not very useful in the computerized SER. A requirement for the medical SER would be that emotions are assessed in a more human way. To bridge this gap between human and computer perception of the emotion, I propose to start with a more human-like classification, for which the *TGI+* is a suitable framework.

There is extensive psychological literature showing that human pattern recognition punishes formation of subpatterns of another recognition class more than it punishes other deviations from the prototypical pattern [20], [63]. Human cognition is based on "sub-patterns" stored in and processed by neural clusters. For example, for the human ear the case when the pitch pattern comes from another emotion, and the rest complies with the emotional pattern, is a much less successful expression of emotion than the one when one parameter of pitch is incorrect and one parameter of intensity, and one of tempo – but all together these inconsistencies do not form another pattern. In the first case the cluster of neurons responsible for pitch will fire "we have recognized an inconsistent emotional pattern", and the human pattern recognition would need to backtrack to reconsider the decision.

7.1.2 *TGI+ cognitive*

In order that the formation of subpatterns of a different class is punished, I changed the edit distance calculation algorithm from Chapter 4. In the *TGI+* for computer SER the edit distances were calculated exclusively via the *substitution* operation and with the tree structure not fully exploited. In the case of the *TGI+*$_c$ the tree structure is fully exploited and there are three levels of subpatterns:

1. the level of acoustic parameters (pitch, intensity, etc);
2. the level of prosodic impression (intonation) and segmental impression (voice quality);
3. the level of perception of emotion.

The above is said in the grammar:

{S$\longrightarrow$ *ProsodicFeatures SegmentalFeatures*;

ProsodicFeatures $\longrightarrow$ *Pitch Intensity Energy*;

SegmentalFeatures $\longrightarrow$ *Jitter Shimmer Formants Harmonicity*;

Pitch $\longrightarrow$ *Min Max Quantile Mean Std MeanAbsoluteSlope*;

etc.
}

Moreover, the operation of *substitution* is allowed only at the leaf level, provided that the leaves in the subpattern do not form a subpattern of a different emotion from the one to which the distance is being calculated. If a subtree has been reduced to a subpattern of a different pattern (e.g. while calculating the edit-distance to *anger*, the subpattern of pitch is acceptable by the *neutral* TA), then for the edit-distance calculation algorithm it takes to first erase the node paying the deletion costs and redraw it paying the insertion costs. As in the case of the *TGI+*, the costs are sent upwards, and the tree is processed in the bottom-up manner in post-order enumeration. The higher is the level of a wrong subpattern to which the tree has been reduced, the more costly the repairment is, since there are more nodes to erase and than to insert. Thus, the formation of wrong subpatterns is punished.

I implemented this new version of the *TGI+*, called *TGI+*$_c$, where c stands for *cognitive*. The automatic recognition accuracy on the *EMO-DB* is 61%, which is better than C4.5 but worse than that of other powerful state of the art classifiers, like the *TGI+*, the support vector machine or the multilayer perceptron.

7.1.3 Data

In collaboration with CIMES (Centro de Investigaciones Medico Sanitarias, University of Málaga) I plan to carry out research on the *FAS* data [39]. Two subjects suffering from the *foreign accent syndrome* were recordered five times: pretreatment and four times after long treatment periods. Each time the patient had to read 10 neutral sentences with the four intonations: *neutral*, *happy*, *sad* and *angry*. Thus there are 200 sentences by each patient.

7.1.4 Hypothesis

In this thesis edit distances were applied to the analysis of emotional speech. Edit distance measures how far a given utterance is from the pattern. For example, how far the patient still is from healthy prosody or her previous imitations. I expect edit distances to form the four numeric sequences that would converge to 0 for the aimed emotion and to some k_i greater than 0, for $i = 1, 2, 3, 4$. The correct values of k_i are learned from the corpus of speech with emotional imitations by healthy people.

7.1.5 Usability

The *ESEDA* tool with the $TGI{+}_c$ function will provide a graphical user interface. For the patient's speech sample it draws a tree graph that shows:

- human-readable assessment of correctness of individual acoustic parameters;
- assessment of the closeness of the patient's imitations to the healthy prototypical expression of emotion;
- keeps track of the patient's progress through time.

7.2 Future Research

As far as the future directions of research are concerned, I can indicate the following two paths:

1. feature selection for features embedded in a tree structure,
2. search for a more cognitive algebra for the $TGI{+}_c$.

7.2.1 Feature Selection Embedded in Tree Structure

In the *TGI+.2* I put weights on the initial feature vector, which are determined via a feature selection procedure that treats features as equal elements of a feature set. Feature selection should be possible to be done taking into account the position of the node in the graph.

7.2.2 Towards a more cognitive algebra for $TGI{+}_c$

In tree automata theory rules of the automaton are referred to as *algebra* (as opposed to leaves that are called *constants*). The $TGI{+}_c$ is described in Section 7.1.2. It is based on the idea of the subpattern and fully exploits the hierarchical structure. In the search for a more cognitive algebra, I plan to further explore the psycholinguistic literature.

Appendix A

Three classifiers

In this Appendix I include the explanations of the multilayer perceptron, the support vector machines, and the $RIPPER_k$. These classifiers were the top performers on the databases I work with and therefore they served as baselines for the classification methods proposed.

A.1 Multilayer Perceptron

Artificial neural networks (NNs) [30], [19] were designed with the idea to mathematically model human intellectual abilities by biologically plausible engineering design and to benefit from parallel computation. NNs have been a popular classifier choice in SER experiments [21], [22], [46], [66]. The multilayer perceptron (MLP) is the most widely used NN variant.

The multilayer perceptron outperformed all other classifiers from the state of art weka collection of classifiers. Therefore the multilayer perceptron becomes a competitor for the *TGI+* classifier propose in Chapter 4 *TGI+ classifier.*

For an n-dimensional pattern recognition problem with c classes, a neural network obtains a feature vector:

$$\mathbf{x} = [x_1, ..., x_n] \in \Re^n$$

as its input, and produces values for the c discriminant functions

$$g_1(x), ..., g_c(x).$$

NNs are usually trained to minimize the squared error E on a labeled training set $\mathbf{Z} = \{z_1, ..., z_n\}$, where $z_i \in \Re^n$, and $l(z_j) \in \Omega$:

$$E = \tfrac{1}{2}\Sigma_{j=1}^{N}\Sigma_{i=1}^{c}\{g_i(z_j) \text{ - } I(w_i, l(z_j))\}^2 \text{ (2.)}$$

where I is an indicator function: $I(a, b)$ takes value 1 if $a = b$ and 0 if $a \neq b$. The factor $\frac{1}{2}$ is included for convenience.

The computational model of the neuron is presented in Figure A.1. Let $\mathbf{u}$ be the input vector to the node:

$$\mathbf{u} = [u_0, ..., u_q] \in \Re^{q+1}.$$

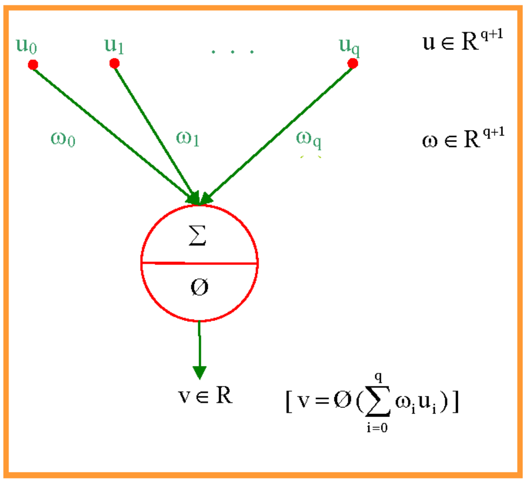


Figure A.1: The computational model of the neuron.

and $v \in \Re$ be the output. Let $\mathbf{w}$ be a vector of synaptic weights:

$$\mathbf{w} = [w_0, ..., w_q] \in \Re^{q+1}.$$

The processing element implements the function:

$$v = \phi(\xi), \text{ where } \xi = \Sigma_{i=0}^{q} w_i u_i, \ (3.)$$

where $\phi : \Re \longrightarrow \Re$ is the activation function and ξ is the net sum. Typical choices for ϕ are:
The threshold function:

$$\phi(\xi) = 1 \text{ if } \xi \geq 0$$

$$\phi(\xi) = 0, \text{ otherwise.}$$

The sigmoid function:

$$\phi(\xi) = \frac{1}{1+exp(-\xi)}.$$

The identity function:

$$\phi(\xi) = \xi. \ (4.)$$

The hyperbolic tangent function:

$$\phi(\xi) = tan(\xi).$$

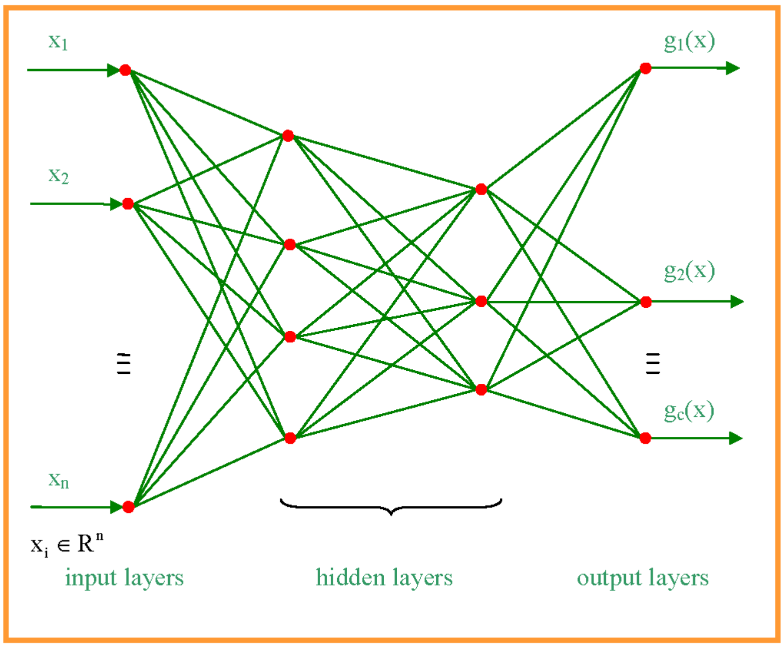


Figure A.2: Multilayer perceptron.

The sigmoid is the most widely used, for the reasons that:

- it can model both linear and threshold functions to a desirable precision, and
- it is differentiable, which is important for NN training algorithms: $\phi'(\xi) = \phi(\xi)[1 - \phi(\xi)]$.

The weight $-w_0$ is used as a bias, and the corresponding input value u_0 is set to 1. Expression (3.) can be rewritten as:

$$v = \phi[\zeta - (-w_0)] = \phi[\Sigma_{i=1}^{q} w_i u_i - (-w_0)],$$

where ζ is now the weighted sum of the weighted inputs from 1 to q. Geometrically the equation:

$$\Sigma_{i=1}^{q} w_i u_i - (-w_0) = 0 \ (6.)$$

defines a hyperplane in $\Re^q$. A node with a threshold activation function responds with value 1 to all inputs $[u_1, ..., u_q]$ on the one side of the hyperplane and with value 0 to all inputs on the other side.

In the late 1950s Rosenblatt defined the famous perceptron with its training algorithm. The perceptron was implemented as defined in (3.); its threshold activation function $\phi(\xi)$ takes 1 if $\xi \geq 0$ and -1, otherwise. This single-neuron classifier separates two classes in $\Re^n$ with a linear discriminant function defined by $\xi = 0$. The algorithm starts with random initial weights $\mathbf{w}$ and modifies them, as each subsequent sample from $\mathbf{Z}$ is fed to the perceptron. The modification is made only the current vector $\mathbf{z}_j$ is misclassified (appears on the wrong side of the hyperplane). The weights are corrected by

$$\mathbf{w} \longleftarrow \mathbf{w} - \nu\eta\mathbf{z}_j, \text{ (6.)}$$

where ν is the output of the perceptron for z_j and η is the learning rate.

Beside its simplicity, perceptron training has the following interesting properties:

1. *The perceptron convergence theorem.* If the two classes are linearly separable in $\Re^n$, the algorithm always converges in finite number of steps to a linear discriminant function that gives no resubstitution errors on $\mathbf{Z}$ for any η.

2. If two classes are not linearly separable in $\Re^n$, the algorithm will loop infinitely through $\mathbf{Z}$ and never converge. There is no guarantee that if the procedure terminates, the resulting linear function is the best one found throughout the training.

A neuron can represent only a linear decision boundary. The way to approach non-linear classification is by connecting many perceptrons in a hierarchical structure called a multilayer perceptron (MLP). A MLP has an input layer, a hidden layer, and an output layer connected in a feed-forward manner; that is, each neuron receives signals from the previous layer as well as sends connections to the next layer (Figure A.2), and there are no connections between the nodes of the same level. Weights are associated with each input and every connection. A MLP receives an input pattern $\mathbf{x}$ and maps this into c discriminant functions $g_1(x), ..., g_c(x)$ at the output layer. The default perceptron properties are:

- the activation function at the input layer is the identity function (4.);
- there are no lateral connections between the nodes at the same layer (feed-forward structure);
- non-adjacent layers are not directly connected;
- all nodes at all hidden layers have the same activation function ϕ.

It was shown that a MLP with a single hidden layer and threshold nodes can approximate any function with a specified precision. The *Backpropagation algorithm* provides the means to train a neural network.

There are two aspects to the MLP's learning: learning the structure of the network and learning the connection weights. The backpropagation algorithm described below is an algorithm for determining the weights given a fixed neural network structure with already chosen number of hidden layers, number of nodes at each layer, and a differentiable activation function. However, although there are many algorithms that attempt to identify network structure, this aspect of the problem is commonly solved through experimentation.

Let Θ be a parameter of the NN and $J(\Theta)$ be some error function to be minimized. In our case this is the squared error function E of Equation (2). The gradient descent method updates Θ by

$$\Theta \longleftarrow \Theta - \eta \tfrac{\partial J}{\partial \theta},$$

where $\eta > 0$ is the learning rate.

An obvious candidate for $J(\theta)$ is the squared error E of the Equation (2).

Backpropagation training algorithm:

1. Initialize the training procedure: assign small random values to all weights (including biases) of the MLP. Specify the learning rate $\eta > 0$, the *max* number of epochs T, and the error goal $\varepsilon > 0$.

2. Set $E = \infty$, the epoch counter $t = 1$ and the object counter $j = 1$.

3. While ($E > \varepsilon$ and $t \leq T$) do
   (a)Submit z_j as the next training example.
   (b)Calculate the output of every node of the NN with the current weights (forward propagation).
   (c)Calculate the error term δ at each node at the output layer by $\delta = \delta_i^0$.
   (d)Calculate recursively all error terms at the nodes of the hidden layers using δ_k^h (back-propagation):

   $$\delta_k^h = (\Sigma_{i=1}^c \delta_i^0 w_{ik}^0) \tfrac{\partial \theta(\xi_k^h)}{\partial \xi_h^k}.$$

   (e)For each hidden and each output node update the weights by

   $$w_{new} = w_{old} - \eta \delta u,$$

   (f)Calculate E using the current weights and (2.).
   (g)If $j = N$ [a whole pass through $\mathbf{Z}$ (epoch) is completed], then set $t = t + 1$ and $j = 0$. Else, set $j = j + 1$.

4. End while.

To overcome the inherent disadvantages of the pure gradient descent, RProp (for resilient propagation) was proposed [50]. RProp performs a local adaptation of the weight-updates according to the behaviour of the error function.

Being a powerful classifier, the MLP is often used in SER. In [21] experiments were conducted on the database of acted speech emotions in English, Slovenian, Spanish and French. Their SER system is based on all features used by [45] with duration features, with the sum of absolute pitch and energy difference which pertain to prosody, and with shimmer and jitter, which pertain to speech quality. Their classification system was based on a MLP with 144 input neurons, 7 output neurons, one for each of six emotional states in addition to the neutral condition, and 49 neurons in the hidden layer. The MLP was trained with the Rprop algorithm using Stuttgart Neural Network Simulator version 4.2. Their testing protocol was 80% of data for the training of the neural network, and the remaining 20% were used for testing. The max-correct method [42] was used to evaluate the correctness of the entire vector. One output vector denoted the emotion of one utterance. Elements in the output and target vector that have the maximum values were searched for. If the element with the maximum value in the target vector and the element with the highest vector denoted the same emotion, then the vector was defined as correct. It evaluated the correctness of the entire output vector, which corresponds to the emotion.

A.2 Support Vector Machines

The paragraph describing the support vector machine (SVM) [19] is included, because it turned out to be the top performer on the *Aibo* data set. In the experiment in question the SVM was a top performer when the classification was done on normal feature vectors. Therefore its performance is an important baseline to beat for a classifier working on fused feature vectors, which proposed in Chapter 5 *Distance to Automaton Features.*

The SVM relies on preprocessing the data to represent patterns in high dimension – typically much higher than the original feature space. With an appropriate nonlinear mapping $\varphi(\cdot)$ to a sufficiently high dimension, data from two categories can always be separated by a hyperplane. For each of the n patterns, $k = 1, 2, ...n$, we let $z_k = \pm 1$, according to whether pattern k is in w_1 or w_2. A linear discriminant in an augmented y space is

$$g(\mathbf{y}) = \mathbf{a}^t\mathbf{y}, \tag{A.1}$$

where both the weight vector and the transformed pattern vector are augmented (by $a_0 = w_0$ and $y_0 = 1$, respectively). Thus a separating hyperplane ensures

$$z_k g(\mathbf{y}_k) \geq 1, \tag{A.2}$$

for $k = 1, 2, ..., n$.

The goal in training an SVM is to find the separating hyperplane with the largest margin; we expect that the larger the margin, the better generalization of the classifier. The distance from any hyperplane to a (transformed) pattern $\mathbf{y}$ is $\frac{|g(\mathbf{y})|}{\|a\|}$, and assuming than a positive margin b exists, Equation A.2 implies

$$\frac{z_k g(\mathbf{y}_k)}{\|a\|} \geq b, \tag{A.3}$$

for $k = 1, 2, ..., n$. The goal is to find the vector $\mathbf{a}$ that maximizes b.

The *support vectors* are the (transformed) training patterns for which Equation A.2 represents an equality, that is the support vectors are equally close to the hyperplane. The support vectors are the training samples that define the optimal separating hyperplane and are the most difficult patterns to classify.

If N_s denotes the total number of support vectors, then for n training patterns the expected value of the generalization error rate is bounded, according to

$$\varepsilon_n \leq \frac{\varepsilon_n[N_s]}{n}, \tag{A.4}$$

where the expectation is over all training sets of size n drawn from the stationary distributions describing the categories. This bound is independent of the dimensionality of the space of the transformed vectors, determined by $\varphi(\cdot)$. Suppose we have n points in the training set, we train an SVM on $n-1$ of them, and we test on the single remaining point. If that remaining point happens to be a support vector for the full n sample case, then there will be an error, otherwise there will not.

The first step of the training is to choose the non-linear φ-functions that map the input to higher-dimensional space. Often this choice is based by the knowledge of the domain. In the absence of such information, polynomials, Gaussians, or other basic functions can be used. The dimensionality of the mapped space can be arbitrary high – in practice limited to the computational resources.

We begin by recasting the problem of minimizing the magnitude of the weight vector constrained by the separation into a unconstrained problem by the method of Lagrange undetermined multupliers. Thus from Equation A.3 and our goal of minimizing $\|a\|$, we construct the functional

$$L(\mathbf{a}, \alpha) = \frac{1}{2}\|\mathbf{a}\|^2 - \sum_{k=1}^{n} \alpha_k [z_k \mathbf{a}^t \mathbf{y}_k - 1] \tag{A.5}$$

and seek to minimize $L()$ with respect to weight vector $\mathbf{a}$ and maximize it with respect to undetermined multipliers $\alpha_k \geq 0$. The last term in Equation A.5 expresses the goal of classifying the points correctly. One of the solution to this is through first applying the Kuhn-Tucker construction and then using quadractic programming.

A.3 *RIPPER$_k$*

RIPPER$_k$ [11] stands for Repeated Incremental Pruning to Produce Error Reduction. The *RIPPER$_k$* classifier is very compatible with C4.5 with respect to error-rates, but it much more efficient on large and noisy datasets. The *RIPPER$_k$* gave maximum improvement in its performance when the distance-to-automaton features were used and operating on the fused vector space it turned out to the top performer on the Aibo data set.

The *RIPPER$_k$* is a result of the evolution of pruning strategies: *PER* $\longrightarrow$ *IREP* $\longrightarrow$ *RIPPER$_k$*. One of the effective techniques to prune trees is *reduced error pruning*, *REP*. In *REP*, the training data is split into a growing set and a pruning set. First an initial rule set is formed that overfits the growing set, using some heuristic method. This overlarge rule set is then repeatedly simplified by applying one of a set of pruning operators. Typical pruning operators would be to delete any single condition or any single rule. At each stage of simplification, the pruning operator chosen is the one that yields the greatest reduction of error on the pruning set. Simplification ends when applying any pruning operator would increase error on the pruning set. *REP* for rules usually does improve a generalization performance on noisy data, however it is computationally expensive for large data sets.

In response to the inefficiency of *REP*, *incremental reduced error pruning* was proposed *IREP*. It was shown to be competitive with *REP* with respect to error rates and to also to be much faster.

Algorithm for 2-class version of *IREP*

```
procedure $IREP(Pos, Neg)$
begin
Ruleset := $\emptyset$.
while $Pos \neq \emptyset$ do
[grow and prune a new rule]
split$(Pos, Neg)$ into $(GrowPos, GrowNeg)$
Rule := GrowRule$(GrowPos, GrowNeg)$
Rule := PruneRule$(Rule, RulePos, RuleNeg)$
if the error rate of Rule on $(PrunePos, PruneNeg)$ exceeds 50%
then return Ruleset
else add Rule to Ruleset;
remove examples covered by Rule from $(Pos, Neg)$
endif
endwhile
return Ruleset
end
```

Further on the way of evolution, three modifications were added to *REP*:

1. a new metric for guiding its pruning phase;
2. a new stopping condition;
3. a technique for optimizing the rules learned by *IREP*.

To this the $RIPPER_k$ adds k iterations of an optimization step that more closely mimic the effect of non-incremental reduced error-pruning. Finally the $RIPPER_k$ was shown to outperform *IREP* on some benchmark datasets with a statistically significant improvement of recognition rates.

Appendix B

Low level features

The feature extraction module extracts 116 global statistical features, both prosodic and segmental. The acoustic parameters are pitch, intensity, formants and harmonicity. The module was implemented in the Praat [4] scripting language. This Appendix provides a full list of the features.

Features 1-5: various raw parameters: energy, power, intensity.

Feature 1: Energy
Feature 2: Power
Feature 3: Energy in air
Feature 4: Power in air
Feature 5: Intensity

Features 6-9 and 14-17: harmonicity Gne
Gne is the glottal-to-noise excitation ratio as proposed by [37].
Feature 6: Lowest X
Feature 7: Highest X
Feature 8: Lowest Y
Feature 9: Highest Y

Feature 14: numberOfRows
Feature 15: numberOfColumns
Feature 16: rawDistance
Feature 17: columnDistance
Feature 18: sum

Features 30-33: Harmonicity Cc && 34-37: Harmonicity Ac[1]

A Harmonicity object represents the degree of acoustic periodicity, also called Harmonics-to-Noise Ratio (HNR). Harmonicity is expressed in dB: if 99% of the energy of the signal is in the periodic part, and 1% is noise, the HNR is 10*log10(99/1) = 20 dB. A HNR of 0 dB means that there is equal energy in the harmonics and in the noise.

[1]CC stands for cross-correlation function, Ac stands for auto-correlation function.

Harmonicity can be used as a measure for:
- The signal-to-noise ratio of anything that generates a periodic signal.
- Voice quality. For instance, a healthy speaker can produce a sustained [a] or [i] with a harmonicity of around 20 dB, and an [u] at around 40 dB; the difference comes from the high frequencies in [a] and [i], versus low frequencies in [u], resulting in a much higher sensitivity of HNR to jitter in [a] and [i] than in [u]. Hoarse speakers will have an [a] with a harmonicity much lower than 20 dB. We know of a pathological case where a speaker had an HNR of 40 dB for [i], because his voice let down above 2000 Hz.

For harmonicity CC[2]:
Feature 30: minimum
Feature 31: maximum
Feature 32: mean
Feature 33: std

Feature 34: For harmonicity AC[3]:
Feature 35: minimum
Feature 36: mean
Feature 37: std

Features 38-53: statLtas:

Ltas is short for Long-Term Average Spectrum. An object of class Ltas represents the power spectral density as a function of frequency, expressed in dB/Hz relative to 2^{10-5} Pa.
Feature 38: Lowest frequency
Feature 39: highest frequency
Feature 40: numberOfBins
Feature 41: binWidth
Feature 42: binNumberFromFrequency
Feature 43: valueAtFrequency
Feature 44: valueInBin
Feature 45: minimum
Feature 46: frequencyOfMinimum
Feature 47: frequencyOfMaximum
Feature 48: maximum
Feature 49: mean
Feature 50: slope
Feature 51: localPeakHeight
Feature 52: localPeakHeight1
Feature 53: standard deviation

Features 10-13: global F0 statistics && Pitch (acc): Features 70-76 &&

[2]The abbreviation stands for "by cross-correlation function".
[3]The abbreviation stands for "by auto-correlation function."

Features 77-83: Pitch (cc):
The features are statistical measurements (mean, max, min, range, std) connected to the following.
A Pitch object represents periodicity candidates as a function of time. It does not mind whether this periodicity refers to acoustics, perception, or vocal-cord vibration. It is sampled into a number of frames centred around equally spaced times.
Feature 10: mean
Feature 11: minimum
Feature 12: maximum
Feature 13: std

Pitch Acc:
Feature 70: min
Feature 71: max
Feature 72: quantile
Feature 73: mean
Feature 74: std
Feature 75: meanAbsoluteSlope
Feature 76: slopeWithOctaveJumps

Pitch CC:
Feature 77: min
Feature 78: max
Feature 79: quantile
Feature 80: mean
Feature 81: std
Feature 82: meanAbsoluteSlope
Feature 83: slopeWithOctaveJumps

Features 84-95: Pitch Point Process:

A PointProcess object represents a point process, which is a sequence of points ti in time, defined on a domain [tmin, tmax]. The index *i* runs from 1 to the number of points. The points are sorted by time.
Feature 84: numberOfPoints
Feature 85: lowIndex
Feature 86: highIndex
Feature 87: nearestIndex
Feature 88: numberOfPeriods
Feature 89: meanPeriod
Feature 90: stdevPeriod
Feature 91: jitterLocal
Feature 92: jitterLocalAbsolute
Feature 93: jitterRap (the relative average pertubation)
Feature 94: jitterPpq5 (the pitch pertubation quotient)
Feature 95: jitterDpd (the differential phase detection)

Features 96-103: Pitch SPINET[4] && **Features 104-112: Pitch Shs**

To perform a pitch analysis based on a spectral compression model. The concept of this model is that each spectral component not only activates those elements of the central pitch processor that are most sensitive to the component's frequency, but also elements that have a lower harmonic relation with this component. Therefore, when a specific element of the central pitch processor is most sensitive at a frequency f0, it receives contributions from spectral components in the signal at integral multiples of f0.

Algorithm used: The spectral compression consists of the summation of a sequence of harmonically compressed spectra. The abscissa of these spectra is compressed by an integral factor, the rank of the compression. The maximum of the resulting sum spectrum is the estimate of the pitch.
Feature 96: min
Feature 97: max
Feature 98: quantile
Feature 99: mean
Feature 100: std
Feature 101: meanAbsoluteslope
Feature 102: slopeWithoutOctaveJumps
Feature 103: linearFit

Feature 104: min
Feature 105: max
Feature 106: quantile
Feature 107: mean
Feature 108: std
Feature 109: meanabsoluteSlope
Feature 110: outerViewport
Feature 111: slopeWithoutOctaveJumps
Feature 112: lowestFrequency
Feature 113: highestFrequency
Feature 114: numberOfFrequencies
Feature 115: frequencyDistance
Feature 116:frequencyInHerz
Feature 117:frequencyInMel

Features 19-23: Intensity statistics

An Intensity object represents an intensity contour at linearly spaced time points ti = t1 + (i - 1) dt, with values in dB SPL, that is dB relative to 2^{10-5} Pascal, which is the normative auditory threshold for a 1000-Hz sine wave.
Feature 19: mean
Feature 20: minimum

[4]SPINET stands for a model of pitch perception called the spatial pitch network.

Feature 21: maximum
Feature 22: range
Feature 23: std

Features 54-69: Ltas Pitch corrected

An object of class Ltas represents the power spectral density as a function of frequency, expressed in dB/Hz relative to 2?10-5 Pa.
Feature 54: lowestFrequency
Feature 55: highestFrquency
Feature 56: numberOfBins
Feature 57: binWidth
Feature 58: binNumberFromFrequency
Feature 59: valueAtfrequcncy
Feature 60: valueInBin
Feature 61: minimum
Feature 62: frequencyOfMinimum
Feature 63: frequencyOfMaximum
Feature 64: maximum
Feature 65: mean
Feature 66: slope
Feature 67: localPeakHeight
Feature 68: localHeight1
Feature 69: std

Features 112-117: Formants && Features 24-29: Formant LPC[5]

An object of type FormantFilter represents an acoustic time-frequency representation of a sound: the power spectral density P(f, t), expressed in dB's. It is sampled into a number of points around equally spaced times ti and frequencies fj (on a linear frequency scale).
Feature 112: lowestFrequency
Feature 113: highestFrequency
Feature 114: numberOfFrequencies
Feature 115: frequencyDistance
Feature 116: frequencyInHerz
Feature 117: frequencyInMel

Feature 24: min
Feature 25: max
Feature 26: quantile
Feature 27: mean
Feature 28: std
Feature 29: numberOfLPCKoefficiencies

[5]LPC stands for linear predictive coding.

Appendix C

Publications

Journal submissions:

- a paper about *TGI+* is ready for submission;
- a paper about tree-automaton high-level features is ready for submission.

A-level (CORE ranking) or impact = 0.62 (CS conference ranking) conference:

- Sidorova, J. Speech emotion recognition with TGI+.2 classifier. EACL-2009 European Association for Computational Linguistics. Student Workshop. Athens, Greece. March, 30 – April, 03. pp. 54-60. [The acceptance rate was 28%.]

Other international conferences:

- Sidorova, J. and Badia, T. ESEDA: tool for enhanced speech emotion detection and analysis. The 4th International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution (AXMEDIS 2008). Florence, November, 17-19. pp. 257–260. IEEE press. 2008. [The acceptance rate was 30%.]
- Sidorova, J., Badia T. Syntactic learning for ESEDA.1, tool for enhanced speech emotion detection and analysis. Internet Technology and Secured Transactions Conference 2009 (ICITST-2009), London, November 9 -12. IEEE press. [The acceptance rate was 55%.]

Book chapter:

- Sidorova J., McDonough J., Badia T. Automatic Recognition of Emotive Voice and Speech. In (Ed.) K. Izdebski. Emotions in The Human Voice, Vol. 3, chapter 12, pp.217-242, Plural Publishing, San Diego, CA, 2008.

Publications in other languages than English:

- Sidorova, J.A. Recognizing emotions from the acoustic signal, man-machine interface perspective. In J. Programming Systems and Instruments. Press of the faculty of Computational Cybernetics and Mathematics, Moscow State University. [In Russian] 2006

Bibliography

[1] Ai H., Litman D.J., Forbes-Riley K., Rotaru M., Tetreaut J., Purandare A. *Using system and user performance features to improve emotion detection in spoken tutoring dialogs.* Proc. INTERSPEECH'2006, Pittsburgh, 2006.

[2] Barra, R., Montero, J.M., Macias-Guarasa, DHaro, L.F., San-Segundo, R., Cordoba, R. *Prosodic and segmental rubrics in emotion identification.* Proc. of the 2005 International Conference on Acoustics, Speech and Signal Processing (ICASSP-2005), Philadelphia, PA, March 2005.

[3] Bloedorn, E., Michalski, R. *Data-driven constructive induction: a methodology and its applications.* IEEE Intelligent Systems, Special issue on Feature Transformation and Subset Selection, pp. 30-37, March/April, 1998.

[4] Boersma P. *Praat, a system for doing phonetics by computer.* Glot International, vol. 5, no 9/10, pp. 341-345, 2001.

[5] Breiman, L., Friedman, T., Olshen, R., Stone, C. *Classification and regression trees* Wadsworth, 1984.

[6] Bunke, H., Sanfeliu, A. (Eds.) *Syntactic and structural pattern recognition: theory and applications.* World Scientific. 1990.

[7] Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W., Weiss, B. *Database of German Emotional Speech.* Proc. Interspeech 2005, ISCA, pp 1517-1520, Lisbon, Portugal, 2005.

[8] Burkhart, F., van Ballegooy, M., Englet, R., Huber, R. *An emotion aware voice portal.* Proc. Electronic Speech Signal Processing ESSP. 2005.

[9] Burkhardt F., Ajmera J., Englert R., Stegmann J., Burleson W. *Detecting anger in automated voice portal dialogs.* Proc. INTERSPEECH'2006, Pittsburgh, 2006

[10] Busso C., Lee S., Narayanan Sh. *Analysis of emotionally salient aspects of fundamental frequency for emotion detection* IEEE Transactions on audio, speech and language processing. Vol. 17, No 4, May 2009.

[11] Cohen, W. *Fast Effective Rule Induction* In Proc. Machine Learning: Proceedings of the Twelfth International Conference, Lake Tahoe, California. 1995.

[12] Cohen S, Rokach L., Maimon O. *Decision-tree instance-space decomposition with grouped gain-ratio* In J. Information Sciences, vol. 177, issue 17, pp. 3592–3612. Elsevier. 2007.

[13] Comon, H., Dauchet M., Gilleron R., Jacquemard, F., Lugiez, D., Löding Ch., Tison S., Tommasi, M. *Tree Automata Techniques and Applications* Retrieved from http://tata.gforge.inria.fr/ on Nov., 20.

[14] Devillers, L., Vidrascu, V. *Real-life emotion detection with lexical and paralinguistic cues on Human-Human call cetre dialogs.* Proc. INTERSPEECH'2006, Pittsburgh, 2006.

[15] Duda R.O., Hart P.E., Stork D.G. *Pattern Classification.* John Wiley & Sons, Inc. USA, Canada. 2001.

[16] Engberg, I.S., Hansen, A. V. *Documentaion of the Danish Emotional Speech Database DES.* Aalborg, Denmark, 1996.

[17] Eyben, F., Wöllmer, M., Schuller, B. *Speech and Music Interpretation by Large-Space Extraction*, INTERSPEECH'2009, Brighton, UK, 2009.

[18] Hall, M. A. *Correlation-based Feature Subset Selection for Machine Learning.* Hamilton, New Zealand. 1998.

[19] Hastie, T., Tibshirani, R., Friedman J. *The elements of statistical learning; Data Mining, Inference and Prediction.* Chapter 11 for neural networks and Chapter 12 for support vector machines. Springer. 2001.

[20] Hebb, D.O. *The organization of behaviour.* NY: Wiley. 1949.

[21] Hozjan V, Kai Z. *Context-independent multilingual emotional recognition from speech signals.* Int. journal of speech technology 6, 311-320. Kluwer Academic Publishers. 2003.

[22] Hozjan V. and Kai Z. *Improved Emotion recognition with Large Set of Statistical Features*, Eurospeech, 2003.

[23] Hozjan V., Kai Z., Moreno A., Bonafonte A., Nogueiras A. *Interface databases: design and collection of a multilingual emotional speech database.* Proc. LREC 2002, Grand Canaria: European Language Resources Association, 2002, vol. 6, pp. 2024-2028. 2002.

[24] Hunter, G.M., Steiglitz, K. *Operations on images using quad trees.* IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 1, No 2, pp. 145-153, 1979.

[25] Jain, A.K., Duin, R., Mao, J. *Statistical Pattern Recognition: A Review.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 1, January 2000.

[26] Jiang X., Tian L., Han M. *Separability and recognition of emotion states in multilingual speech.* 2005 International Conference on Communications, circuits and Systems Proceedings, HongKong, China, vol II: 861-864, May 2005.

[27] Goldberg D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, Inc., 1989.

[28] Kanda T., Iwase K., Shiomi M, Ishiguro H. *A tension-moderating mechanism for promoting speech-based human-robot interaction.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005), Edmonton, Alberta, Canada, August 2-6, 2005.

[29] Kumar R., Rose C. P., Litman D.J. *Identification of confusion and surprise in spoken dialog using prosodic features.* Proc. INTERSPEECH'2006, Pittsburgh, 2006

[30] Kuncheva L. I. *Combining pattern classifiers: methods and algorithms.* John Wiley & Sons, Inc. USA, Canada. 2004.

[31] Li Y. and Zhao Y. *Recognizing emotions in speech using short-term and long-term features.* Proc. of the international conference on speech and language processing. pp. 2255-2258, 1998.

[32] Liscombe J., Riccardi G., Hakkani-Tuer. *Using context to improve emotion detection in spoken dialog systems.* Proc. Interspeech 2005, ISCA, pp 1517-1520, Lisbon, Portugal, 2005.

[33] Liu, H., Motoda H. Section 6.2. for Feature Construction. *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers. Norwell, MA, USA. 1998.

[34] Lopez, D., Sempere J.M., Garcia P. *Error-correcting analysis for Tree Languages.* Int. J. on Pattern Recognition and Artificial Intelligence (IJPRAI). Vol 14, No 3, pp. 357-368. 2003.

[35] Lopez D., Espana, S. *Error-correcting tree-language inference.* Pattern Recognition Letters 23, pp. 1-12. 2002

[36] Lugger, M., Yang B., W. Wokurek. *Robust estimation of voice quality parameters under real world disturbances.* Proc. of the 2006 International Conference on Acoustics, Speech and Signal Processing (ICASSP-2006), Toulouse, 2006

[37] Michaelis, D. Gramms T., Strube H. W. *Glottal-to-Noise Excitation Ratio - a New Measure for Describing Pathological Voices* J. ACUSTICA Acta Acustica, Vol. 83, pp. 700-706. 1997.

[38] Montero J. M., Gutierrez-Arriola J., Cordoba R., Enriquez E., Pardo J. M. *Spanish emotional speech: towards concatenative synthesis* COST 258, 1998.

[39] Moreno-Torres, I. , Berthier M. *The study for the FAS corpus.* To be submitted.

[40] Moriyama T., Ozawa O. *Emotion recognition and synthesis system on speech* IEEE ICMCS 99, 1999.

[41] Neiberg, D., Elenius K., Laskowski K. *Emotion recognition in spontaneous speech using GMM.* Proc. INTERSPEECH'2006, Pittsburgh, 2006

[42] Noam, A., Kerret, O. , Karlinski, D. *Classifying emotions in speech: A comparison of methods.* In Proceedings of Eurospeech 2001, Scandinavia, pp. 127-130. 2001.

[43] Nogueiras A., Moreno A., Bonafonte A., Marino B. *Speech emotion recognition using Hidden Markov models.* In Proc. Eurospeech 2001.

[44] Oommen B. J., Loke R.K.S. *On the Pattern Recognition of Noisy Subsequence Trees.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 9, pp. 929-946, September, 2001.

[45] Oudeyer, P.-Y. *Novel useful features and algorithms for the recognition of emotions in human speech.* Proceedings of the 1st Int. Conf. on Speech Prosody. April 2002, Aix-en-Provence, France.

[46] Petrushin, V. A. *Emotion recognition in speech signal: experimental study, development, and application.* Proc. of the 6th International Conferences on Spoken Language Processing (ICSLP 2000). 2000.

[47] Piramuthu, S., Sikora R. T. *Iterative feature construction for improving inductive learning algorithms.* In Journal of Expert Systems with Applications. Volume 36 , Issue 2 (March 2009), Pp. 3401-3406, 2009

[48] Quinlan, J.R. *C4.5: Programs For Machine Learning.* Morgan Kaufmann, Los Altos. 1993.

[49] Rahman, M.M., Bhattacharya, P., Desai B.C. P. 318 for the definition of the early-fusion of features. *Similarity searching in image retrieval with statistical distance measures and supervised learning.*, In Pattern Recognition and Data Mining. Third International Conference on Advances in Pattern Recognition. ICAPR-2005. Bath, UK, August 2005. Proceedings Part 1. LNCS 3686. 2005.

[50] Riedmiller, M., Braun, H. *A direct adaptive method for faster Backpropagation learning: The RPROP algorithm.* In Proceedings of the IEEE International Conference on Neural Networks, pp. 586-591, 1993.

[51] Sakakibara, Y. *Recent advances of grammatical inference.* Theoretical Computer Science 185, pp. 15-45. Elsevier. 1997.

[52] Scherer, K. *A cross-cultural investigation of emotion inferences from voice and speech: implications for speech technology.* ICSLP 2000, Beijing, China, Oct. 2000.

[53] Schiel, F., Steininger, S, Tork U. *The SmartKom Multimodal Corpus at BAS.* Proc. of the 3rd Language Resources and Evaluation Conference (LREC) Las Palmas, Gran Canaria, Spain. 2002.

[54] Schuller B., Reiter S., Mueller R., Al-Hames M., Lang M., Rigoll G. *Speaker-independent speech emotion recognition by ensemble classification.* Proc. of the IEEE International Conference on Multimedia Expo (ICME-2005), Amsterdam, Netherlands, 2005.

[55] Schuller, B. Reiter, S. Rigoll, G. *Evolutionary Feature Generation in Speech Emotion Recognition.* In Proc. IEEE International Conference on Multimedia and Expo, 2006.

[56] Schuller, B., Rigoll, G., Lang, M. *Hidden Markov Model-Based speech Emotion Recognition.* Proc. ICASSP 2003, Vol. II, pp. 1-4, hong Kong, China, 2003.

[57] Schuller B., Steidl S., Batliner A. *The INTERSPEECH 2009 Emotion Challenge.* Proc of INTERSPEECH'2009. Brighton. UK. 2009.

[58] Sempere J. M., Lopez D. *Learning decision trees and tree automata for a syntactic pattern recognition task.* Pattern Recognition and Image Analysis. Lecture notes in CS. Berlin. Volume 2652. pp. 943-950, 2003.

[59] Sidorova, J. Badia, T. *ESEDA: tool for enhanced speech emotion detection and analysis.* The 4th International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution (AXMEDIS 2008). Florence, November, 17-19. pp. 257–260. IEEE press. 2008.

[60] Sidorova J., McDonough J., Badia T. *Automatic Recognition of Emotive Voice and Speech.* In (Ed.) K. Izdebski. Emotions in The Human Voice, Vol. 3, chapter 12, pp.217-242, Plural Publishing, San Diego, CA, 2008.

[61] Sidorova, J. *Speech emotion recognition with TGI+.2 classifier.* Proc. EACL-2009 European Association for Computational Linguistics, Student Workshop. Athens, Greece. March, 30 – April, 03. pp. 54-60. 2009.

[62] Sidorova, J. Badia, T. Syntactic learning for *ESEDA.1*, a tool for enhanced speech emotion detection and analysis. The 4th Int. Conf. for Internet Technology and Secured Transactions. ICITST-2009. London, November 9-12. IEEE press. 2009.

[63] Smyth E. Kosslyn S. *Cognitive Psychology: Mind and Brain.* Prentice Hall. 2007.

[64] Steidl St., Levit M., Batliner, A., Noeth, E. Niemann H. *Of all the things the measure is man Automatic classification of emotions and inter-labeler consistency.* Proc. of the 2005 International Conference on Acoustics, Speech and Signal Processing (ICASSP-2005), Philadelphia, PA, March 2005.

[65] Stefan, St. *Automatic Classification of Emotion-Related User States in Spontaneous Children's Speech*, Logos Verlag, 2009.

[66] Tosa, N., Nakatsu, R. *Life-like communication agent.* Proceedings of the IEEE International Conference on Multimedia Computing and Systems, pp. 12-19. 1996.

[67] Vapnik, V. *The nature of statistical learning theory*, Springer, 1999.

[68] Vogt, T. Andre, E. *Comparing feature sets for acted and spontaneous speech in view of automatic emotion recognition.* Proc. of the 2005 IEEE International Conference on Multimedia Expo (ICME-2005), Amsterdam, Netherlands, 2005.

[69] Wagner R.A., Fischer M. J. *The String-to-String Correction Problem.* Journal of Association for Computing Machinery. Vol. 21, No 1, pp. 168 – 173, January 1974.

[70] Wilting J., Krahmer E., Swerts M. *Real vs acted emotional speech.* Proc. INTERSPEECH'2006, Pittsburgh, 2006

[71] Witten I.H., Frank E. Sec. 7.1 (for feature selection) and Sec. 10.4 (for multilayer perceptron) in *Data Mining. Practical Machine Learning Tools and Techniques.* 2nd edition, San Francisco, Elsevier. 2005.

[72] Wolpert, D. H. *The lack of a priori distinctions between learning algorithms.* Neural Computation, 8(7), 13411390. 1996.

[73] Zhang, K., Shasha, D. *Simple Fast Algorithms for the editing distance between trees and related problems.* SIAM J. Computation. Vol. 18, No 6., 1245-1262. 1989.

[74] Zhang, K., Statman, R., Shasha D. *On the editing distance between unordered labeled trees.* Information Processing Letters 42, pp. 133-139. 1992.

[75] Zhang, K., Jiang, T. *Some MAX SNP-hard results concerning unordered labeled trees.* Information Processing Letters, 49, pp. 249-254. 1994.