University of Ljubljana
Faculty of electrical engineering

Marc Escalona Mena

EMOTION RECOGNITION FROM SPEECH SIGNALS

ERASMUS EXCHANGE PROJECT WORK

Ljubljana, March 2012

UNIVERSITY OF LJUBLJANA
Faculty of Electrical Engineering

Marc Escalona Mena

EMOTION RECOGNITION FROM SPEECH SIGNALS

ERASMUS EXCHANGE PROJECT WORK

Supervisor: Assoc. prof. Dr. Andrej Košir

Ljubljana, 2012

Acknowledgements

First of all, I am very thankful to my supervisor, Andrej Košir, whose guidance, suggestions and support from the initial to the final stage of the project enabled me to develop this thesis. Even granted me access to one of the LDOS laboratories so that I could develop my work inside the faculty.

I am also indebted to many of my university colleagues for the support and help during all these years, both the ones with who I have been through all the years of my degree, and the new colleagues that I met during my stay in Ljubljana. Without my colleagues' and friend's encouragement, I would not have finished the degree.

Finally, I would like to thank my family for supporting and encouraging me to pursue this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The two main objectives of this project are to analyse the efficiency of several techniques widely used among the field of emotion recognition through spoken audio signals, and, secondly, obtain empirical data that proves that it is actually plausible to do so with a more than acceptable performance rate.

For that purpose, our research will follow four main stages. First, we will board the theoretical approach behind affective states as well as their classification. After that, we will introduce the relationship between spoken signals and their intrinsic acoustic features, through which it is possible to decode information about emotion states. The second stage will cover the presentation of our available materials and chosen methodology. To do so, a description of our source of speech files – emotion database – , as well as a thorough analysis of the most widely used acoustic features meant to decode the hidden information about emotions will be covered. The final part of this stage will treat several feature selection and classification techniques, as well as the means through which we will be presented with the final data.

The third stage, we will cover the presentation of the selected tools that will make possible the whole experimentation part: the feature extraction tool, and the data-mining software that will deliver the final classified data. The fourth and final stage will contain all our experimental work and the final conclusions. Here, we will work with different sets of acoustic features, we will classify them under different nature classifiers and analyse the results. Even more, we will apply several feature selection techniques to these extracted attribute sets, classify them and compare their performance. Each experiment will have its own conclusions, but there will also be one last chapter of general discussion that will take into account all the experimentation.

CHAPTER 1: Introduction

1.1. Motivation

The exploration of how we, as human beings, react to the world and interact with it and with each other remains to be one of the greatest scientific challenges. Our innate abilities such as perception, learning processes, and our capabilities to adapt to the world around us are commonly labelled as part of our “intelligent” behaviour. But what does it mean to be intelligent? During the last decades, there has been a growing research in the fields of neuroscience, psychology, and cognitive science which argue that our common view of intelligence is too narrow; they state that a crucial range of abilities that matter immensely towards our success in life has been ignored for too much time. This group of abilities conform the so-called *emotional intelligence,* and it encloses one's ability to have, express, and recognize affective states; as well as the ability to regulate them, employ them for constructive purpose, and skilfully handle the affective arousal of others. The skills of emotional intelligence have been argued to be a better predictor than other indicators – such as IQ – for measuring aspects like success in life, especially in interpersonal communication, and learning and adapting to what is important.

Needless to say, this research field is, by itself, of great value and complexity. But what if we wanted to aim towards an even wider range of possibilities, going far beyond human-human interaction and setting the next frontier into improving Human-Computer/machine Interaction (HCI) using this newly acquired knowledge? A better understanding of the essence of emotional intelligence would have a positive impact into making HCI more human-like, more effective and even more efficient. One can easily imagine situations where the man-machine interaction could be improved by having machines capable of adapting to their user's affective state. Plausible positive consequences could be a more natural, trustworthy and even more persuasive than before [1]. These findings, together with recent advances in sensing, tracking, analysing, and animating human non-verbal communicative signals, have produced an increase of interest in the *affective computing* field within the HCI domain.

Moreover, a wide list of potential commercial applications of automatic human affect recognition has also contributed to keep moving the wheel of such theoretical research.

Among these potential applications, we may find affect-sensitive systems for customer services, call-centers, intelligent auto-mobile systems, as well as the game and entertainment industries as a whole. Further research and enhancement of such systems would change the ways in which we interact with computer systems [2]. Another important application of automated systems lies within the affect-related research – in psychology, psychiatry, or neuroscience to state a few –, where such systems could improve the reliability of measurements and even speed up the currently tedious manual task of processing data on human affective behaviour. For instance, research areas like mother-infant interaction, tutoring, psychiatric disorders, or even research on social and emotional development would certainly reap substantial benefits from such automatic tools.

Because of both this theoretical interest and practical importance, automatic human affect analysis meant to improve HCI has been attracting the interest of many researchers in the last three decades. An early attempt consisting of vocal emotion analysis was performed by C.Williams and K.Stevens in 1972 [3], and it wouldn't be that long when another study focusing on the analysis of facial expressions appeared : M. Suwa, 1978 [4]. It wouldn’t be until the 1990s when an increased number of efforts towards this research field was experienced. Among them, the studies of H.Kobayashi, F.Dellaert and L. Chen [5] between 1991 and 1998 are solid proof of the mentioned tendency. However, while the 1990s represented the advance towards achieving audiovisual affect recognition, the vast majority of data sets (audio, audio-video) of affective displays – which conformed the basis of all experiments and research – were not only small, but also were recorded under highly constrained conditions. By that we mean that all data contained only deliberate affective displays: all sets were performed by actors (no spontaneous recording), under ideal conditions of noise-polluted environments, and under very specifically and controlled timed windows. Furthermore, there was still no 3D data, or even non combined face and body displays of affective behaviour.

In a certain way, technology advances had not coped up with such research field demands yet. Thus, the decade of the 2000s aimed to palliate these issues by progressively incorporating technology improvements to the building of better data sets from where to further advance into the field. In a parallel way, advances on the topic made it possible to successfully and accurately detect several basic emotions coming from these mentioned audio or video data sets. However, detecting any expressions of human affective behaviour in less constrained settings still proved to be a very challenging problem. The major problems regarding this are to be attributed to the fact that spontaneous affective behaviour differs in visual appearance, audio profile, and even timing; non controllable parameters that are eliminated with deliberate, acted, affective behaviour data sets. But, undoubtedly, going beyond these issues was the focus of the researchers.

An increasing number of studies regarding the automatic analysis of spontaneously displayed affective behaviour began to emerge, explaining how these kind of facial and vocal expressions could be detected. It was also shown that integrating the information from audio and video simultaneously leaded to an improved performance of affection recognition. Variations of head position, eye and brows movement, clutter, or slightly appreciable face muscles movement proved that by making use of the complementary information of these two channels – audio and video – would improve the reliability of affective behaviour recognition [6]. And this leads us to the actual state of the art, entering the second decade of the XXI century.

1.2. Aim of the thesis

Nowadays, this relatively young field mainly focuses on building computational models that simulate human perception of affective states. An intelligent HCI would be one that has the abilities to automatically - and even in real-time – detect, interpret and react appropriately the user's affective state shown. Apart from that, it would be needed to synthesize (and even animate) all affective expressions in order to give a response to the user. And finally, include all that may be needed in order to design this affect-sensitive HCI. But the core that conforms the basis of this field is none other that all the data that can be extracted from the human interaction process: dialogues. On one hand, all the non-verbal communicative cues that include facial expressions, body movements and physiological reactions. While we human beings are able to detect and interpret those signals with little or no effort, designing and developing an automated HCI system that accomplishes these tasks is rather difficult. On the other hand, all the intrinsic verbal data that can be extracted from the communicative act, which can be, by itself, powerful enough to attain the objectives of this field of research. Both channels conform the required audiovisual data flows needed for reliable affective behaviour detection.

In fact, the main part of this project focuses into analysing several techniques used to automatically detect human affective state from the latter mentioned group of data available from any human-to-human communicative act; that is, all the verbal communicative characteristics. Thus, all non-verbal communicative cues have been discarded. Our main goal will be to correctly recognize emotion states from human spoken audio files that belong to reliable databases and experiment with the data in a similar way through which it would be automatically done by a machine/computer in order to make its automatic affection detection. Needless to say, there are multiple ways to achieve this, but now we will briefly describe our choices in order to give a whole view of our work.

First of all, we will need the human data required in every human-machine interaction. As mentioned above, that is why audio file samples will be more than enough for our interests. These demands will be satisfied by incorporating to our work special audio databases that were meant for the purpose of emotion recognition. Needless to say, operate with proper, well built databases will be critical for our results. A common feature of these databases is that each audio file is intended to simulate a specific emotional state – labelled under anger, boredom, fear, etc. However, and despite one could think in advance, they will differ a lot between each other, but this will be through-fully discussed in latter chapters.

The next step of our work will consist of extracting solid and trustworthy features from the audio files. These is arguably the most vital part of our work, since if we were to follow the machine/computer process into properly interpreting one's affective state, the extracted features would be the basis of its decision making procedure. For this purpose, we will be using openSMILE (Speech&Music Interpretation by Large Space Extraction) software, a fast, real-time, powerful open source audio feature extractor. We will use its functionalities to generate a formatted output file for each audio file input containing all the extracted data. Even more, we will want to build a file made from several emotion types correctly attached, mainly for classification purposes, and this tool will not present added problems. It will automatically append the information belonging to another audio file rightly after the last data. Everything will be discussed in detail in chapter 3.

Once we are in possession of the formatted output files containing the extracted features from various input audio files of one database, it will be the turn of classifying the data. Our choice has been to work with WEKA (Waikato Environment for Knowledge Analysis), an also open source software written in Java that contains a collection of machine learning algorithms for data mining tasks. Undoubtedly, it is a powerful tool that we will use for the processing and classification stage of our experiments. Its compatibility with the openSMILE output generated files will prove to be very useful, avoiding unnecessary data formatting adaptation. Once everything is correctly set up and ran, WEKA will deliver a summary log containing the results of our desired classifications. To strengthen the reliability of these results, a ten-fold cross validation technique will always be applied. The proven advantages of such technique when used under the goal of prediction – keep in mind that, in a nutshell, we aim to estimate the accuracy and reliability of a feature extractor – will cope well with our research. Another key element that we would like to briefly introduce is the confusion matrix. Also provided by WEKA output data log, this matrix will contain valuable information regarding the classifier performance, and will be one of the basis from where to deduce the proper conclusions.

CHAPTER 2: Problem statement

In this chapter we will discuss the theoretical background that has shaped the basis of this research field and, at the same time, has established the road path for its further developing. The first matter to attend will be to present and define the way to properly recognize emotions; for that, the starting point will be to define the very own word “affection” by taking a psychological approach. This will be necessary in order to understand the association between such rather abstract term with the data flows and audio signals that will be originated from it. After that, we will focus on describing the most commonly used multi-dimensional spaces to classify affections. These spaces provide a more tangible and graphical view of such abstract terms, and thought they may not be as accurate as they should, they are a valid form of classification.

The next step will be to present the main characteristics of speech databases, and to justify our choice. Afterwards, we will set up the link between spoken signals and feature vectors – filled with extracted speech features – through which we will be able to recognize emotions. That is right, key features need to be extracted from those audio signals in order to build a solid and reliable base that contains precious data about affective behaviour recognition. An overview of the major approaches will be the focus of the next section. Finally, we will comment the major challenges that the research field of automatic affection recognition is facing.

2.1. The description of affect

The description of affect has been a long standing problem in the area of psychology. Three major approaches can be distinguished: categorical, dimensional and appraisal-based. According to the categorical approach, there exist a small number of emotions that have proven to be recognized universally, no matter cultural or historical issues. P. Ekman and his research team conducted several cross-cultural experiments, and concluded in the early 80s that six basic emotions belonged to this hard-wired attachment in our brain: happiness, sadness, surprise, fear, anger and disgust [7]. The influence of such theory resulted in the fact that most of the existing studies of automatic affect recognition focus on recognizing these basic emotions. An intuitive categorization of emotions and the fact of being universal are undoubtedly its main advantages. However, it fails to represent the wide range of emotions that occur in natural, spontaneous communication situations: complex affective states may be too difficult to handle by this approach, thus it only covers a small part of our daily emotional displays.

As for the dimensional approach, it stands on the fact that affective states are not independent from one another; they are related in a systematic manner. Thus, an affective state is characterized in terms of a small number of latent dimensions rather than a small number of emotion categories. In particular, a specific 3D space and a 2D projected spaces have been widely used for researchers in terms of classifying emotions. This approach is also widely used on research and classification, and there is psychological evidence that suggest that these dimensions are intrinsically intercorrelated. However, due to the projection of such high-dimensional emotion states into these two or three dimensional spaces, some loss of information is experimented. In particular, some emotions may either lie outside the space grid or become indistinguishable from one another. But this will be further discussed in next section.

Finally, we get to the appraisal-based approach, an extension of the dimensional approach described above. In this representation, each emotion is claimed to be generated through a continuous subjective evaluation of both our own internal state and the specific state of the outside world. Thus, emotions are characterised by changes in components such cognition, pleasantness, motivation, physiological reactions or even goal-based significance. The main advantage of this model is that it is focused on the variability of different emotional states produced by different types of appraisal patters, instead of limiting emotional states to a fixed number of discrete categories or a few dimensions. Despite being the most complete scheme, using it for automatic emotion recognition still remains too challenging for researchers.

2.2. Emotion spaces

2.2.1. Valence-Arousal: 2D emotion space

In 1980, James A. Russell introduced a circular configuration called *Circumflex of Affect* and proposed that each basic emotion represents a bipolar entity being a part of the same emotional continuum [8]. The proposed polars were arousal and valence, conforming the two axis of the plane. The valence dimension refers to how positive or negative the emotion is, ranging from unpleasant to pleasant feelings; the arousal dimension reflects the degree of excitation that every emotion produces on us, and ranges from sleepiness/boredom to full excitement. Furthermore, this space consists of four distinct quadrants: low and high arousal positive and low and high arousal negative. As argued by Russell, in this way it was possible to characterize all emotions by their valence and arousal, and different emotional labels could be plotted at various positions on this two-dimensional plane.

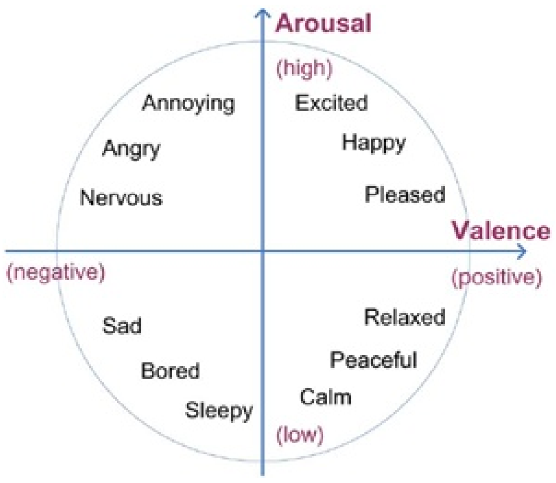


Figure 2.1: Valence-Arousal plane and emotion distribution. Source: http://www.culturehacking.fm/2011/03/identifying-emotion-in-music/

2.2.2. Valence-Arousal-Dominance (VAD): 3D emotion space

This sub-section will cover an specific dimensional space widely used by researchers, a three dimensional space proved to cover the majority of affect variability and whose axis are valence, arousal and dominance (or potency). As described above, the valence (or pleasure) dimension refers to how positive or negative the emotion is, while the arousal dimension refers to how excited or apathetic the emotion is. The other dimension, dominance, refers to the degree of power, or sense of control, over the specific emotion.

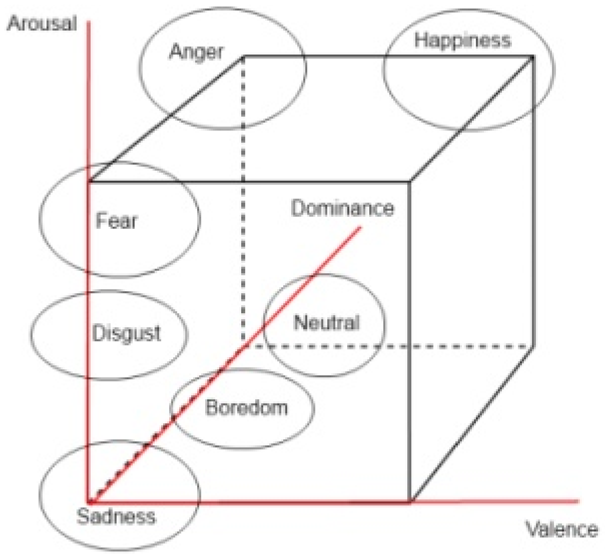


Figure 2.2: Emotions presented in a 3D space – illustrative graph

Despite having such advantages, dimensional approach has also received a number of criticisms. Firstly, the reduction of a former high-dimensional emotion space to two or three dimensions is extreme and results in loss of information. Secondly, while some basic emotions, seem to fit well and in quite different sub-space regions of the dimensional space, some other basic emotions may become indistinguishable, overlapped (e.g., fear and anger), or even some may lie outside the space (e.g., surprise). We may get a glimpse of such undesirable effect while seeing figure 2.1, and specifically all the labelled emotions of one same quadrant. The third dimension of the VAD space is mainly intended to diminish this effect of overlapping, considering that information about potency (dominance) can be vital to differentiate emotions that appeared indistinguishable under the two-dimensional space. Thirdly, it still remains unclear how to determine the position of other affect-related states such as confusion.

2.3. Speech databases

In the following lines we will present a general view of the characteristics of such databases, and then specifically focus on the ones we have been able to experiment on. The major distinguishable parameter refers to each database authenticity; there seems to be two clearly types of databases used within the emotion recognition research field.

Firstly, databases made of acted emotions. These are built by asking actors to speak with a predefined emotion, and then each sample is manually labelled under its specific emotion. Aside from being built under a highly constrained and probably little realistic environment, speech data that has been collected that way, we will see that it has some advantages as well. As each utterance is isolated from the interaction context, this strategy is fond in using correlations between the paralinguistic displays and the linguistic content: a characteristic that plays an important role for affect recognition. The second type of databases are the ones built from real-life systems (for example, call-centers, interviews, or meetings), thus containing authentic emotional speech. As well as the first type, each data set is then manually labelled under the corresponding emotion. The optimal choice for our study would seem to be quite obvious, but let us discuss several issues that prevent us from doing so.

Many efforts have been done towards the construction of databases of spontaneous human affective behaviour, but these type of databases imply a real challenge when it comes to labelling each data sample: the decision is not quite as simple as with the first type of databases, where each emotion was a pre-set variable. And even more of a challenge when we move beyond the six basic emotions mentioned at section 2.1. In other words, data labelling becomes subjective, with an specially high degree of subjectivity when dealing with only audio data. This is because we lack the video data channel that would undoubtedly contain valuable information towards the decision making procedure: eyebrow movement, head position, and other facial expressions that are strongly linked with speech. In fact, objectively coding vocal behaviour still remains an open issue.

Furthermore, the majority of these datasets are not publicly available. Some are still under construction, some are in the process of data publication, and some seem to lack the agreement of subjects that performed the database to make in publicly available. More specifically, spontaneous displays of emotions, especially in audiovisual format but also present in spoken data, reveal personal experience; privacy issues jeopardize the public accessibility of many databases. Ethical problems with publishing the content of call-centers, interviews or meetings need to be dealt with.

Aside from that, context and cultural issues have to be specially taken into account in this type of database: where were the recordings made, the presence of other people, used stimuli, environment, etc. We have to keep in mind the example of the call-center belonging to this category of databases. Contrary to the first type, here there are even more context variables that may influence the masking of emotional reactions. For instance, as we will see later on, the German database (acted database) audio samples were recorded inside an anechoic chamber. Thus, most of the data of such databases currently lacks reliable labelling, which is of crucial importance for engineers responsible to construct automated human affect recognition.

Even constituting a bigger problem is the context dependency emotional recognition. It is a fact that our automatic interpretation of human behavioural signals is context dependent. For instance, a smile can reflect joy, politeness, or even irony. And there are thousands of such signs that need to be taken into account. But as a matter of fact, to properly interpret such signs, it is key to know the context under which this signal was displayed. We are talking about environment factors like where and who the expresser is, what his current task is, who the receiver is, their relationship, etc. though it may seem that such signals should only be taken into account when experimenting with audio-visual databases, context dependency signals may also reflect into speech parameters. Following the same example, subtle changes of tone during a laughter may indicate satire or irony instead of straightforward joy or gratefulness, and would undoubtedly reflect in our spoken signal. This area of research will remain out of our scope, since the used databases assure us that such events do not constitute a parameter of the audio samples. In other words, the utterances do not contain context dependent signals.

This is why despite the recently strong objections emerged from using acted databases, we chose to work with them. In fact, they still serve some purposes, and are very useful when theoretical research is done rather than construction a real-life application for the industry. But even if the choice is already made and we presented solid arguments that support it, it would be unwise to deny that this kind of data, and the emotion classes modelled by it, cannot simply be transferred into realistic data. They are part of theoretical models built under high constrained environments.

So, after the core challenges regarding speech databases have been discussed, we just had to gain access to publicly available acted databases, which unfortunately were a very few number among the total available.

2.4. Affection recognition from speech

Speech is an important communicative modality in human-to-human interaction. Vocal signals contain affective information through explicit (linguistic) and implicit (paralinguistic) messages that reflect the way that the words are spoken. As the linguistic content is concerned, some information about the speaker's affective state can be inferred directly from the surface features of words, while the rest of the affective information lies below the text surface and can only be detected using the semantic context.

Most of the existing efforts from researchers aim at the recognition of a subset of basic emotions from speech signals, while also trying to find dependent and more complex affective states such as confusion, frustration or confidence among others. In addition, few efforts have been directed towards the automatic recognition of non-linguistic vocalizations also present within our speeches such as laughters, coughs, yawns or even cries. This is of particular importance because recent studies in cognitive science showed that us, human listeners, seem to be rather accurate in decoding some non-basic affective states – like distress, anxiety or sexual interest – from these non-linguistic vocalizations [9].

Regarding the existing approaches to vocal affect recognition, the trending way seems to be to use acoustic features as classification input based on the acoustic correlation. The most common features are prosodic features (e.g., pitch-related feature, energy-related features, and speech rate) and spectral features (e.g., MFCC and cepstral features) [10]. However, a few years ago emerged a successful method of obtaining systematically generated static feature vectors using either the so called acoustic Low-Level Descriptors (LLD) or descriptive statistical functionals. A description of them will be covered in the next chapter. The growing number and presence of LLD and functionals in this research field has eased the extraction of large feature vectors via brute-force methods, even achieving many thousands of extracted features. It will undoubtedly be our case scenario. Our feature extraction tool, openSMILE, will easily surpass – or at least be within – the thousand features barrier. Such brute-force extracting method of acoustic parameters does, however, require another step that was not previously mandatory. That is to implement a feature selection method once obtained the large feature vector. This procedure will be covered in latter sections, but its main goals are to reduce computational costs and spacial dimensionality by carefully eliminating uninformative or redundant features.

Still, the automatic extraction of such related features presents several issues. One challenge in audio expression analysis is how we can identify affect-related features in speech signals. When aiming to detect spontaneous emotion expressions, we have to take into account both linguistic and paralinguistic cues that mix together in the audio channel. Although there is a set number of linguistic and paralinguistic features proposed in the core of the field's literature, the optimal features set has not yet been established. Thus, maybe the implemented and somehow universally used set of LLD and functionals could be further improved. In fact, existing automatic speech recognition (ASR) systems cannot reliably recognize the verbal content of emotional speech . Second, extracting semantic and linguistic features of discourse information is even more challenging. Note that these part of features are also represented by several low-level descriptors (LLD), but its discussion was skipped due to the lack of solid and reliable background on the subject.

Another challenge is how can we reliably extract these features from the audio signals under the constrain of real-time behaviour. Even more important if we are using these recently emerged LLD and functionals that automatically extract these features under brute-force techniques and result in thousands of features vectors. Indeed, we should, at least ideally, expect from this software kit to be able to do it in real-time for future enhanced HCI systems, yet with this feature extraction approach we are obligated to add a feature selection post-processing. This leads to a dropping recognition rate of the existing Automatic Speech Recognition (ASR), specially if complex emotional speech need to be processed.

Aside from the discussed specific issues related with the present methods of feature extraction, there are also a number of challenges that deeply affect the research field as a whole. Those issues are the ones that really prevent quicker advance on the topic, and are mainly related with lack of universal and solid standardised methodology. For instance, comparability between research results in the field is considerably low. The majority of them tend to adapt different evaluation strategies, making a high diversity of corpora available – which not always is a good sign. Additionally, there is practically no same feature set found twice: high diversity is not only found in the selection of low-level descriptors (LLD), but also in the perceptual adaptation, speaker adaptation, and – most of all – selection and implementation of functionals. This phenomena is rather paradoxical, since there certainly is a more or less settled and clearly defined group of features, like MFCC, pitch modelling or PLP coefficients that allow for higher comparability in speech recognition. Even more, different types of feature selection techniques are used with low documentation on parameter configuration. To get a general idea, it seems like everyone is fighting their own battles without paying excessive care, not only on joining forces towards standardization and regulation, but even not paying attention on what other researchers are focused on.

CHAPTER 3: Materials and methods

3.1. Materials

We have one database available for the present study, and it will be described in this section.

3.1.1. Berlin database of Emotional Speech (EMO-DB)

As a part of a funded research project within the Technical University of Berlin, a reliable and trustworthy German database of more than 500 emotional utterances was built between 1997 and 1999 [11]. The database was composed by 10 different actors – 5 males and 5 females – ranging from the age of 25 to 35 years old, who were told to read portions of 10 different texts while simulating seven different emotions. The chosen emotions were anger, boredom, disgust, anxiety/fear, happiness, sadness and finally a neutral version. Everything was properly recorded inside an anechoic chamber of the department of Technical Acoustics from the mentioned German university [12].

From within their web-page, one can have access to the whole database, weighting 45 MB and containing 535 wav audio files that can be easily classified under their original emotional state simulation. For instance, under the emotion labelled “anger”, there are 128 files, while under the one labelled “disgust” there are 46 audio files; the remaining emotions are characterized by a number of files that are in between the two above. To properly organize it, a set of seven folders was created – each one labelled as an emotion plus the Neutral state –, and the corresponding audio files from each emotion were put inside it.

Anger	Boredom	Disgust	Fear	Happiness	Neutral	Sadness
128	81	46	68	71	79	62

Table 3.1 : distribution of the 535 audio emotion files

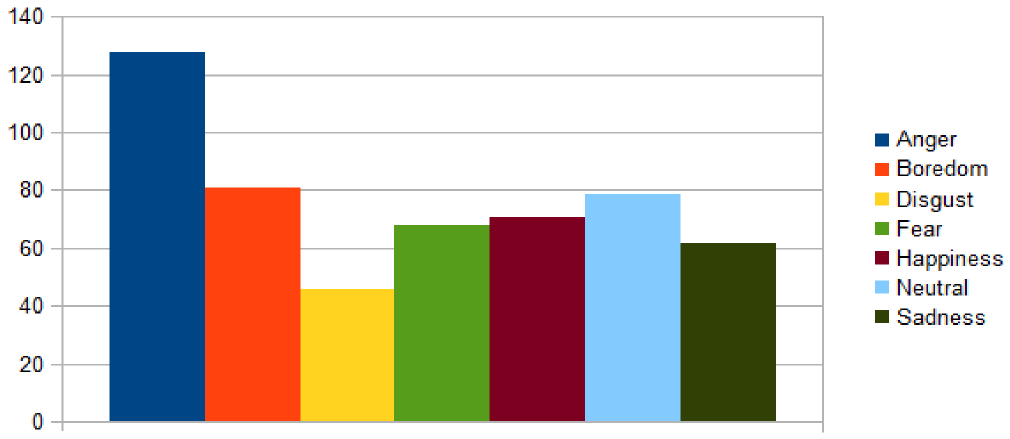


Figure 3.1: Bar-graph showing the number of files per emotion

3.2. Methods

This sub-section covers the methods used for this research project. First, we present how we have achieved interoperability between different software tools, which basically consists on script coding to operate with our extraction tool under an optimal way. Secondly, we proceed with the discussion of the specific methodological steps done for reliably recognize affection. Such steps include: (1) feature extraction from our database audio signals, (2) several feature selection procedures aiming to lower the huge data available for classification, and (3) reliable emotion classification methods. Notice that, in a nutshell, these mentioned steps are the ones that are intended to be part of the core of any automatic affect recognition system applied to improve HCI. What we will be doing is, in a way, simulating and analysing how this machines may operate. However, as we will see, automatic affect recognition is inherently a multidisciplinary research field that involves psychology, linguistics, computer vision, speech analysis and machine learning. Indeed, a very complex field that, almost paradoxically, we humans have the innate ability to successfully reproduce many times during each human-to-human interaction.

3.2.1. Script testing

Our feature extraction software, openSMILE, is only accessible via console command lines and was primarily designed to be handled via Linux OS. Considering the key role of this software for our research, these conditions restrained a lot our possible approaches. We decided to operate at maximum compatibility with it, and our first operations with openSMILE were done via the console command window. But we realized that if we wanted to run this extraction tool several times, computing different audio files, proceeding in a manual way would be a meaningless and time consuming task.

This is why we decided to develop script code to simulate an iterative, controlled and automatic way of running this tool. Four basic parts needed to be taken care of. First, setting a proper generated output file format that we could immediately recognize from all the others: a file name that contained the date, time and a brief reference to the specific experiment that it was referring was our choice of formatting. Second, the ordered access to our speech database files and constant correct handling of our working folders. This is vital since openSMILE needs to know information such as the absolute path of the input file (speech database), the path of the specific configuration file (inside the package folders) and the absolute path and name of the about to be generated (or added information at the end of an already existing one) output file with the extracted features. So, setting the appropriate working folder at any given time was a key condition. Third, building the "main code" of the script file. In here, basic and specific instructions for each experiment have to be written down. For example, the successive iterations while accessing to different audio files, or the iterative feature extraction into a single output file (while controlling the correct format of the output data). And forth, we had to take special care with closing any opened files, fixing compiling and debugging errors and achieving a correct functionality of the code.

3.2.2. Signal feature extraction

As discussed in section 2.3, the common trend of feature extraction consists of extracting key features from speech samples and creating large vectors. In particular, prosodic and spectral features – always included in such vectors – are extracted through the mentioned low-level descriptors – named acoustic LLD –, while a number of operators and functionals are applied afterwards to normalise these vectors over time (it is important to obtain feature vectors of equal size). In the following paragraphs, we will see what are these relevant acoustic features and we will comment on how can they be extracted. Our extraction tool, openSMILE, behaves in a very similar way to what we will describe.

3.2.2.1. Acoustic low-level descriptors

Duration: These features model temporal aspects, providing temporal properties about both voiced and unvoiced segments. Its extracted attributes can be distinguished according to their extraction nature. On one hand, there are those that represent temporal aspects of other acoustic base contours. On the other hand, those that represent the duration of specific phonemes, syllables, words or pauses. In general, different types of normalisation can be done with all of them (mean, averaging, etc.)

Intensity: These features model the loudness (energy) of every sound simulating the way it is perceived by the human ear by calculating the sound amplitude in different intervals. Thus, the extracting technique is based basically on two main characteristics. One, it is known that as the intensity of a stimulus increases, the hearing response grows logarithmically. And two, sound perception also depended on both the spectral distribution and on its duration. Aside from that, loudness features are sequentially extracted on a frame base and put together into a so called loudness contour vector. Finally, energy features contain information of the arousal level of emotions, and are obtained by applying functionals to the loudness contour.

Pitch: The pitch signal, or glottal waveform, has information about emotion because it depends on the tension and vibration of the vocal folds. Two features related to this signal are widely used, namely the *pitch frequency* (F0) and the *glottal velocity volume* . The latter denotes the air velocity through the glottis during the vocal fold vibration. High values of such velocity indicates emotions like happiness, joy or surprise; while low values of it are representatives of affective states such anger or disgust. Regarding F_0, it contains information of the vibration rate of the vocal folds, and is also known as *fundamental frequency of the phonation F_0.* Many algorithms for estimating the pitch signal exist, but we will discuss one in particular since it will be used by our feature extractor software tool, openSMILE.

This algorithm is based on the Autocorrelation of Center-clipped Frames (ACF), designed by Sondhi in 1968 [13]. Basically, the signal is low filtered at 900 Hz and segmented to short-time frames of speech. Then, short-term autocorrelation is calculated. From the resulting autocorrelation frames, F_0 can be computed. Also, the glottal velocity volume is obtained by finding the maximum value of the autocorrelation.

Formants: These features capture spectral information about the position and shape of the formants, which are one of the quantitative characteristics of our vocal tract, the cavity where the sound source is filtered. Lower formants are specially rich of speaker characteristics. Each formant is characterized by its central frequency and its bandwidth, and it has been proven to contain information about emotion. For instance, humans under stress or depression do not articulate voiced sounds in the same way than while being under more neutral emotional states. These changes lead to difference in formants bandwidths. The estimation of formant frequencies and bandwidths is commonly based on using linear predictor analysis, via LPCs (Linear Predictor Coefficients).

Spectrum: The spectrum is characterised by these mentioned formants that model the spoken content. In fact, once the spectral envelope is estimated by using LPC method, further spectral features can be computed. Among them, there are the centroid, flux, roll-off, or even the ratio of spectral flatness. Furthermore, the long term average spectrum can be also obtained, a feature that gives important, general, spectral trends. Besides these specific features, other classical spectral features are also computed. Tools like Fast Fourier Transformation (FFT) easily allow us to get a glimpse over parameters such as phase, magnitude, intensity, or power coefficients in decibel scaling. However, as we will see, several techniques that divide more optimally the spectral space will also be used, always taking into account to exploit our human ear response bands. These are related with the cepstral space and diving it in Mel Frequency Bands (MFB).

Mel-Frequency Cepstrum Coefficients (MFCCs): These coefficients result from a transformation to a cepstrum space, in order to capture information of the time-varying spectral envelope. The cepstrum space is yet another spacial transformation that results from taking the inverse spectral transform of the logarithm of the spectrum. Its basic unit is the quefrency, and its main advantages are to be relatively robust space against noise that also empathizes changes or periodicity in the spectrum. Specific features proven to be useful under practically any speech processing task can be obtained under this space: the so called MFCCs [14].

Low-Frequency Power Coefficients (LFPCs): Similarly to the MFCCs, this coefficients are used to get information about the energy of certain frequency bands. In practice, it is believed that these are better features than MFCCs for emotion classification since they also include pitch signal information. LFPCs are derived by filtering each short-time spectrum with 12 bandpass filters specifically designed to exploit the critical bands of human auditory frequency response.

Perceptual Linear Predictive Coefficients (PLPs): PLP analysis models the auditory perception by making use of the critical-band curve, equal-loudness curve and intensity-loudness power law. PLP coefficients [14] are also used as emotion recognition features, despite being very similar to MFCCs. They both try to smooth a short-term spectrum in order to approximate the human ear system, but it is believed that PLP approach by an autoregressive model and a Bark filter bank is also plausible to take into account.

Wavelets: These are widely used in signal processing field, and particularly in our domain, they give a short-term multi-resolution analysis of time, energy and frequencies of a speech signal. It is believed to be an even superior model of temporal aspects than MFCCs or PLPCs.

Voice Quality Parameters: These features are closely related to phonation and are based on the acoustical model of the vocal folds. They are calculated by first inverse-filtering the speech signal. Features such signal to noise ratio (SNR), harmonic to noise ratio (HNR), or jittering noise can be classified under this group.

Non-linguistic vocalisations: Under this section fall all the features that are the fruit of non-verbal phenomena, such as breathing, laughter, or yawns. For ASR engines to correctly work, the systems have to be also trained to recognize such implicit characteristics present in our daily speech.

Appendix A includes graphical representation of several of these acoustic features, aiming to show them in an illustrative way. For instance, duration, intensity, pitch and some of the frequency domain features will be covered.

3.3.2.2. Functionals

Besides LLD feature extraction, a number of operators and functionals are also applied. This process can be pictured as a cascade procedure that occurs after LLD extraction, and has as main advantages that the total number of features can be scaled (expanded or shrunk) with respect to the number of initial LLD. By using functionals, we are able to obtain one feature vector per word, and even with a constant number of elements, if we wish it to be like this. Consequently, data is formatted and ready to be modelled by a static classifier, as we will see in sub-sections below. Functionals can range from statistical to curve fitting methods. Regarding the first group, statistical functionals, the most popular cover:

First order moments: these include typical measures widely known, such as mean (arithmetic, absolute, root-squared...) or standard deviation, and other probably less known but also greatly used under the field of statistics and probability theory. Among those, the computation of the skewness – or the measure of the asymmetry of the probability distribution function - and kurtosis calculations. The latter measures the “peakedness” of the mentioned function, and describes its shape.

Percentiles: Quantiles are points taken at regular intervals from the cumulative distribution function of a random variable. Specifically, quartiles – a particular quantile – are functionals for speech feature extraction. They are the three points that divide the data set into four equal groups, and their use is highly linked with the computation of percentile rank parameters.

Zero Crossing Rate (ZCR): The Zero Crossing Rate counts how many times the speech signal changes its sign.

Temporal: In here, all the functionals retaining temporal information are gathered. Duration of utterances, number of segments, position of maximum and minimum values

Spectral: the most important functional among the spectral statistical group is probably the computation of Discrete Cosine Transformation (DCT) coefficients, proven to be useful for speech recognition.

As for curve fitting methods, they are mainly the result of regression analysis. These techniques are used for modelling and analysing several variables when the focus is centred under finding relationships between dependant and independent variables. regression coefficients and regression errors. The results of such curve-fitting methods derive into either regression coefficients, that represent the rate of change of one variable as a function of change of another one, and regression errors, computed in order to quantify errors between the regression curves and the original LLD.

Table 3.2, below, is intended to aid to understand the common LLD and functionals used for feature extraction purposes. Deriving, filtering, and chunking processing undergo to, among other purposes, normalise and format the resulting large feature data vector. Notice that the majority of them have already been treated in this sections, specially those that will be used by our feature extraction tool, openSMILE, deeply discussed under section 3.3.

Low-Level Descriptors		**Deriving**	**Filtering**	**Chunking**	**Functionals**		**Deriving**	**Filtering**
Intonation	F0, pitch modelling	Raw LLD, delta regression coeff. , autocorrelation, cross-correlation coeff., Cross-LLD, LDA, PCA...	Smoothing, normalising...	Absolute, relative, syntactic, semantic, emotional	**Extremes**	Min, Max, ranges...	Raw functionals, hierarchical, .Cross-functionals, contextual, LDA, PCA..	Smoothing, normalising...
Intensity	Energy, loudness...				**Mean**	Arithmetic, absolute, square-rooted...		
Linear Prediction	LPCC, PLP...				**Moments**	Std. Deviation, skewness, kurtosis		
Cepstral Coefficients	MFCCs				**Regression**	Coeff.,error...		
Formants	Amplitude, bandwidths, centroids...				**Spectral**	DCT coeff...		
Spectrum	MFB,NMF				**Temporal**	Duration, position....		
FT-Transformation	Wavelets				**Segments**	Number, duration...		
Harmonicity	HNR...				**Percentiles**	Quartiles,ranges		
Perturbation	Jitter,shimmer...				**Peaks**	Number, position...		
Non-linguistics	Laughter, yawns....				**Zero-crossing**	ZCR, Mean-crossing rate...		
Duration	Pauses, syllables , words...				**Onsets**	Number, rel. pos. of first/last on-/offset		

Table 3.2: List of acoustic LLD and functionals widely used for research, and how features are extracted via them [15]

3.2.3. Feature selection

Up to this point, we have described the way to obtain a large set of features from our speech signals. We are ready to begin our experiments, but there is yet another step that, if fulfilled, will gives us a number of improvements. That is, specific selection of features from the original data set. If done right, this procedure will eliminate irrelevant features that might hinder the recognition rates. Consequently, it lowers down both the input dimensionality and the computational time of each experiment, since we will be dealing with a smaller, yet al least equally effective, set of data. It will be even easier to state generalizable conclusions, because the remaining key features should be less correlated.

Using the WEKA environment, the mentioned techniques are defined with a two stepped procedure: 1) choosing a desired attribute evaluation method; 2) picking the attribute selection method. We present this section before boarding the presentation of our chosen classifiers because, although we will not use feature selection techniques for the first experiments, when we decide to use them the logical course of action will be to do it before data classification.

Starting with the first step mentioned above, we will define our two choices. A couple of widely used techniques that attend to this matter are Principal Component Analysis (PCA) [16] and Linear Discriminant Analysis (LDA). The first technique uses an orthogonal transformation to convert a set of correlated variables into a smaller set of uncorrelated ones, referred to as principal components. Among that set, the first principal component is the one with the highest variance possible, while the subsequent components have the highest variance possible under the constraint that they need to be uncorrelated (orthogonal) with its preceding components. The new components are linear combinations of the former ones, and organized from higher to lower variance. PCA technique is implemented in WEKA, and experimenting with it will be part of the subject of our study. On the other hand, LDA is a method that searches for the linear transformations that maximize the ratio of the determinants of each between-class covariance matrix. That way, differences between groups of two-classes are taken into account, and further dimensional reduction can be achieved. However, it is not present in Weka package, so it will be out of our scope.

The other attribute evaluation method that we will use appears listed as *CfsSubsetEval,* and its basic course of action is defined by evaluating the worth of a subset of attributes by considering the individual predictive ability of each feature and contrasting it with its degree of redundancy between them. In other words, it computes the correlation of each class attributes and eliminate those that achieve the greater correlation – since they are marked as highly redundant.

In order for such techniques to work, we need to define another parameter: the attribute selection method, which carries out the second step of feature selection techniques under Weka. This will act like the "agent" who will retrieve only the attributes that fulfil one specific condition from those that have already undergone one of the attribute evaluation methods described above. One condition that will vary depending on the specific selection method chosen.

Among the list, we will definitely take into account the *Best First*, *Linear Forward Selection, Ranker* and *Scatter Search* selection logic. A brief description of these goes as follows:

1) Best First: Searches the space of attribute subsets by greedy hill-climbing techniques [17] that iteratively:
    - evaluate a candidate subset of features (S1),
    - modifies the subset by adding new candidate features (S2)
    - evaluates if the new subset (S2) is an improvement of the old (S1)
    - if so, he will add the new features to his candidate set

   It also has a backtracking facility, which means that it may start with the empty set of attributes and search forward, or start with the full set of attributes and search backward, or even start at any point and search in both directions.

2) Linear Forward Selection*:* Seen as an extension of Best First, LFS takes a restricted number of *k* attributes into account easily specified by the user. The search uses either the initial ordering to select the top *k* attributes, or performs a ranking (with the same evaluator the search uses later on). The search direction can be forward, or floating forward selection (with optional backward search steps). For more information, refer to [18]

3) Scatter Search: Performs an SS [19] through the space of attribute subsets. Starts with a population of many significant and diverse subset, and stops when the results are higher than a given threshold or there is not any improvement. It can be configured to do the search following either a *greedy* (SS-GC) or a *reduced greedy* combination (SS-RGC) between each of the thousands of subsets that it will work with. Aside from performance differences, the main variation between both configurations will be the amount of time necessary to deliver the final results: much greater with the greedy combination.

4) Ranker: Ranks attributes by their individual evaluations and has to be used in conjunction with several attribute evaluators, like PCA. In other words, WEKA only allows the search method of ranker when PCA transformation is applied.

What has been hinted in the ranker search method will decrease our degree of possible combinations, often following a trial and error scheme, while trying to find the best feature selection method for our data sets. However, when using the correlation-redundancy method (*CfsSubsetEval*) we will be able to interact with the four searching methods.

3.2.4. Reliable emotion classification

Similarly to what we are facing when deciding which would be the optimal feature selection algorithm, choosing an appropriate classifier to train our features does not hold a universal answer. Thought not being clear which constitutes the best choice, there are a number of factors that need to be taken into account for achieving successful results. First of all, the classifier's tolerance of high dimensionality. In our case, we are dealing with such data sets, and it will be very reasonable to perform a feature selection technique before the actual classification takes place. At least, as we have described above, it would ease the computational cost, experimental time consumption and diminish CPU memory usage. But the following section choices do not depend on the optional – though recommended – step of feature selection techniques. In fact, we will present an overview of the different classifiers nature available worldwide and then focus on the ones we will use for our research.

Popular classifiers for emotion recognition are Linear Discriminant Classifiers (LDCs) and k-Nearest Neighbour (kNN) classifiers. However, nowadays they are being surpassed by Support Vector Machines (SVM) classifiers, which outmatch the two above. Although SVM may not necessarily be the best classifiers for every constellation, their general optimistic performance has lead them to be nowadays conceived as a sort of state-of-the-art classifier.

Another key factor when determining which classifier to use is the size of the data set being worked on. For smaller data sets, it is believed that non-linear discriminative classifiers like Artificial Neural Networks (ANNs) or decision trees are the best option. However, they are rarely used on both acted and spontaneous emotional database, so we will only experiment with one type of functional tree algorithm that seems to be established as the most reliable classifier: Logistic Model Trees (LMT) [20]. This family of algorithms tries to combine the advantages of two popular methods for classification, which are linear logistic regression and tree induction, in order to attain an overall better performance.

Besides these, there are the so called dynamic classifiers: Hidden Markov Models (HMM), Dynamic Bayesian Networks, or Dynamic Time Warp among others. Specially the first has been widely used under speech recognition, both for acted and spontaneous databases. Its main advantage is that they implicitly warp observed features sequence over time.

Once presented an overall view of the scenario, we will focus on our choices for the research at hand. We have decided to test several classifiers that are included in the Weka package, aiming to compare classifiers of different nature. We are aware that one of the ways to find the most optimal choice is to test the classifiers on a same large and representative database. While this method of trial and error is far from being the optimal, it has become easy and straightforward if ran under this data-mining software. But, taking into account the families described above, we realised that Weka included representatives of almost all of them. Thus, our choices are:

1) SMO , which implements John C.Platt's Sequential Minimal Optimization algorithm [21]. Belongs to the family of SVMs classifiers, and it has proven to have better scaling properties and faster processing than SVMs training algorithms. We will see if it represents a good choice when faced with our type of data.

2) Naïve Bayes, As for the naïve Bayes classifier, it is based on the application of Bayes theorem with strong independent (naïve) assumptions.
3) LMT,or Logistic Model Tree, as the representative of the already discussed functional tree family of classifiers.

3.2.5. Ten-fold cross-validation

Cross-validation is a statistical method of evaluating and comparing learning algorithms by dividing into two segments: one used to train a model, and the other used to validate the model. One particular case of such methodology is k-fold cross-validation. Here, the data is first partitioned into *k* equally sized segments (or folds). Then, *k* iterations of training and validations are performed such that, under each iteration, a different fold of data is used for validation while the remaining *k-1* are being used for training.

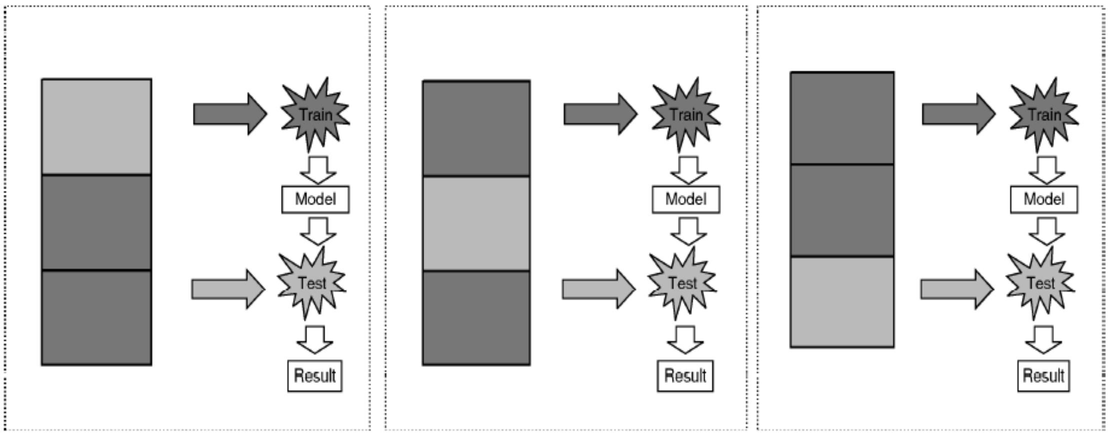


Figure 3.2: Example of 3-fold cross-validation. Source [22]
Training data are the darker sections (2/3 of data)
Test data are the lighter-gray sections (1/3 of data)

For data mining purposes, this approach serves as a standard procedure for performance estimation and model selection. We will be using 10-fold cross-validation technique as a Weka pre-established parameter for performance estimation of our chosen learning/classifier algorithm. The reason to proceed this way is not only due to the fact that it is already a standardised parameter of Weka, but also because it is considered to be one of the best approaches or cross-validation [22]. We will be repeatedly using 90% of the available data (audio extracted features) to build a model and test its accuracy on the remaining 10%. As we will be experimenting with several classifiers, a reliable estimation of each of them performance will be given by the results of such validation.

3.2.6. Summary of our experimental methodology

The following flow-chart illustrates what we have been discussing in section 3.2. Starting with feature extracting methods from our source audio files belonging to speech databases, advancing to the feature selection step that, though we have seen that its implementation will (at least theoretically) only bring advantages, it will not be done in all of our experiments. That is why it is underlined as an “optional step” in the flow-chart. The remaining boxes simulate the 10-fold cross-validation technique that we have been discussed, under the constraint of the chosen classification model, which will vary through our experiments. The sub-section 3.2.6 already covers the most important results from our classification performance. That is why it was not included in this chart; next section, 3.3., covers the tools that we used to make this possible. In other words, we would be putting specific “names” to each one of the rectangular-shaped boxes of this flow-chart. Such names would reflect:

(1) specific feature extraction configuration files available from openSMILE

(2) specific feature selection algorithms from Weka package

(3) specific Weka classifiers

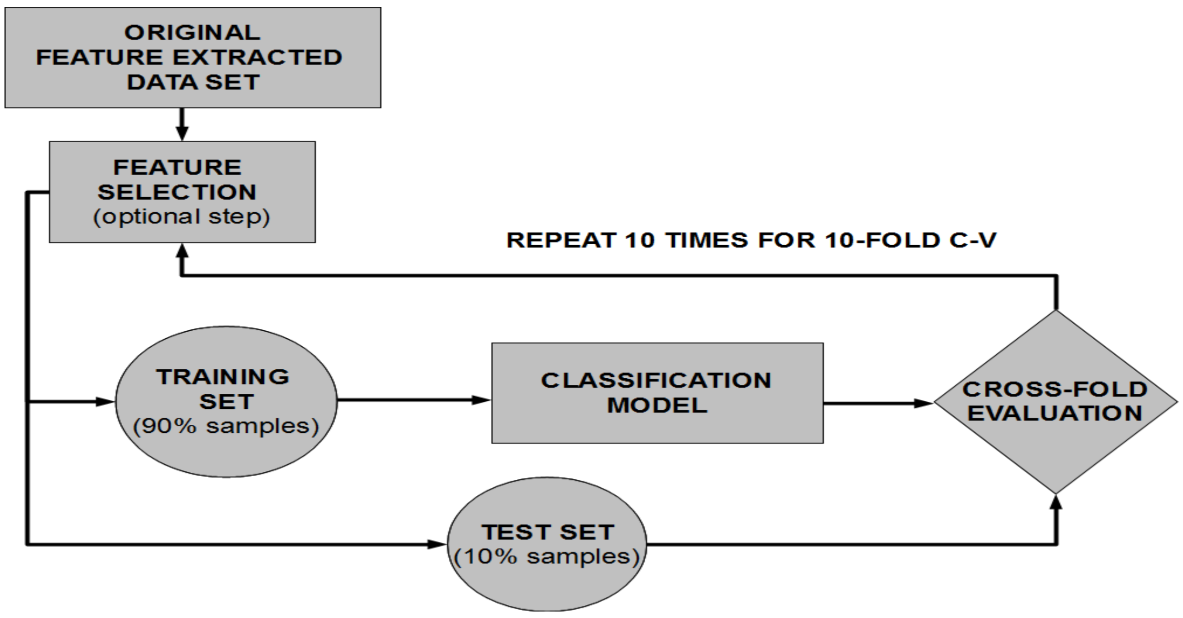

ORIGINAL FEATURE EXTRACTED DATA SET
FEATURE SELECTION (optional step)
REPEAT 10 TIMES FOR 10-FOLD C-V
TRAINING SET (90% samples)
CLASSIFICATION MODEL
CROSS-FOLD EVALUATION
TEST SET (10% samples)

Figure 3.3: Flow-chart of methodology used up to this point

3.2.7. Confusion Matrix

The computation and analysis of confusion matrices will be a key component towards concluding if we undertook a good or bad approach while classifying the data features. Its computation will be part of Weka final results and will be presented to us once the classification has been complete. In [23] we present a graphical document showing a visual explanation of such matrix characteristics.

Confusion matrices are widely used graphical tools that reflects the performance of an algorithm. Each column of the matrix represents the instances of a predicted class, while each row represents the instances of an original class. Thus, it is easy to visualize the classifier's errors while trying to accurately predict each original class instances. In section 4.1.2, data results, we dig into further analysing such matrices with the results obtained with our first experimental conditions. However, figure 6 shows an example of a confusion matrix belonging to a three class results with a total of 30 instances distributed as:

1) 10 cats
2) 7 dogs
3) 13 rats

3x3 CONFUSION MATRIX		**Predicted class**		
		Cat	**Dog**	**Rats**
Original class	**Cat**	**7**	1	2
	Dog	2	**5**	0
	Rats	4	0	**9**

Table 3.3: example of 3x3 confusion matrix

Where, as we said, rows contain original classes and columns the predicted results. All instances that fall under the matrix diagonal represent the correctly classified (21), while the rest are to be interpreted as incorrect results (9). This being said, a further degree of difficulty has to be done if we want to achieve a more detailed analysis on the nature of the errors.

For that purpose, we will now define some key parameters that we will widely use in our experiments, and reflect our will to obtain more reliable results of our classifications than just a mere proportion of correct guesses.

True Positive (TP):	original instance of the class at hand that has been correctly classified
False Positive (FP):	original instance belonging to another class that has been classified as part of the class at hand
False Negative (FN):	original instance of the class at hand that has been classified as belonging to another class
True Negative (TN):	the remaining instances, correctly classified as non belonging to the class at hand

These four parameters conform a deeper analysis of the classifier performance per each specific class, and we may illustrate their use using the cat class of our example:

TRUE POSITIVES: 7 *7 cats correctly classified as cats*	**FALSE NEGATIVES: 3** *1 cat incorrectly classified as dogs* *2 cats incorrectly classified as rats*
FALSE POSITIVES: 6 *2 dogs misclassified as cats* *4 rats misclassified as cats*	**TRUE NEGATIVES: 14** *all the remaining animals correctly classified as non-cats (the sum of the rest of the matrix)*

Table 3.4: Table of confusion of the '*cat*' class performance

The same can be done for the other two classes. Notice that though each row adds up to the original number of instances of each class, adding the elements of each column does not necessarily sum up to the total of the specific class instances. This is not a paradox, since columns represent predicted class performance and not the original one. As a final note, comment that we will not use the TN parameter for our experimentation.

3.3. Tools

3.3.1. Operating System

Regarding the OS, we chose to develop this thesis under Linux/Ubuntu environment mainly because of its flexibility when running applications from the console command line. As we will see in the following software section, some of the required software applications were primarily developed to run under Linux OS, even presenting less conflicts if installed via console command entries rather than under Windows platforms. This is the case of openSMILE, Octave and QtOctave, which we will talk further ahead. However, other used software applications, like WEKA, have also a solid and trustworthy behaviour under Win 7 OS.

3.3.2. OpenSMILE

The Munich open Speech and Music Interpretation by Large Space Extraction (openSMILE) [24] [25] tool-kit is an open-source modular and flexible feature extractor meant for signal processing and machine learning applications. Its primary focus is clearly put on audio-signal features, enabling the user to extract large audio feature spaces in real-time, an essential part of tasks such as Automatic Speech Recognition (ASR), analysis of paralinguistic in speech or Music Information Retrieval (MIR). It combines features from two different world domains – the mentioned MIR and Speech Processing –, all of them easily selectable via a simple configuration file. For this thesis, we will be using it off-line for processing our data-set audio files.

Furthermore, thanks to its interoperability, openSMILE supports reading and writing of various data formats commonly used in the field of data mining and machine learning. From all of them, we are specially interested in output ARFF file format (Weka Data Mining). Thus, one strength of openSMILE , due to its modular architecture is that almost all intermediate data which is generated during the feature extraction process (such as windowed audio data, spectra, etc.) can be accessed and saved to files for visualisation or further processing.

openSMILE is intended to be used for research applications, demonstrators, and prototypes in the first place. In fact, nowadays openSMILE is used by researchers and companies all around the world, which are working in the field of speech recognition (feature extraction front-end, keyword spotting, etc.), the area of affective computing (emotion recognition, affect sensitive virtual agents, etc.), so, we can be considered to be under their main target group. Let us advance and discuss the feature extractors included within openSMILE package and through which we will be able to obtain our experimental results. All the following information is based on the software's comprehensive manual [26].

The program is able to extract several Low-Level Descriptors (LLD), all of which are listed in the table below, as well of able to apply various filters, functionals and transformations to these LLD (see table 3 below) in the appropriate and discussed way of section 2.4. LLD can be computed thanks to the already available feature extraction tool-kits used for speech research. For instance, the computation of Mel-frequency features (Mel-Bands, MFCC)s, as well as the Perceptual Linear Predictive coefficients (PLPCs), rely on the supplied tool-kit Hidden Markov Model Tool-kit (HTK). The realization of these computations are covered in [27]. On another hand, delta regression coefficients can be computed from the Low-Level Descriptors, and a moving average filter can be applied to smooth the feature contours.

Besides LLD extraction, openSMILE also grants the possibility to apply various filters, functionals and transformations to these low-level features. Thus, all data vectors can be processed with elementary operations such as add, multiply, and power, which enables the user to create custom features by combining existing operations. This is a not only a common technique in the emotion recognition domain, but it is used as well in Music Information Retrieval. Table 3.5 shows the list of available functionals.

Notice that both the list of LLD and functionals are based on the conclusions achieved in CEICES (Combining Efforts for Improving automatic Classification of Emotional user States) [28], a co-operation initiative between seven sites dealing with classification of emotional user states via speech. Therefore, they belong to a list of at least partly standardised feature extraction techniques. In other words, using openSMILE is a solid and reliable choice.

Provided Low-Level Descriptors		Provided functionals	
Feature Group	**Description**	**Category**	**Description**
Signal energy	Root Mean-Square & logarithmic	Extremes	Extreme values, positions, and ranges
Loudness	Intensity & approx. loudness	Means	Arithmetic, quadratic, geometric
Waveform	Zero-Crossings, Extremes, DC	Moments	Std. dev., variance, kurtosis, skewness
FFT spectrum	Phase, magnitude (lin, dB, dBA)	Percentiles	Percentiles and percentile ranges, quartiles
Cepstrum	Autocorrelation (ACF) and Cepstrum	Regression	Linear and quadratic approximation coefficients,regression error
Mel/Bark spectrum.	Bands 0-N_{mel}	Peaks	Number of peaks, mean peak distance,mean peak amplitude
Semitone spectrum	FFT based and filter based	Segments	Number of segments based on delta threshold, mean segment length
Cepstral	Cepstral features, e.g. MFCC, LPCC	Sample values	Values of the contour at configurable relative positions
Pitch	F_0 via ACF and SHS methods,Probability of Voicing	Times/durations	Up- and down-level times, rise/fall times, duration
Voice Quality	HNR, Jitter, Shimmer	Onsets	Number of onsets, relative position of first/last on-/offset
LPC	LPC coeff., Line Spectral Pairs (LSP)	Discrete Cosine Transformation (DCT)	DCT coefficients
Auditory	Auditory spectra and PLP coeff.	Zero-Crossings	Zero-crossing rate, Mean-crossing rate
Formants	Centre frequencies and bandwidths, Energy in N user-defined bands,multiple roll-off points,centroid,entropy and flux		
Tonal	CHROMA, CENS, CHROMA-based features		

Table 3.5: Description of available LLD and functionals in openSMILE package

Proceeding into how we will use this software for our purposes, our goal will be to execute this tool over our database audio files in order to extract features for emotion recognition. This will be done from command line instructions, while running the extraction tool under one of the default feature sets configuration files provided by openSMILE developers. Among those already available configuration files, we find a wide selection of possibilities. For instance, the computation of musical Chroma features (widely used for key and chord recognition), PLP and MFCC parametrization for speech recognition [29], as well as several configuration files set for emotion recognition. We will focus on the latter ones for our interests, specially three that best serves our interests since they are considered as standardised feature sets that contain widely used audio features.

First, there is the '*emobase2010*' reference set. Based on the Interspeech 2010 Paralinguistic Challenge [30] and represented by the file *'emobase2010.conf',* this feature set contains an enhanced set of LLD. It contains a total of 1.430 features, and it is the developers recommended set to be used since it represents the best approach to the current state-of-the-art in terms of affect recognition. Those features result from a base of 34 low-level descriptors (LLD), labelled as follows:

LLD	**OpenSMILE label**	**Description**
Intensity & Loudness	*pcm loudness*	The loudness as the normalised intensity raised to a power of 0.3.
Cepstrum	*mfcc*	15 MFCCs (0-14)
	logMelFreqBand	logpower of 8 Mel-frequency bands (0 – 7) distributed btween 0 and 8 kHz)
LPC	*lspFreq*	8 LSP frequencies computed from 8 LPC coefficients.
Pitch	*F0final*	The smoothed fundamental frequency contour
	F0nEnv	The envelope of the smoothed fundamental frequency contour.
Voice quality	*jitterLocal*	The local (frame-to-frame) Jitter (pitch period length deviations)
	jitterDDP	The differential frame-to-frame Jitter (the `Jitter of the Jitter')
	shimmerLocal	The local (frame-to-frame) Shimmer (amplitude deviations between pitch periods)
	voicingFinal	The voicing probability of the final fundamental frequency candidate.

Table 3.6. List of available LLD for 'emobase2010' configuration file

This data is combined with 21 functionals applied to each of these LLD contours. The names of the functionals, as they appear in the Arff file, are documented in table 3.7:

Functional type	Name under openSMILE	Description
Extremes	*maxPos*	The absolute position of the maximum value (in frames)
	minPos	The absolute position of the minimum value (in frames)
Means	*amean*	The arithmetic mean of the contour
Regression	*linregc1*	The slope (m) of a linear approximation of the contour
	linregc2	The offset (t) of a linear approximation of the contour
	linregerrA	The linear error computed as the difference of the linear approximation and the actual contour
	linregerrQ	The quadratic error computed as the difference of the linear approximation and the actual contour
Moments	*stddev*	The standard deviation of the values in the contour
	skewness	The skewness (3rd order moment).
	Kurtosis	The kurtosis (4th order moment).
Percentiles	*quartile1*	The 1rst quartile (25% percentile)
	quartile2	The 1rst quartile (50% percentile)
	quartile3	The 1rst quartile (75% percentile)
	iqr1-2	The inter-quartile range: quartile2-quartile1
	iqr2-3	The inter-quartile range: quartile3-quartile2
	iqr1-3	The inter-quartile range: quartile3-quartile1
	percentile1.0	The outlier-robust minimum value of the contour, represented by the 1% percentile.
	percentile99.0	The outlier-robust maximum value of the contour, represented by the 99% percentile.
	pctlrange0-1	The outlier robust signal range `max-min' represented by the range of the 1% and the 99% percentile.
Times/duration	*upleveltime75*	The percentage of time the signal is above (75% * range + min).
	upleveltime90	The percentage of time the signal is above (90% * range + min).

Table 3.7: List of available functionals for 'emobase2010' configuration file

Secondly, we have another set containing 386 acoustic features for emotion recognition based on the Interspeech 2009 Emotion Challenge [31]. This feature set, specified by the file '*emo_IS09.conf',* includes 16 LLD also present on table 3.6, as well as 12 functionals from table 3.7. At first glance, it is obvious that this set is far from the span of the first discussed set: it has less than half of LLD and functionals. But it will be interesting to study its performance, even if it is just to objectively prove the qualitative leap that was achieved in one year of work. Needless to say, having approximately the 25 % of the latter features will drop a lot of computational time consumption, making it easier and lighter to work and stablish modifications with it.

Finally, we considered yet another set, named under the file *'emobase.conf'*. Here, 988 features are computed from 26 low-level descriptors also presented on table 3.6, then 18 funtionals are applied to these LLD. What is interesting about this set is that it has specifically configured by openSMILE developers, and, as we can see, stands somewhat in the middle of the other pair of sets. Has roughly 66% of features from the most comprehensive set, and almost triples the lightest set. The main changes do not affect the most reliable and important LLD, such as Intensity and Loudness, MFCC (12 are computed instead of 15), pitch derived features, LSF (8 as well), or ZCR. Furthermore, list of available functionals does not differ notoriously from the other configuration.

Having presented our feature extraction possibilities, the next topic to be covered will be to describe how these audio features are stored. OpenSMILE automatically allocates them into an ARFF file. Under its header section, all the feature names, attributes and type of data are listed; on its data section, a single line of specific parameters for each source audio file is filled with its corresponding acoustic features values.

During our experiments, we will use all the stated standardised configuration sets. Though it is clear that the first one is a more updated version of the others, it is also true that its processing time will also be the largest. Thus, when achieving an Arff file that contains more than 100 audio files from our database, or even the whole database, the computation time for our classification will be far greater than the one we would get using the other two sets. As commented before, it will also be one of our main goals to compare results and performance of each configuration file.

Despite using different sets, a common and invariable fact is how we manage the last instance of our generated Arff files. One last data instance that contains the name of the emotion that belongs to the specific audio file while it is being processed. Each defined emotion from our databases will automatically define a specific class name -i.e., class emotion may contain the instances Anger, Boredom, Fear, etc.-, and will be of critical importance once processing and arranging the extracted features under Weka. This part will be deeply explained in section 4.1.

3.3.3. Octave and QtOctave

Due to our OS selection, GNU Octave [32] stable version 3.4.3 was our choice. Being a powerful, free, high-level interpreted language -similar to Matlab-, that provides enough capabilities for performing numerical experiments. Besides that, it also provides extensive graphics capabilities for data visualization and manipulation. Another main force of Octave, and of great value for our purposes, is its capability of calling command line command entries from a script text file named with its correct file extension; that is , a file *.m containing all the instructions that are to be compiled by Octave. In fact, under the command '*system ()*' it is possible to call and execute *command line* entries. This will be a great advantage when fulfilling our experiments, since we will be able to call the openSMILE feature extraction tool from within the script file, and even iterate this action to further include more audio files from our database. However, the biggest counterpart from this choice is its lack of a graphical user interface (GUI) by default. Contrary to Matlab, Octave is normally used through its interactive command line interface, with a lack of a native scrip text editor and a proper debugger.

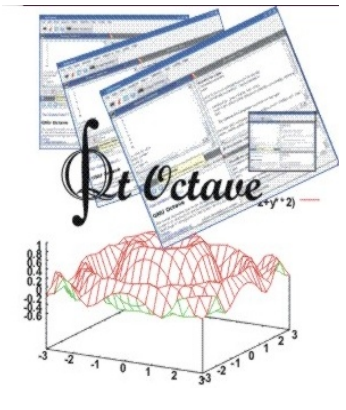
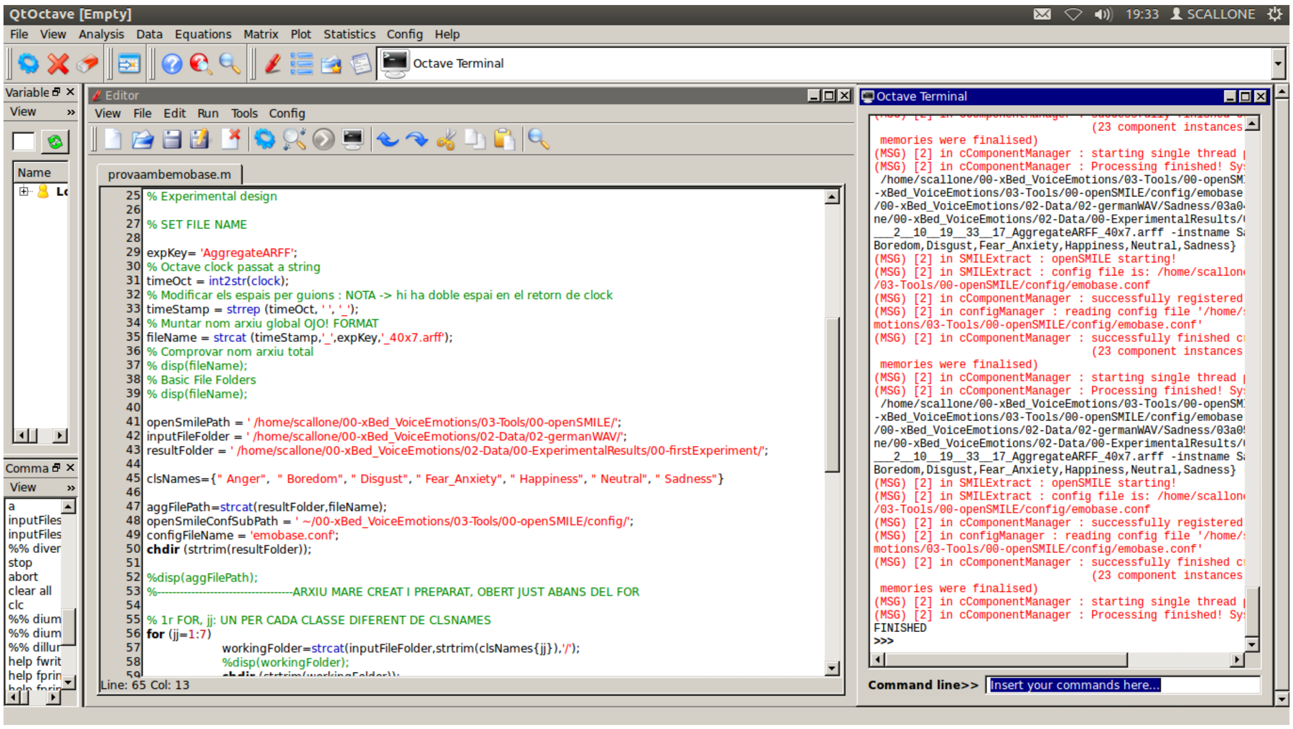


Figure 3.4: Qt Octave logo

This is why QtOctave [33] was chosen. A project born in 2006 and with a solid stable last version 10.1 from 2011, QtOctave is a front-end user interface for Octave language. Not only it simplifies its use since it is now not mandatory to operate through its original, command based version, but it also offers to the user a native text editor, a debugger, the proper command line window, and a dynamic help while typing commands as well as Octave's help finder system. Its performance and style of the GUI is similar to Matlab, resulting in a very practical all-in-one environment. See figure 3.5 below:



```
% Experimental design

% SET FILE NAME

expKey= 'AggregateARFF';
% Octave clock passat a string
timeOct = int2str(clock);
% Modificar els espais per guions : NOTA -> hi ha doble espai en el retorn de clock
timeStamp = strrep (timeOct, ' ', '_');
% Muntar nom arxiu global OJO! FORMAT
fileName = strcat (timeStamp,'_',expKey,'_40x7.arff');
% Comprovar nom arxiu total
% disp(fileName);
% Basic File Folders
% disp(fileName);

openSmilePath = ' /home/scallone/00-xBed_VoiceEmotions/03-Tools/00-openSMILE/';
inputFileFolder = ' /home/scallone/00-xBed_VoiceEmotions/02-Data/02-germanWAV/';
resultFolder = ' /home/scallone/00-xBed_VoiceEmotions/02-Data/00-ExperimentalResults/00-firstExperiment/';

clsNames={" Anger",  " Boredom", " Disgust", " Fear_Anxiety", " Happiness", " Neutral", " Sadness"}

aggFilePath=strcat(resultFolder,fileName);
openSmileConfSubPath = ' ~/00-xBed_VoiceEmotions/03-Tools/00-openSMILE/config/';
configFileName = 'emobase.conf';
chdir (strtrim(resultFolder));

%disp(aggFilePath);
%----------------------------------ARXIU MARE CREAT I PREPARAT, OBERT JUST ABANS DEL FOR

% 1r FOR, jj: UN PER CADA CLASSE DIFERENT DE CLSNAMES
for (jj=1:7)
        workingFolder=strcat(inputFileFolder,strtrim(clsNames{jj}),'/');
        %disp(workingFolder);
```

```
                                    (23 component instances
 memories were finalised)
(MSG) [2] in cComponentManager : starting single thread
(MSG) [2] in cComponentManager : Processing finished! Sy
 /home/scallone/00-xBed_VoiceEmotions/03-Tools/00-openSM
-xBed_VoiceEmotions/03-Tools/00-openSMILE/config/emobase
/00-xBed_VoiceEmotions/02-Data/02-germanWAV/Sadness/03a0
ne/00-xBed_VoiceEmotions/02-Data/00-ExperimentalResults/
___2__10__19__33__17_AggregateARFF_40x7.arff -instname S
Boredom,Disgust,Fear_Anxiety,Happiness,Neutral,Sadness}
(MSG) [2] in SMILExtract : openSMILE starting!
(MSG) [2] in SMILExtract : config file is: /home/scallon
/03-Tools/00-openSMILE/config/emobase.conf
(MSG) [2] in cComponentManager : successfully registered
(MSG) [2] in configManager : reading config file '/home/
motions/03-Tools/00-openSMILE/config/emobase.conf'
(MSG) [2] in cComponentManager : successfully finished c
                                    (23 component instances
 memories were finalised)
(MSG) [2] in cComponentManager : starting single thread
(MSG) [2] in cComponentManager : Processing finished! Sy
 /home/scallone/00-xBed_VoiceEmotions/03-Tools/00-openSM
-xBed_VoiceEmotions/03-Tools/00-openSMILE/config/emobase
/00-xBed_VoiceEmotions/02-Data/02-germanWAV/Sadness/03a0
ne/00-xBed_VoiceEmotions/02-Data/00-ExperimentalResults/
___2__10__19__33__17_AggregateARFF_40x7.arff -instname S
Boredom,Disgust,Fear_Anxiety,Happiness,Neutral,Sadness}
(MSG) [2] in SMILExtract : openSMILE starting!
(MSG) [2] in SMILExtract : config file is: /home/scallon
/03-Tools/00-openSMILE/config/emobase.conf
(MSG) [2] in cComponentManager : successfully registered
(MSG) [2] in configManager : reading config file '/home/
motions/03-Tools/00-openSMILE/config/emobase.conf'
(MSG) [2] in cComponentManager : successfully finished c
                                    (23 component instances
 memories were finalised)
(MSG) [2] in cComponentManager : starting single thread
(MSG) [2] in cComponentManager : Processing finished! Sy
FINISHED
>>>
```

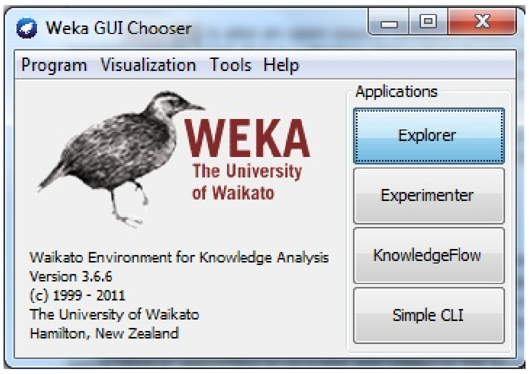
Figure 3.5: Basic QtOctave GUI window

The biggest window from this screen shot is the native editor included in the package, where we can see the smart-colouring typical of such editors. Commentaries are in green font, strings in red, etc. The window on the right acts as the Octave terminal, simulating its use through the command line window. From there, it is even possible to type commands in this classic way. Of course, it shows the results of compiling and running the scripted code. In case some errors were found, they are reported in this window. In case the code is error free, we get the results of implementing it. Keeping focused on figure 3.5, we can see the iterative process of accessing and extracting features of different audio files. Each red font block contains openSMILE running responses, containing information of each state of the extraction. For instance, informing of a successful extraction, or if the input file was correctly loaded, or even how many features were extracted and allocated into the output file and the amount of time that it took. The other black font blocks are just to ease the identification of possible errors, since its the visualization – via the command *disp()* – of specific key variables such as strings.

The last discussed factor, visualizing specific variables on screen, has been of vital importance for error recognition, as they served of visual aid. And the reason of proceeding that way is due to the fact that we were unable to properly run the debugging options included on QtOctave package. Despite the icon 'start debugging' is easily accessible from the tool bar, it would not behave they way it was expected to. The lack of automatic debugging for error spotting was a major drawback that we had to cope with during our research. But even if this meant that the polishing, error code cleaning stage was bigger than we had previously calculated, QtOctave still proved to be a very useful tool. In fact, we might even link this non functionality with some specific Ubuntu incompatibilities, or even to some error while installing the software via its source code.

3.3.4. Weka: data-mining software in Java

Weka [34] is also an open source software issued under the GNU license, now serving version 3.6.6. It is a collection of machine learning algorithms for data mining tasks. Undoubtedly, it is a powerful tool that we will use for the processing stage of our experiments. Its interoperability with Arff files has proven to be very useful, since we will be able to apply data-mining algorithms to the openSMILE output files that contain all the extracted features for emotion recognition. A brief description of our *modus operandi* regarding this software will be typed as follows, but it will be vastly developed in the next chapter, while commenting the experimental process. The required knowledge has been obtained through the software's built-in manual of use.

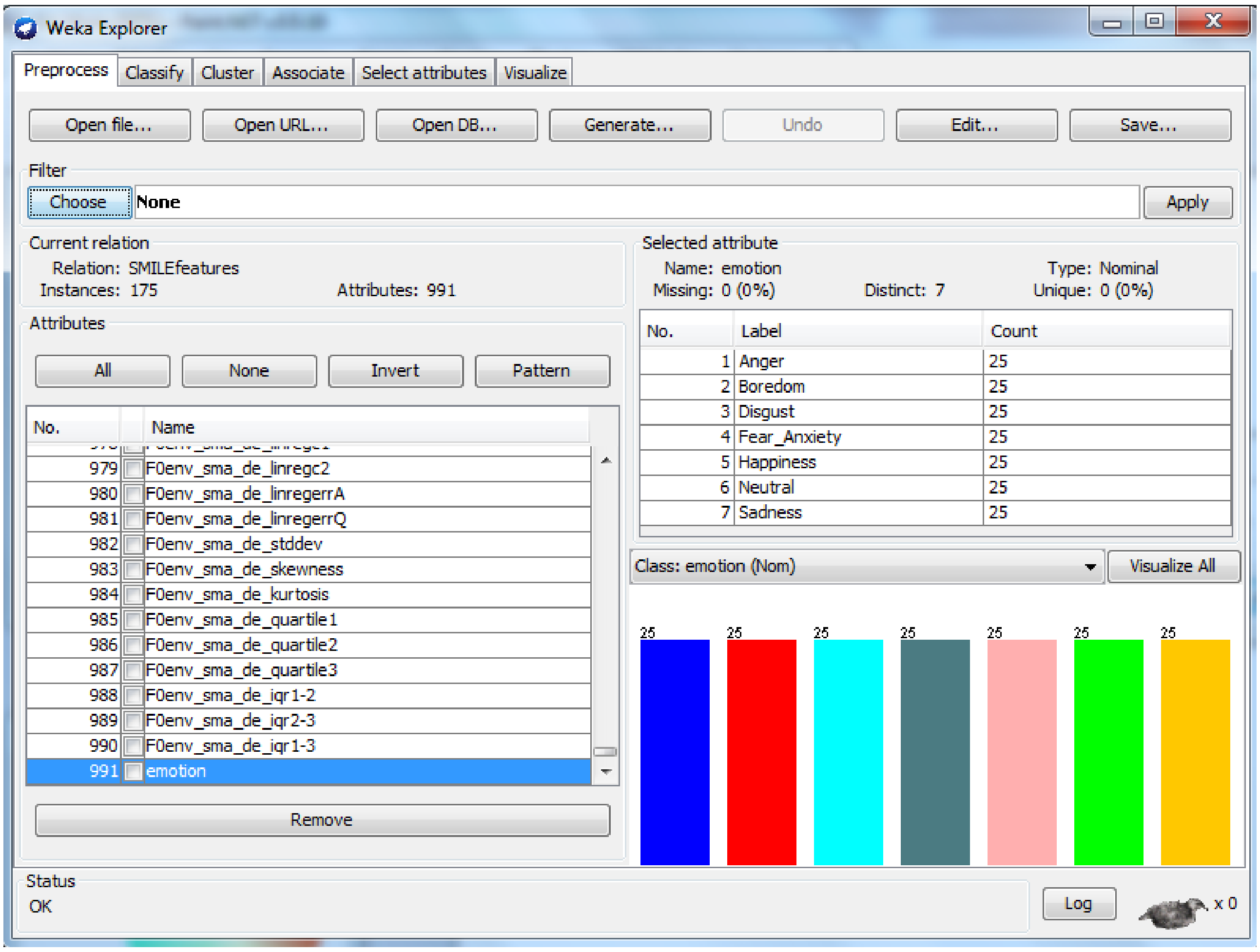


Waikato Environment for Knowledge Analysis
Version 3.6.6
(c) 1999 - 2011
The University of Waikato
Hamilton, New Zealand

Figure 3.6: Weka main GUI window

From its console menu, under the main *Explorer* tab, it is possible to choose an input Arff file that contains the original data set. In our case, it will contain the extracted speech features. Once loaded, a list of the detected attributes will be displayed, and, by clicking to either instance, we will be able to see a graphical distribution of the classes under this specific instance. From now on, a vast list of options is available to the user. For instance, the feature selection methods prior to the classifications can be done right now, as a filtering procedure to decrease the total number of instances while staying with the most relevant ones at the same time.

The next images reflects all this described information. Notice that feature selection is referred as *Filter,* with its corresponding buttons '*choose*' and '*apply*' to select one among the list (blank in the image) and to execute the attribute selection method. By doing so, the list of attributes that in the example picture shows a number of 991 would be noticeable decreased. This will be used in our experiments, when our focus will be to select an optimal, reduced number of features extracted from the original list of instances. Apart from this, the specific attribute selected from the list of 991 is the last one, labelled 'emotion', that contains the number of emotion instances from each emotion class. In this example, 25 instance of each of the seven classes, each of one with a specific colour to ease visualization. All this is under the first tab of the new “Weka explorer” window called *Preprocess*.



Filter: None

Current relation
Relation: SMILEfeatures
Instances: 175 Attributes: 991

Attributes

No.	Name
979	F0env_sma_de_linregc2
980	F0env_sma_de_linregerrA
981	F0env_sma_de_linregerrQ
982	F0env_sma_de_stddev
983	F0env_sma_de_skewness
984	F0env_sma_de_kurtosis
985	F0env_sma_de_quartile1
986	F0env_sma_de_quartile2
987	F0env_sma_de_quartile3
988	F0env_sma_de_iqr1-2
989	F0env_sma_de_iqr2-3
990	F0env_sma_de_iqr1-3
991	emotion

Selected attribute
Name: emotion Type: Nominal
Missing: 0 (0%) Distinct: 7 Unique: 0 (0%)

No.	Label	Count
1	Anger	25
2	Boredom	25
3	Disgust	25
4	Fear_Anxiety	25
5	Happiness	25
6	Neutral	25
7	Sadness	25

Class: emotion (Nom)

Status
OK

Figure 3.7: Screen capture of the Pre-process tab of Weka Explorer

The next step that we need to comment begins with clicking the second of the selectable tabs, named *Classify.* In this phase, we have entered a mode in which we will be able to classify our loaded data under various methods, depending on which classifier we choose. This is done through the vast list of available classifiers. For our experiments, we will be using several algorithms to process and organize the audio features while performing the already discussed 10- fold cross-validation (see 3.2.5) method, resulting in a more accurate and reliable results. After that, we are finally ready to execute the data mining algorithm and wait for the formulation of a summary that will contain very valuable information regarding the classifiers performance. Figure 3.8, below, shows a list of available classifiers, with the inclusion of the Bayesian theory ones, the Support Vector Machine algorithms, as well as several decision trees classifications. Figure 3.9, next page, represents the '*Classify*' tab of the Weka Explorer, and we can easily spot the discussed parameters inside this paragraph (left side). On the right side, however, we can see the summary results achieved from this particular example. Particularly, the image shows several parameters and the confusion matrix of this dummy example. Its analysis will be deeply discussed under chapter 4.

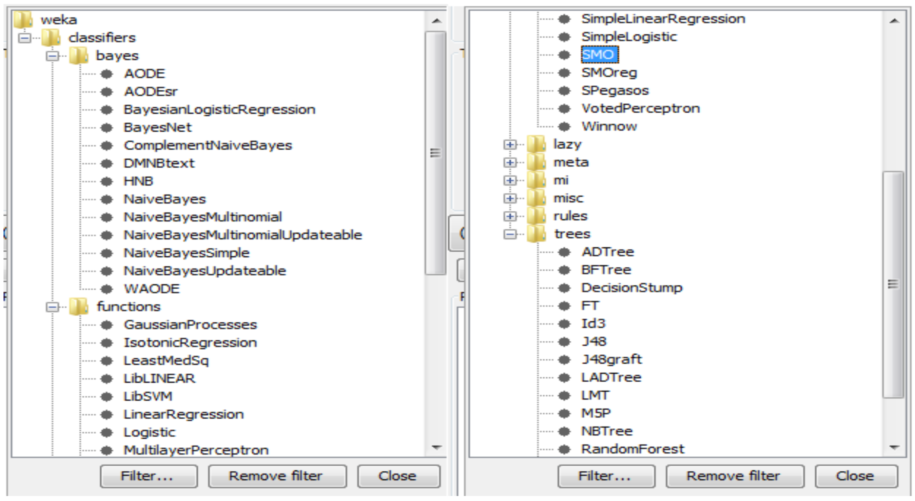
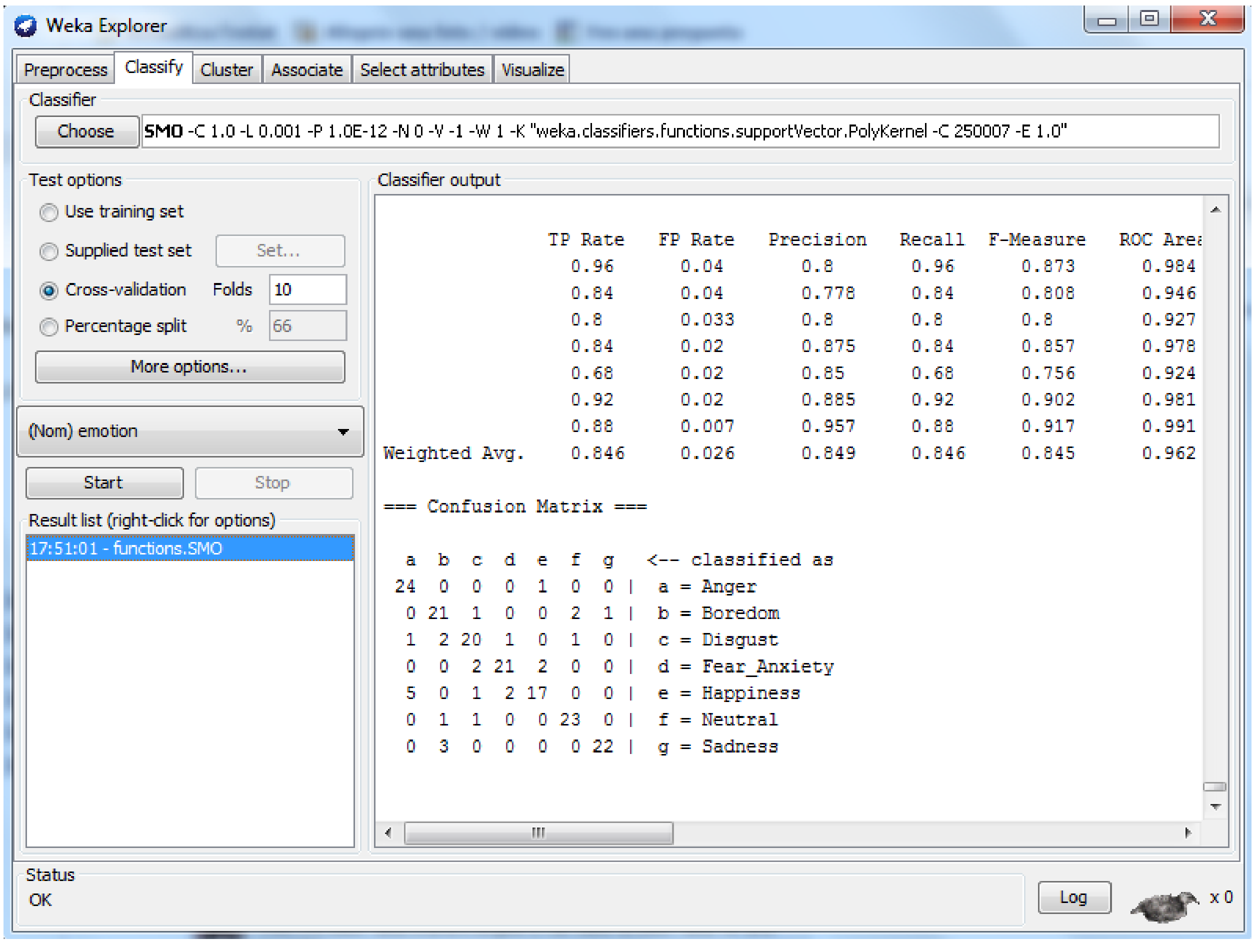


Figure 3.8: List of several available classifiers



SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"

Cross-validation Folds 10

(Nom) emotion

17:51:01 - functions.SMO

Status: OK

Classifier output

```
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area
                 0.96      0.04      0.8        0.96      0.873     0.984
                 0.84      0.04      0.778      0.84      0.808     0.946
                 0.8       0.033     0.8        0.8       0.8       0.927
                 0.84      0.02      0.875      0.84      0.857     0.978
                 0.68      0.02      0.85       0.68      0.756     0.924
                 0.92      0.02      0.885      0.92      0.902     0.981
                 0.88      0.007     0.957      0.88      0.917     0.991
Weighted Avg.    0.846     0.026     0.849      0.846     0.845     0.962

=== Confusion Matrix ===

  a  b  c  d  e  f  g   <-- classified as
 24  0  0  0  1  0  0 |  a = Anger
  0 21  1  0  0  2  1 |  b = Boredom
  1  2 20  1  0  1  0 |  c = Disgust
  0  0  2 21  2  0  0 |  d = Fear_Anxiety
  5  0  1  2 17  0  0 |  e = Happiness
  0  1  1  0  0 23  0 |  f = Neutral
  0  3  0  0  0  0 22 |  g = Sadness
```

Figure 3.9: Screen capture of a finalized classification as discussed

The results of such processing are typed automatically into the programs log, as a summary, and to conclude its output, we are given the resulting confusion matrix of the classifier. This matrix contains valuable data since it not only reflects the produced errors during the process, but it only gives information of the nature of each one. Hence, analysing the elements that do not belong to the matrix diagonal, we can extract useful information regarding the emotion classes that we work with.

Two more tabs that are of our interest remain to be commented: the *Select attributes* and *Visualize;* both can be seen on the screen capture above. The first one is basically identical to the discussed feature selection / filtering process discussed while presenting the first Weka Explorer tab, *Preprocess*. While the previous filtering was done prior to the classification, and its application returned a new set (and reduced number) of instances automatically, ready for being classified, here the same feature selection methods can be done but with no direct effect on the original attribute list. Here, the program returns the list of the attributes that passed the selection algorithm, but just as an informative way. The user should then manually select these from the original list, a quite tedious task. The main advantage of this tab, however, is that we are presented with valuable information regarding the attributes that were chosen, all presented as a log summary as well. So, while it is no practical to behave this way when feature selection methods have to be deployed to our original data sets, it sure is interesting to know the justification of such attributes choice. That is why we will use also this tab in our experiments, to understand how Weka proceeded.

The last tab that is of our interest is the one labelled *Visualize*. We are presented with a huge graphical grid that contains every 2-D graphical combination between two specific attributes containing all the instances of such specific two attributes in the 2-D space. In other words, a matrix of NxN (N being the total number of attributes – 991 in our example), and matrix element being this 2-D plot with x-axis being one of the attributes and the y-axis being another one. Ideally, we could easily spot possible correlations and associations between them graphically. Furthermore, we can choose which attributes we want to see in a specific graph, by picking them from the list available under the 'select attributes' sub-menu. We will use specific graphs, like the ROC plot, to graphically illustrate important information.

CHAPTER 4: Experimental results

Prior to jumping into the proper experimental achievements, we will present the specifications of the portable computer used. We will be running our experiments under an Intel dual Core i5-2410M processor with 4 GB of RAM available. The OS will be the mentioned Ubuntu v11.10 mainly for compatibility software issues.

That being said, in this chapter we present the experimental procedures, each of one with its specific section. The methodology is common in all of them, and is boarded once the global dataset and parameters of the experimentation have been discussed. Starting with a quick summary on what we want to achieve and the presentation of the specific chosen parameters, followed by a quick review of the specific set-up of our tools, to end it with the presentation of the final data and the proper conclusions.

General dataset

Our single source of data comes from the speech database presented under section 3.1. EMO-DB grants us 535 spoken files distributed under 6 basic emotion states, plus the class labelled neutral that simulates a non-affective speech. We will not deal with all the available data in every experiment, just sampling a representative number of audio files for each affection, or even choosing to work with smaller groups of labelled emotions.

Experiment parametrization

The aim of this section is to present a list of the general parameters that will have their role during our experiments. Logically, we will not be able to deal with all of them at once, that is why each specific experiment will gather several key parameters of that list. But to get a better view of the global scenario, we decided to include them before the tasks at hand.

Such parameters range from the labelled emotions, to the discussed configurations files of our feature extractor tool, even up to the available Weka classifiers and feature selection methods. Proceeding this way will ease the presentation and understanding of the specific experiments, since each of them will contain different configuration variables but all of them will certainly belong to this list.

Labelled emotions	Description		Weka	Description		
	Anger	128 files		Classifiers	SVM	Support Vector Machine
	Boredom	81 files			SMO	Sequential Minimal Optimization algorithm
	Disgust	46 files			LMT	Logistic Model Tree
	Fear	68 files			Naïve Bayes	Bayesian logic classifier
	Happiness	71 files		Feature transformation methods	PCA	Principal Component Analysis
	Sadness	62 files			CfsSubsetEval	Correlation-redundancy feature classification
	Neutral	79 files		Attribute selection logic	Best First	Hill-climbing technique
OpenSMILE configuration files	Emobase2010	1.430 features			LFS	Linear Forward Selection
	Emobase2009	386 features			Ranker	Presents a list of ranked features
	Emobase	990 features			Scatter search	Performs a scatter search

Table 4.1: List of global parameters

4.1. Experiment 1

First, we wanted to know how would our set-up behave and what potential conclusions could be extracted from our approach to the topic of emotion recognition. For our first experiment, we decided to run the openSMILE feature extraction tool over 40 audio files from each emotion of the Berlin database of Emotional Speech. A total of 280 instances, equally divided under Anger, Boredom, Disgust, Fear/Anxiety, Happiness, Neutral and Sadness. We decided to do this equally distributed partition to normalise the instances from the quite asymmetrically distributed database (see table 3). Afterwards, they would be processed by our feature extraction tool following the default configuration set '*emobase2010*', able to extract 1.430 attributes from each audio file. The resulting generated Arff file containing all the information above would be then loaded into Weka for further analysis.

4.1.1. Building the appropriate Arff file

For the purpose of building such an Arff file that contained the mentioned instances, the main step was to be able to iterate the console command line that enabled the execution of the openSMILE feature extraction tool with the right configuration options. Doing it manually was discarded for obvious reasons, so we began to develop the first script text that would be able to do it correctly. By using the native text editor included in the QtOctave software package, compiling and executing it under Octave we found an effective way to proceed. The script code can be found under Appendix B.1.

From an eagle-eyed point of view, the code should look sequentially into each folder of the mentioned database and execute the openSMILE command line instruction 40 times per folder. Once the latter iteration was completed, it should move towards the next folder, in order to start the same procedure. This methodology was implemented by a double *for* iteration, one enduring 7 times and the other (placed inside the first) consisting of 40 iterations. Due to the openSMILE solid implementation, in order to append new data into the same generated Arff file, one should only call the same output file as much as it is needed, so there should be no problems such as writing new formatted data into a previous opened file: it is automatically and well put at the previous end-of-file. However, there was one thing to be concerned about. How could we create an specific attribute that contained the different emotion names? This was crucial for further analysis and organization of the generated data, since otherwise Weka algorithms would not correctly identify each emotion class. This problem was solved by adapting the default command line that was meant to execute openSMILE extraction tool into a more complete one, adding other arguments than the ones that were described in its user's guide. Let's see it as an example of the command line entry:

SMILExtract -C config/emobase.conf -I input.wav -O output.arff

This is the default, simplest form of command line to generate a desired Arff file (argument -O stands for output, and *output.arff* shall be the name of this file) with all the features extracted from the input file (argument -I stands for input, and *input.wav* is the name of the audio file) following one specific configuration default feature set (argument -C stands for configuration, the rest is the path to this so called file). By doing this, the last attribute coded into the output file is assigned into a dummy class label named emotion, and filled by one class '*unknown*' by default. To change this behaviour and assign custom classes and class labels to each individual instance, openSMILE grants us to deploy several more arguments into the command line entry. Those are:

i). the parameter *-classes ,* which specifies the list of nominal classes including to be put between brackets
ii). the parameter -*classlabel,* which specifies the class value of the instance that is currently being computed from the given input (-I)

Hence, the appropriate body that will fulfil our purpose will look like for an audio file from the Anger emotion:

SMILExtract -C config/emobase.conf -I input.wav -O output.arff -instname Anger -classes {Anger, Boredom, Disgust,Fear_Anxiety, Happiness, Neutral, Sadness} -classlabel Anger

4.1.2. Data analysis

As described in 3.3.4, we should carry on with the data analysis by using Weka data-mining possibilities. The chosen classifier for this experiment will be the Weka labelled *SMO,* a SVM based classifier with improved performance*.* After loading our data file, selecting the mentioned classifier and starting the experiment, Weka returns us the summary in form of a text log, attached in the following three tables.

Correctly Classified Instances	231 (82.5 %)
Incorrectly Classified Instances	49 (17.5 %)
Kappa statistic	0.7958
Mean absolute error	0.2074
Root mean squared error	0.3064
Relative absolute error	84.6825 %
Root relative squared error	87.5509 %
Total Number of Instances	280

Table 4.2: Overall performance statistics

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Anger	0.9	0.033	0.818	0.9	0.857	0.982
Boredom	0.75	0.046	0.732	0.75	0.741	0.942
Disgust	0.875	0.017	0.897	0.875	0.886	0.971
Fear_Anxiety	0.85	0.038	0.791	0.85	0.819	0.959
Happiness	0.7	0.021	0.848	0.7	0.767	0.934
Neutral	0.8	0.033	0.8	0.8	0.8	0.938
Sadness	0.9	0.017	0.9	0.9	0.9	0.984

Table 4.3: Detailed accuracy parameters per each class

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
36	0	1	1	2	0	0	**Anger**
0	30	0	0	0	7	3	**Boredom**
1	1	35	1	0	1	1	**Disgust**
2	0	1	34	3	0	0	**Fear_Anxiety**
5	0	1	6	28	0	0	**Happiness**
0	6	1	1	0	32	0	**Neutral**
0	4	0	0	0	0	36	**Sadness**

Table 4.4: Confusion matrix

The stratified cross-validation paints an accurate picture, since the accuracy is of 82.5 % (see table 4.2) From the total of 280 instances, 49 were mistakenly classified. The kappa statistic measures the agreement of prediction with the true, original class of each instance – a value of 1.0 would mean a complete agreement, while any value greater than 0 would mean that our classifier is doing better than chance. The following error values are not very meaningful for classification tasks.

We will now skip to the confusion matrix results (table 4.4), since as we will see, there is a correlation between its data and the other set of parameters given in the summary. Accordingly to what we described in section 3.2.6 , we have a 7x7 matrix since the number of classes is seven, see Table 9. The number of correctly classified instances is the sum of diagonals in the matrix (231), since each of the diagonals elements carries the information of correctly classified instances for one sole class; all others are incorrectly classified (49).

For example, we can see that the original instances from the class Anger and Sadness are the best classified ones, while classes Boredom and Happiness present the worse results. Specially remarkable are the misclassification due to both false and true positive of both Fear-Happiness (6+3) and Boredom-Neutral (7+6) class instances. Worth mentioning is also the amount of originally happiness instances that fell under the anger class after classification (5). Refer to section 3.2.7 for definition of the confusion matrix associated terms used in the present paragraphs.

We move to discussing the final set of results (table 4.3). Each row represents the specific results for each class; while each column represents the results of all the classes given one of the parameters that we will discuss now.

i). True Positive (TP) rate is the proportion of instances which were classified as class X, among all instances which truly have class X. In other words, how much part of one class was correctly captured. It is equivalent to Recall (forth column). In the confusion matrix, this is the diagonal element divided by the sum over the relevant row. It's the same as dividing the TP value by the sum of itself and the false negatives (FN) of the class.

$$TP\,rate = \frac{TP}{TP+FN}$$, ex: Disgust: $\frac{35}{40}=0.875$ Happiness: $\frac{28}{40}=0.7$

ii). False Positive (FP) rate is the proportion of instances which were classified as class X, but originally belonged to a different class, among all examples which are not of class X. In the matrix, this is the column sum of class X minus the diagonal element, divided by the rows sums of all other classes; i.e.

Anger: $\frac{1+2+5}{40\cdot 6}=0.033$ Sadness: $\frac{3+1}{40\cdot 6}=0.017$

This example helps to clarify the true and false positive differences. Notice that though these two classes have the same TP value (36), their different false positive (FP) values derives into different FP rates, being one the double of the other. This means that despite all the same number original instances under both classes were correctly classified, their features intrinsic correlation among other classes made that the double of instances belonging to different classes ended up being classified as Anger.

iii). Precision is the proportion of the examples which truly have class X among all those which were classified as class X. In the matrix, this is the diagonal element divided by the sum over the relevant column. Or the TP value of the class divided by its same TP and the sum of the rest of FP caused by the remaining classes.

$Precision=\frac{TP}{TP+\Sigma FP}$, ex: Boredom: $\frac{30}{30+1+6+4}=0.732$

Notice that the effect commented in (ii) can be also seen through the Precision column. The class Anger has the highest TP rate, but has not the highest Precision. Sadness has them both (0,9), coincidentally, in this case. This examples shows the Boredom class, having the highest added FP value in total (11), the majority of them being originally instance of the neutral class (6).

iv).F-Measure stands for the following formula, and represents a combined measure for Precision and Recall. Generally, the closer to 1, the better overall results:

$$\frac{2\times Precision\times Recall}{Precision+Recall}$$

v).The ROC Area coefficients presented in the last column stand for the computed area under the ROC curve for each classified class. This is highly linked to what Receiver Operating Characteristics (ROC) [35] graphs are used in the domain of machine learning and data mining. The ROC space is a 1x1 2D grid with TP rate [0..1] plotted on the Y-axis and the FP rate [0..1] is plotted on the X-axis, hence each point conforming the curve represents a depiction of a classifier performance. Having in mind this 2D grid, the ideal behaviour of such curves would be to be allocated as near as possible to the grid point (0,1), which would mean zero FP rate and perfect TP rate. So, by computing the area under the curve (AUC), WEKA tests the ability of the classifier to correctly classify its classes – the closest to 1 the better, while an AUC below or equal 0.5 would be worthless (i.e., classified by pure chance).

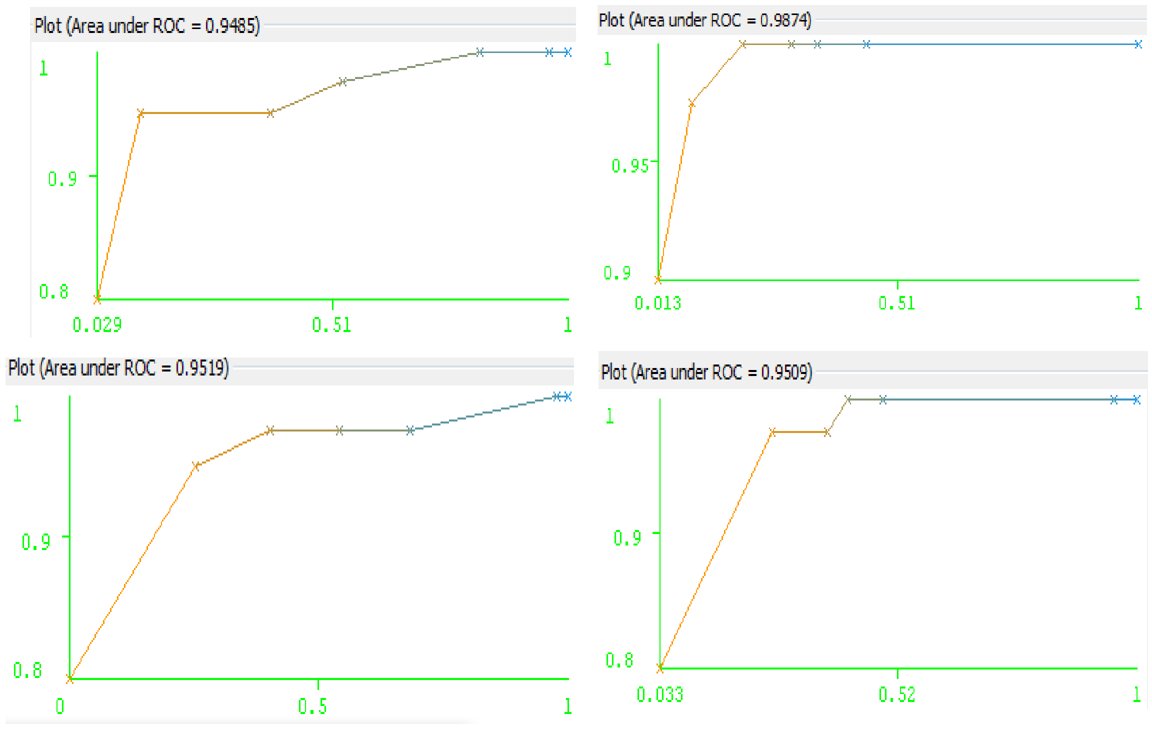


Figure 4.1: 2-D ROC plot for 4 emotions: Fear, Sadness, Happiness and Boredom (from left to right; up to down)

4.1.3. Conclusions

The performance of SMO classifier has proven to be a very good choice for our research, and we will continue to use it for further experimenting. Despite being an environment test to try all the tools that we have in our hands, we can extract several facts from it that will hopefully remain true later on.

Firstly, the global accuracy of 82,5 %. From 280 files, only 49 emotive audio files were mistakenly classified. Towards building a HCI, call-center, artificial intelligence entity, such results would certainly be very welcome. Secondly, and focusing on the analysis of tables 8 and 9, we can see several facts discussed during the presentation of emotional spaces and their classification (see section 2.2). In particular, the limitations of the dimensional approach are slightly shown. For instance, we can see that emotions Boredom and Fear contain the highest false positive rates (11 and 9) – or if we are looking at the confusion matrix, the columns with more non-zero elements.

On the other hand, Happiness and Boredom also have the highest false negative (FN) values (10 and 12) – quickly seen from their class rows, non-zero elements. If we imagine a V-A or VAD space, we would see that such emotions occupy an unclear and not distinguishable subspace, but more likely a portion of space that is common or overlaps with other emotion subspaces. This would explain the errors of the classifier. And third and finally, the Neutral, emotionless, class needs a comment for itself. Sharing the third highest false positive and false negative rates, we can see that its true nature - speech without any emotion - also misleads the classifier. Specific language traits linked to German language are most probably the explanation of this matter, and it would surely be interesting to see what happened if using different language databases.

Figure 4.2 illustrates the distribution of each class errors, either taking into account the FN and the FP errors (a total of 49 per each category). It will illustrate the confusion matrix non-diagonal elements distribution, as well as it will help us to easily identify the major problems of this classifier when it comes to correctly classify the instances.

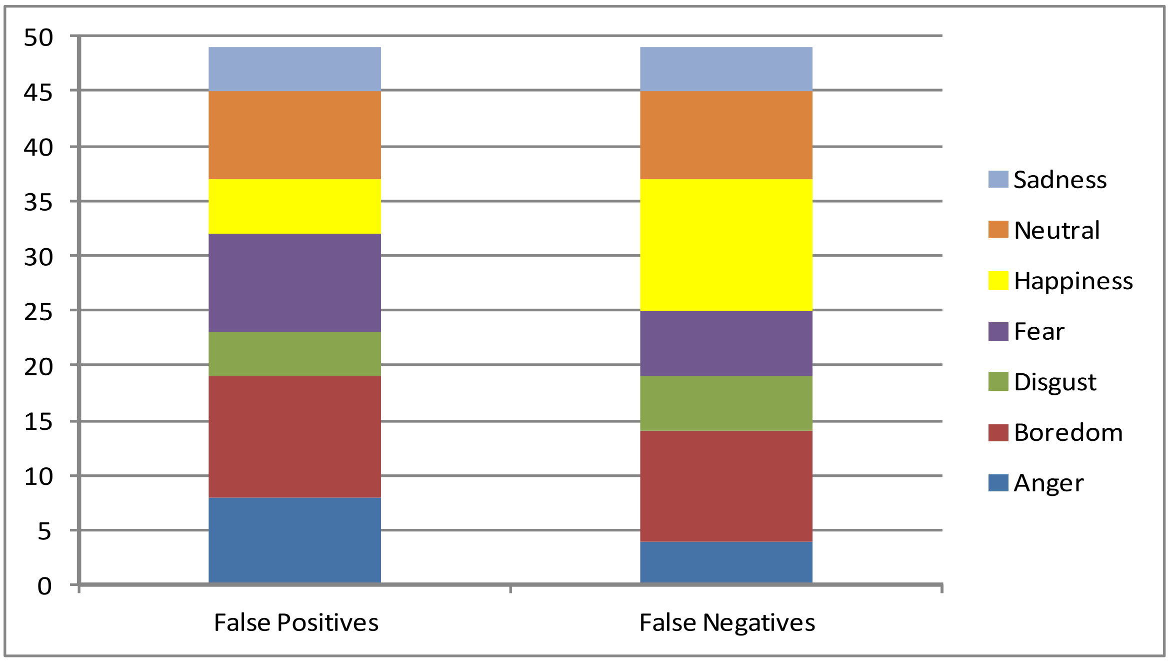


Figure 4.2: Classifier's errors weight distribution per class

Notice the changes, weight-related, of classes such happiness (250 % increase) or anger (200 % decrease) and even fear when the different error nature is taken into account, while others remain almost unchanged. For instance, sadness and disgust achieve good results in both charts (low contribution to the overall errors), while errors concerning the neutral and boredom classes are almost equally distributed by the classifier - and equally damaging, in this case, since their contribution is far greater.

It will be interesting to see the evolution of the characteristics that we sketched in the last paragraphs during our future experiments, whether they will be aggravated or decreased accordingly to what classifier we use, or the percentage of the available database that we take into account. But without doubt, the most important conclusion of this text experiment is that the Sequential Minimal Optimization algorithm for training a support vector classifier offers an overall satisfying performance with our data classification.

4.2. Experiment 2

After getting familiar with all the main actors of our research, we considered that our next step would be to compare the performance of various classifiers when working with the whole database. The procedure is similar to the last experiment, but now we will be handling 535 audio files from which we will get their acoustic features extracted according to the three configuration files introduced in section 3.3.2. To that matter, the script code will also have to be modified as we will see below. Once we are able to run classification algorithms with WEKA, it will be time for testing several classifiers of different nature to evaluate and compare their performance. The results will be presented at the end of this section.

So, we are pursuing two major objectives in this experiment. One, obtain substantial proof of the progress made by researchers regarding the detection of the best acoustic features to be extracted from speech audio signals from less than two years work. This will be done by comparing the performance of one classifier when excited with data belonging to the three different feature set files, since one of them is from the year 2009, and the other two from the end of 2010. The second objective will be to compare the usefulness of various classifiers. We have chosen ones from different nature, very distinct from each other, and analysing the output information, we will be able to stablish their strengths and weaknesses.

4.2.1. Building the different feature extracted Arff files

The first topic in our hand was to adapt the octave script file to our needs. Several changes had to be done, while everything else would still prove useful now thanks to our scope on making the code easily adaptable. On one hand, all changes which depended on the fact that we were dealing with the whole database now, not just 40 files from each emotion. And, as table 3.1 showed, each emotion has a different number of audio files.

That is why, for example, the iterative *for* needed to be adapted to the cause; instead of a final integer to end the loop, a local variable containing the specific information of the number of files present on the local working folder was used. On the other hand we find all code changes necessary to successfully interact with the appropriate configuration files needed at each time. The most important change affected the main code line, the one that calls our feature extraction tool needed to be adapted to the new configuration files. Appendix B.2 contains the general code used for this experiment.

After having our script file correctly coded, we ran it under our chosen three configuration sets (see 3.3.2), and we achieved three different Arff files with notoriously differences in file weight due to the great variety of total data: 10 MB the heaviest, coming from '*emobase2010*' and 2.5 MB the lightest, '*emo_IS09*'.

4.2.2. Data analysis

The next step includes all the WEKA related work, that is, applying different classification algorithms to the three data files. Among the list, we selected one that follows the Bayesian logic (Naïve Bayes), the previously used SMO algorithm that showed very good performance, and a third one based on decision trees algorithms (LMT). We will present the results under sub-sections, each one containing the results with a specific classifier and the three different feature set files in a similar way and with already familiar parameters of the first experiment: confusion matrix, detailed accuracy parameters per class and global parameters.

4.2.2.1. SMO classification

Performance using acoustic features based on Interspeech '09 (386 features)

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
114	0	0	1	13	0	0	**Anger**
0	68	1	1	0	6	5	**Boredom**
3	3	35	2	0	1	2	**Disgust**
6	1	3	52	4	1	1	**Fear_Anxiety**
11	0	1	4	55	0	0	**Happiness**
1	6	0	0	1	71	0	**Neutral**
0	8	1	0	0	0	53	**Sadness**

	Anger	Boredom	Disgust	Fear	Happiness	Neutral	Sadness
TP Rate	0.891	0.84	0.761	0.765	0.775	0.899	0.855
FP Rate	0.052	0.04	0.012	0.017	0.039	0.018	0.017
Precision	0.844	0.791	0.854	0.867	0.753	0.899	0.869
ROC Area	0.961	0.958	0.956	0.957	0.931	0.98	0.982

Table 4.5. SMO: Confusion matrix and specific performance parameters for '*emobase2009*' set

Performance using acoustic features based on Interspeech '10 (1430 features)

Anger	**Boredom**	**Disgust**	**Fear_Anxiety**	**Happiness**	**Neutral**	**Sadness**	
113	0	0	2	13	0	0	**Anger**
0	72	0	0	0	5	4	**Boredom**
2	1	38	2	0	2	1	**Disgust**
4	1	0	57	4	2	0	**Fear_Anxiety**
13	0	1	3	54	0	0	**Happiness**
0	5	1	4	1	68	0	**Neutral**
0	2	0	0	0	1	59	**Sadness**

	Anger	**Boredom**	**Disgust**	**Fear_Anxiety**	**Happiness**	**Neutral**	**Sadness**
TP Rate	0.883	0.889	0.826	0.838	0.761	0.861	0.952
FP Rate	0.047	0.02	0.004	0.024	0.039	0.022	0.011
Precision	0.856	0.889	0.95	0.838	0.75	0.872	0.922
ROC Area	0.964	0.973	0.974	0.962	0.933	0.972	0.993

Table 4.6: SMO: Confusion matrix and specific performance parameters for '*embase2010*'

Performance using acoustic features based on 'emobase' (990 features)

Anger	**Boredom**	**Disgust**	**Fear_Anxiety**	**Happiness**	**Neutral**	**Sadness**	
118	0	1	2	7	0	0	**Anger**
0	72	1	0	0	4	4	**Boredom**
1	2	38	3	0	2	0	**Disgust**
6	0	2	55	2	1	2	**Fear_Anxiety**
12	0	1	4	54	0	0	**Happiness**
1	4	0	1	0	73	0	**Neutral**
0	7	1	0	0	1	54	**Sadness**

	Anger	**Boredom**	**Disgust**	**Fear_Anxiety**	**Happiness**	**Neutral**	**Sadness**
TP Rate	0.922	0.889	0.826	0.809	0.761	0.924	0.871
FP Rate	0.049	0.029	0.012	0.021	0.019	0.015	0.013
Precision	0.855	0.847	0.864	0.846	0.857	0.913	0.9
ROC Area	0.973	0.977	0.955	0.964	0.946	0.982	0.989

Table 4.7: SMO: Confusion matrix and specific performance parameters for '*emobase'* set

Global performance comparison

Config. file	emobase09	emobase2010	emobase
Correctly Classified Instances	448 (83.7383 %)	461 (86.1682 %)	464 (86.7289)
Incorrectly Classified Instances	87 (16.2617 %)	74 (13.8318 %)	71 (13.2711 %)
Kappa statistic	0.8067	0.8358	0.8422
Computational time	3”347	7”189	5”122

Table 4.8: SMO: Comparison of overall performance

We obtained very similar results, with an accuracy performance of 3 % variation that directly reflects on these maximum of 16 instances incorrectly classified. Although we could say that this one year of research (2009-2010) reflected on the different extracted features might not have made the desired impact on the performance values, we believe that it is too soon to make this statement. We will definitely board this issue in the final section of this experiment, once we have worked with the different classifiers.

That being said, we will comment the general strengths and weaknesses of the SMO classifier now that we have tested it more deeply than in the first experiment (see table 4.8). On one hand, we can see that the performance is well above 80% just like in the former experiment: despite dealing with the whole database, the results are excellent, thus making this classifier a very trustworthy one. Even more, the feature set that achieved the best results is the specifically designed by openSMILE developers (third column), that combines the approach of Inter Speech 2010 with what seems to be, at least until this point, a more optimal set of features. Tough having only 3 more correct instances of the total of 535 than the '*emobase2010*' set, we must not forget this set has approximately 40% less of features than the other and yet it performs equally/ slightly better. Lastly, we can't skip to comment the very good results of the oldest feature set, performing almost up to 84 % despite having one thousand less features than the heaviest set. It will be interesting to see its results on upcoming experiments.

On the other hand, we will board the discussion of its weaknesses. To begin with, we shall comment the differences between such feature sets based on the final classification (see table 4.7). In general, '*emobase2010*' is a slight improvement of the oldest set, '*emobase2009*', and the 13 instances correctly classified basically come from improving the 'boredom' class precision rate (up to 50 % less errors) and the 'sadness' instance classification. However, there is a small decrease of performance regarding the correct detection of the classes 'fear' and 'neutral', while the already present problem of the 'anger' versus 'happiness' misleads has not been solved via the 2010 feature set. Concerning the last set, the one specifically designed by openSMILE developers, we can see that its performance is almost the same as the largest set with the only improvement of the better classification of the 'happiness' class (50% of the errors). Several instances now incorrectly classified prevent this last feature set from achieving a notable improvement from '*emobase2010*'.

Focusing on the difficulty that SMO has to correctly tell apart instances from the Anger and Happiness classes, whose incorrect matches (specific TP and FP of both classes) add up to a 27%, 35 % and 26 % of the total incorrect classified instances. Of course, this has to do with the fact that the Anger class holds 128 instances, a lot more than the others, but as we will see later when normalizing the specific weights of each class, this effect is still very noticeable. Other minor peaks of this two-class overlapping phenomena occur, but their effect is not that much noticeable and not common along the three different scenarios. The next graph (figure 4.3) illustrates the distribution of the total number of misclassified instances from the three scenarios put altogether (a total of 232). Needless to say, it has been taken into account the fact that each class has a different number of instances: we have computed a specific weight for each class to normalize the scenario.

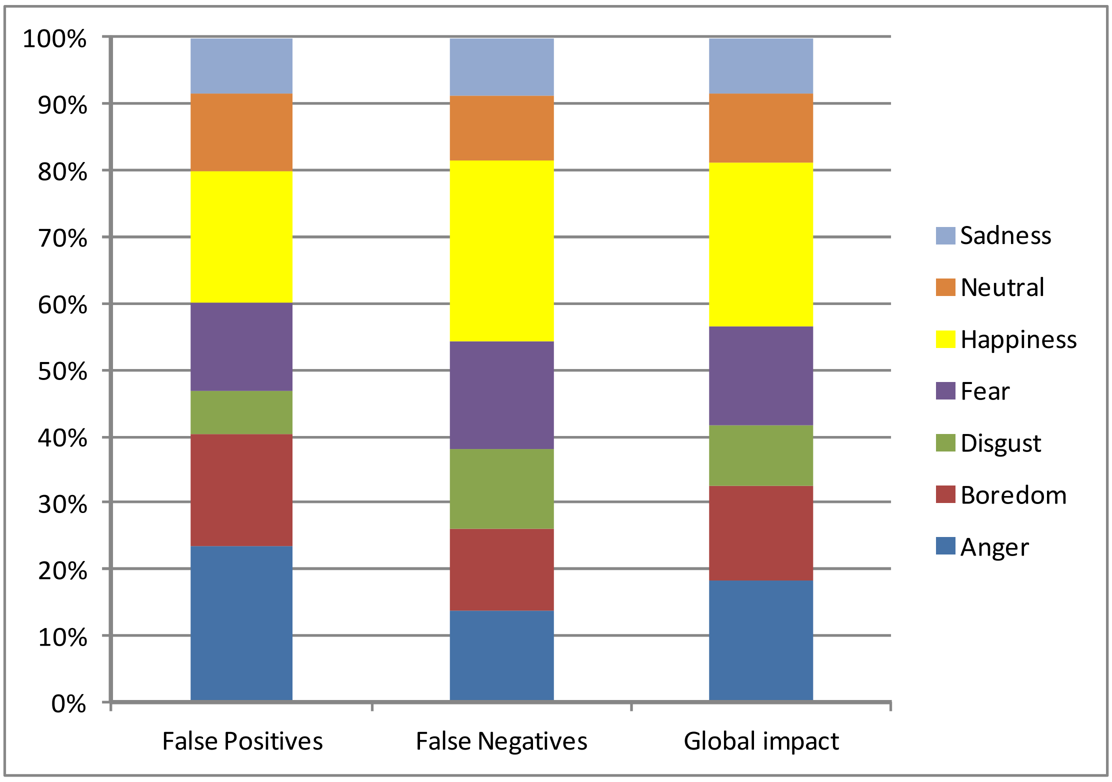


Figure 4.3: Bars showing the FP, FN and average error distribution of each class (percentage-wise)

Notice that the commented effect is present despite having normalized the scenario. Also, the bar on the far right represents the average value of FN and FP per class, thus showing a global view of which are the classes that gather the most error contribution rates. From there, we can easily spot the mentioned duo Anger-Happiness that attain the most errors, while sadness, disgust and neutral remain in the opposite side of such ranking. Certainly very similar results as the ones from the first experiment, a logical fact if we take into account that the same classifier was used. We can al least conclude that having a highly asymmetrically distributed database does not affect the results – once normalized.

Following with the anger and happiness behaviour, it certainly is shocking how these two antagonist emotions (from a social perspective) have this tendency of sharing so many similar feature values, up to the point of misleading the classifier's algorithm. We could impute the German language both linguistic and paralinguistic specific characteristics to that matter. But, besides that, both emotions might share an intrinsic correlation independent to the spoken language. For instance, both emotions are defined by having a positive and strong Arousal and Dominance values (see figures 2.1 and 2.2). Besides these two emotions, the class' boredom' also shares an important part of these errors. Characterized as having negative Arousal, Valence and Dominance values, we could link part of the reason for such incorrect classification to the fact that this emotion shares a more common subspace with other emotions such as 'sadness', and quite near to where we would allocate 'fear' and 'disgust'. In fact, by looking to the confusion matrices, we can see that the majority of errors while assigning instances to the 'boredom' class come from misleads from instances originally belonging to the 'sadness' class (17 in total). Of course, for being completely thorough regarding this subject, a complex and deep analysis of the acoustic feature values should be done, but this is out of our scope.

Besides that, is it also important to notice which are the emotions more distinct and uncorrelated to the others; the ones with the best combination of highest precision and lowest false positive rates. Figure 14 also reflects this issue. Among those, we find the classes 'disgust' and 'sadness'. As sketched during the conclusions of the first experiment, we should also write a comment for the 'neutral' class. Allocated in the average group of mislead classification, it is the second main reasons to why the 'boredom' class errors have occurred. A total of 15 originally neutral instances have been labelled as 'boredom' ones after the classification, and curiously, the same number of originally bored states have been labelled as neutral states afterwards. From a social point of view, we find this phenomena more explainable than the issue with 'anger' and 'happiness' classes, since a bored state seems more appealing to be categorized as an emotionless one. Besides that matter, the 'neutral' class seems to be correctly classified by SMO.

4.2.2.2. Naïve Bayes classification

Performance using acoustic features based on Interspeech '09 (386 features)

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
95	0	7	6	19	1	0	**Anger**
0	51	9	1	1	9	10	**Boredom**
1	4	32	5	1	1	2	**Disgust**
10	2	4	36	4	10	2	**Fear_Anxiety**
18	0	1	7	40	5	0	**Happiness**
1	6	4	8	5	53	2	**Neutral**
0	9	1	3	0	1	48	**Sadness**

	Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness
TP Rate	0.742	0.63	0.696	0.529	0.563	0.671	0.774
FP Rate	0.074	0.046	0.053	0.064	0.065	0.059	0.034
Precision	0.76	0.708	0.552	0.545	0.571	0.663	0.75
ROC Area	0.931	0.937	0.873	0.859	0.884	0.909	0.977

Table 4.9: N-B: Confusion matrix and specific performance parameters for '*emobase2009*' set

Performance using acoustic features based on Interspeech '10 (1430 features)

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
103	0	0	7	17	1	0	**Anger**
0	63	2	2	0	9	5	**Boredom**
2	4	35	2	0	1	2	**Disgust**
3	1	2	44	6	10	2	**Fear_Anxiety**
19	0	1	6	42	3	0	**Happiness**
0	10	0	5	1	63	0	**Neutral**
0	4	0	0	0	3	55	**Sadness**

	Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness
TP Rate	0.805	0.778	0.761	0.647	0.592	0.797	0.887
FP Rate	0.059	0.042	0.01	0.047	0.052	0.059	0.019
Precision	0.811	0.768	0.761	0.647	0.592	0.797	0.887
ROC Area	0.955	0.954	0.941	0.931	0.91	0.936	0.986

Table 4.10: N-B: Confusion matrix and specific performance parameters for '*emobase2010*' set

Performance using acoustic features based 'emobase' (990 features)

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
13	0	1	2	112	0	0	**Anger**
0	79	1	0	0	1	0	**Boredom**
0	14	20	2	9	1	0	**Disgust**
3	6	0	27	29	2	1	**Fear_Anxiety**
5	1	0	3	62	0	0	**Happiness**
0	36	0	3	7	33	0	**Neutral**
0	17	0	0	0	0	45	**Sadness**

	Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness
TP Rate	0.102	0.975	0.435	0.397	0.873	0.418	0.726
FP Rate	0.02	0.163	0.004	0.021	0.338	0.009	0.002
Precision	0.619	0.516	0.909	0.731	0.283	0.892	0.978
ROC Area	0.719	0.932	0.924	0.933	0.718	0.951	0.985

Table 4.11: N-B: Confusion matrix and specific performance parameters for 'emobase' set

Global performance comparison

Config. file	emobase09	emobase2010	emobase
Correctly Classified Instances	335 (66.6651 %)	405 (75.0009 %)	279 (52.1495 %)
Incorrectly Classified Instances	180 (33.3349 %)	130 (24.2991 %)	256 (47.8505 %)
Kappa statistic	0.602	0.717	0.4439
Computational time	1”645	3”241	2”386

Table 4.12: N-B: Comparison of overall performance

Contrary to our last analysis SMO classifier, in this case we have a very distinct scenario. We will board the comments by discussing the performance of this Bayesian logic classifier while excited with data belonging to the configuration files '*emobase09*' and '*emobase10*', and we will leave the other results for later.

The newest and largest feature set now proves its usefulness, providing almost a gain of 10% in performance if contrasted with the older, lighter set. This directly links with the increase of 50 instances now correctly labelled, achieving a total accuracy of 75 %. A rather good behaviour, that has the other advantage of being a less time demanding classifying technique. While in the last section it was not clear which configuration set was the way to go, here the election is clear..

Proceeding into explaining the improvement of the classification via the tables we obtained, we can clearly see that those 50 instances now well labelled come from a rather equally distributed improvement of each specific class detection. Thus, in this case the use of more and probably better features is justified since the positive effect affects all classes without exception. In particular, 'boredom' and 'neutral' instances are the most favoured ones. But despite the fact that this improvement is indisputable, this classifier has some serious counterparts that do not get solved even though we used a newer set of features.
The most noticeable one is the mutual harm that the classes 'anger' and 'happiness' do to each other. We are very familiar with this phenomenon since the same occurred with the SMO classifier (see 4.2.2.1), but now with an even bigger effect. From the total of 310 incorrect instances from both sets classification, 72 of them are caused by the misclassification of just these two (labelled instances as 'anger' while being 'happiness' and vice-versa). Particularly dramatic is the case of 'happiness', where almost half of its original instances fall under the 'anger' class after classification (very high FN value). Besides that, other problems arise. Taking a look to the true positive and precision rates, we find that also specially important are the incorrect classifications of originally 'fear' and 'disgust' instances. In fact, just 'boredom' and 'neutral' are slightly above the global performance, while sadness stands alone as the best classified class. The next chart (figure 15) will graphically exemplify all the mentioned above. We have computed the numbers using both sets, and we have proceeded the same way as figure 4.3.

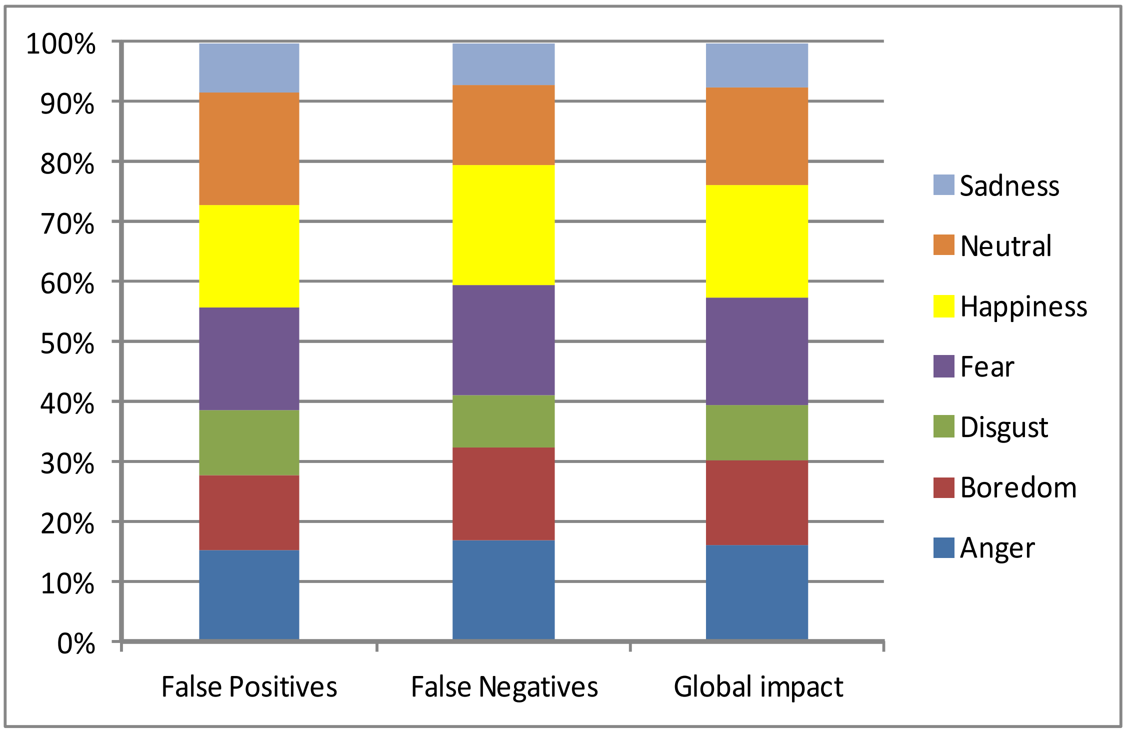


Figure 4.4: N-B: Error distribution per nature and average contribution (percentage-wise)

We can easily see the impact of misleads from the three commented classes: happiness, fear and anger; while neutral, boredom and sadness remain on a more acceptable level. A little bit misleading is the effect of the 'disgust' class. Taking a look at the confusion matrices, we can clearly see that there has been a huge improvement between both feature sets performance regarding this class: a 500 % increase of accuracy, going from 26 errors to just 5. That is why on this computation from both sets performance the terrible performance gets smoothed by the latter.

4.2.2.3. Logistic Model Tree (LMT) classification

Performance using acoustic features based on Interspeech '09 (386 features)

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
113	0	1	4	10	0	0	**Anger**
0	67	4	1	0	6	3	**Boredom**
3	1	36	3	0	2	1	**Disgust**
5	2	2	49	3	4	3	**Fear_Anxiety**
16	0	2	4	49	0	0	**Happiness**
1	10	1	2	2	63	0	**Neutral**
0	9	1	0	0	0	52	**Sadness**

	Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness
TP Rate	0.883	0.827	0.783	0.721	0.69	0.797	0.839
FP Rate	0.061	0.048	0.022	0.03	0.032	0.026	0.015
Precision	0.819	0.753	0.766	0.778	0.766	0.84	0.881
ROC Area	0.971	0.894	0.882	0.915	0.939	0.927	0.91

Table 4.13: LMT: Confusion matrix and specific performance parameters for 'emobase2009' set

Performance using acoustic features based on Interspeech '10 (1430 features)

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
114	0	2	3	8	1	0	**Anger**
0	71	2	0	0	4	4	**Boredom**
1	1	38	2	0	2	2	**Disgust**
3	0	1	53	5	4	2	**Fear_Anxiety**
18	0	1	5	46	1	0	**Happiness**
1	7	0	3	0	66	2	**Neutral**
0	4	0	0	0	2	56	**Sadness**

	Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness
TP Rate	0.891	0.877	0.826	0.779	0.648	0.835	0.903
FP Rate	0.057	0.026	0.012	0.028	0.028	0.031	0.021
Precision	0.832	0.855	0.864	0.803	0.782	0.825	0.848
ROC Area	0.976	0.946	0.956	0.958	0.947	0.922	0.931

Table 4.14: LMT: Confusion matrix and specific performance parameters for '*emobase2010*' set

Performance using acoustic features based on specific 'emobase' (990 features)

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
112	0	1	3	12	0	0	**Anger**
0	67	2	1	0	8	3	**Boredom**
0	1	37	2	2	3	1	**Disgust**
5	0	0	57	4	1	1	**Fear_Anxiety**
19	0	0	4	45	2	1	**Happiness**
1	8	0	1	1	67	1	**Neutral**
0	8	2	0	0	0	52	**Sadness**

	Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness
TP Rate	0.875	0.827	0.804	0.838	0.634	0.848	0.839
FP Rate	0.061	0.037	0.01	0.024	0.041	0.031	0.015
Precision	0.875	0.798	0.881	0.838	0.703	0.827	0.881
ROC Area	0.978	0.918	0.969	0.976	0.94	0.914	0.889

Table 4.15: LMT: Confusion matrix and specific performance parameters for '*emobase*' set

Global performance comparison

Config. file	emobase09	emobase2010	emobase
Correctly Classified Instances	429 (80.1869 %)	444 (82.9907%)	437 (81.6822 %)
Incorrectly Classified Instances	106 (19.8131 %)	91 (17.0093 %)	98(18.3178 %)
Kappa statistic	0.7645	0.7979	0.7822
Computational time	2'29"567	11'20"117	7'15"662

Table 4.16: LMT: Overall parameters for the three attribute sets

The data classifications results with our third and final choice - LMT, belonging to the group of the functional trees - are very solid, performing at a maximum accuracy of 83 % and a minimum of 80,2 %. In fact, we achieve very good behaviour with almost every emotion class that we are dealing with, with the only notable exception regarding the 'happiness' class, specially with its mixed results with the 'anger' class. Besides these errors, the other two misclassification peaks appear when the instances from the 'neutral' and 'boredom' classes are to be classified.

That being said, the most remarkable counterpart regards the computational time necessary to obtain the classification results. Two and a half minutes for the lightest instance data set; far more than ten minutes for the heaviest one. This is undeniable linked to how the decision tree algorithms are constructed, demanding a high time-cost model building time. This is probably the main reason to which this family of classifiers are not used for emotion recognition whose aim is towards real-time, automatic detection, as pointed in 3.2.5.

Strictly talking about its performance, it is undeniable that is has a lot in common with the SMO results, although this time the best feature set is the largest, heaviest one 'emobase2010'. The best classified classes are 'disgust' and 'sadness', with valid performance over the 'neutral' and 'boredom'. However, quite damaging false positive (FP) values for the 'anger' class, specially linked with the false negative (FN) of the 'happiness' instances through all three sets (16+18+19), and vice-versa: 30 (10+8+12) false positive 'anger' values that fall under 'happiness' class once classified. The other remarkable misbehaviour affects the FP of the 'boredom' class (41), mainly derived by the 'neutral' and 'sadness' classes FN values of classes. We believe that due to its similar performance and strengths and weaknesses to the SMO, no further analysis is needed. We will proceed to plot the distribution of errors by classes considering the three runs at the same time.

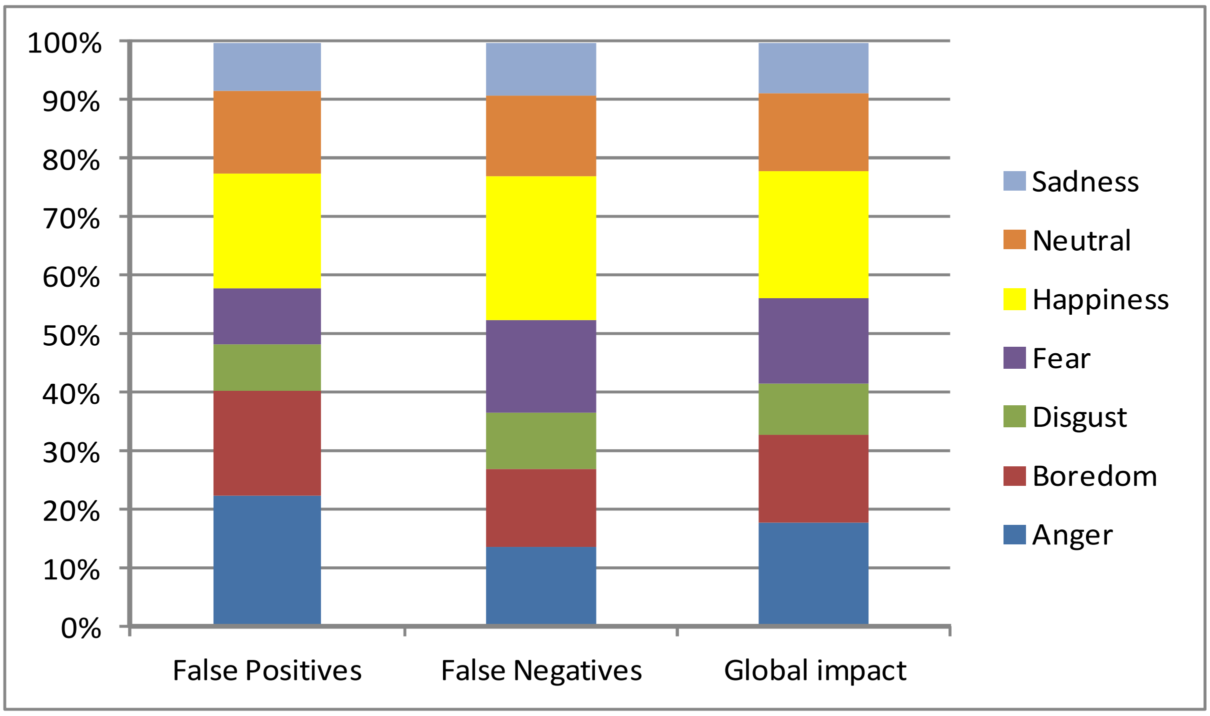


Figure 4.5: LMT: Error distribution per nature and average contribution (percentage-wise)

4.2.3. Conclusions

Up to this point, we have applied three nature distinct classifiers to three different audio feature sets that were extracted from one common source: our German emotion database. For each classifier we have derived some specific conclusions, and this final section of the second experiment will serve the purpose of deriving general conclusions putting the three data results together. In fact, through these specific conclusions we have seen that some common similarities exist, so next paragraphs will cover the resemblances as well as the main differences. But prior to this, the first step will be to choose which feature set has obtained the best performance taking into account all the scenarios. Under this restrain, we consider that it is the largest, heaviest and the most modern set: the labelled '*emobase2010'*. This is why tables 4.17, 4.18 and figures 4.6 and 4.7 (start at page 81) have been built, in order to obtain a deep and thorough analysis of the effectiveness of such feature set. There we will find detailed information of the performance carried out under the three classifiers, considering the absolute values of true positives (correct instances), false negatives and true negatives.

Regarding the likenesses between all classifiers performance, we can clearly see the most notable one. The already discussed mutual misclassification of instances originally from the 'anger' and 'happiness' classes, specially abrupt in the Naïve Bayes scenario where their mutual 'anger' FP and FN values even double compared to the other scenarios. This effect is also easily spotted in figure 4.7. Besides that, 'fear' clearly occupies the third spot when worst FP and FN values are concerned, always percentage wise. Another common characteristic is the best classified emotion, 'sadness', independently of the classifier at hand. It attains the best ratio of correctly classified instances through all three scenarios, specially high when the '*emobase2010*' set is under use (see figure 4.6). The class 'disgust' might mislead the observer, but remember the asymmetrical nature of the database where this specific class has only 46 instances: working with ratios and percentages is the correct way to determine which is the best classified class. However, as we will see, the mentioned class occupies the second rank of best performance.

Back in section 4.2.2.1, we sketched several ideas about these presumably more correlated emotions (anger-happiness-fear) and the most uncorrelated ones (sadness-disgust). Firstly, we talked about some degree of guilt due to the German language's both linguistic and paralinguistic specific characteristics. However, going deeper analysing the specific characteristics of the language are out of our scope. Secondly, such results would definitely show prove of how such emotions share an intrinsic correlation independent to the spoken language, and thus, independent to the used classifier. For instance, Anger and Happiness emotions have positive and strong Arousal and Dominance values, but have opposite Valence values (see figures 2.1 and 2.2). Fear emotion would even share more characteristics with the Anger emotion, with just a lower Arousal value.

On the other hand, Sadness, having negative values for Arousal, Valence and Dominance would be the best allocated in such 3D space. Going one step further, we can see by looking at the confusion matrices that the majority of errors originated by this class are FP ones (even doubling the FN in most cases, which are always very low). That means that it is effortless for the classifier to identify original classes of 'sadness' as 'sadness' classes, but it is more easily mistaken (although with best overall ratios) while assigning original instances from 'boredom' or 'fear' classes to the 'sadness' class. We could explain this phenomena by saying that they do occupy close subspaces, since these three would share a negative Arousal and Valence. Such common behaviour would mean that there is still some degree of uncertainty while automatically detect emotions that cannot be overcome, at least for now, since it is intrinsic to the correlation between the acoustic features extracted from audio signals. Changing the classifier would not solve the problem, at least for those that we have analysed, and most likely a different source language would not consist a notorious change.

Boarding the differences section, the first thing we have to acknowledge is the group of emotions that are notoriously differently classified when the various classifiers were at hand. Despite that the effectiveness of the naïve Bayes classifier is the worst of them, there exists some correlation among the final rank of performance per each emotion with SMO and LMT. However, there are some of them that do not comply this general rule. For instance, the 'neutral' class experiences a boost of misled instances under the naïve Bayes scenario, specially regarding the false positive value that increases until becoming the highest FP of the classifier. Another example of this implies the 'disgust' class and the lightest emotional feature set, '*emobase2009*', with 26 FP.

Thus, we believe that focusing on the analysis of the set '*emobase2010*' is a good choice since it contains all the mentioned above. In the following pages, we present the most important data from such feature set results. We will start by showing some overall performance values, then specifically jumping towards each class classification results to finish by showing the big picture that will help to better understand all the discussed during this experiment.

Emobase2010	**SMO**	**Naïve Bayes**	**LMT**
Correctly Classified Instances	461 (86.1682 %)	405 (75.0009 %)	444 (82.9907%)
Incorrectly Classified Instances	74 (13.8318 %)	130 (24.2991 %)	91 (17.0093 %)
Computational time	7”189	3”241	11'20”117

Table 4.17: General parameters for '*emobase2010*'

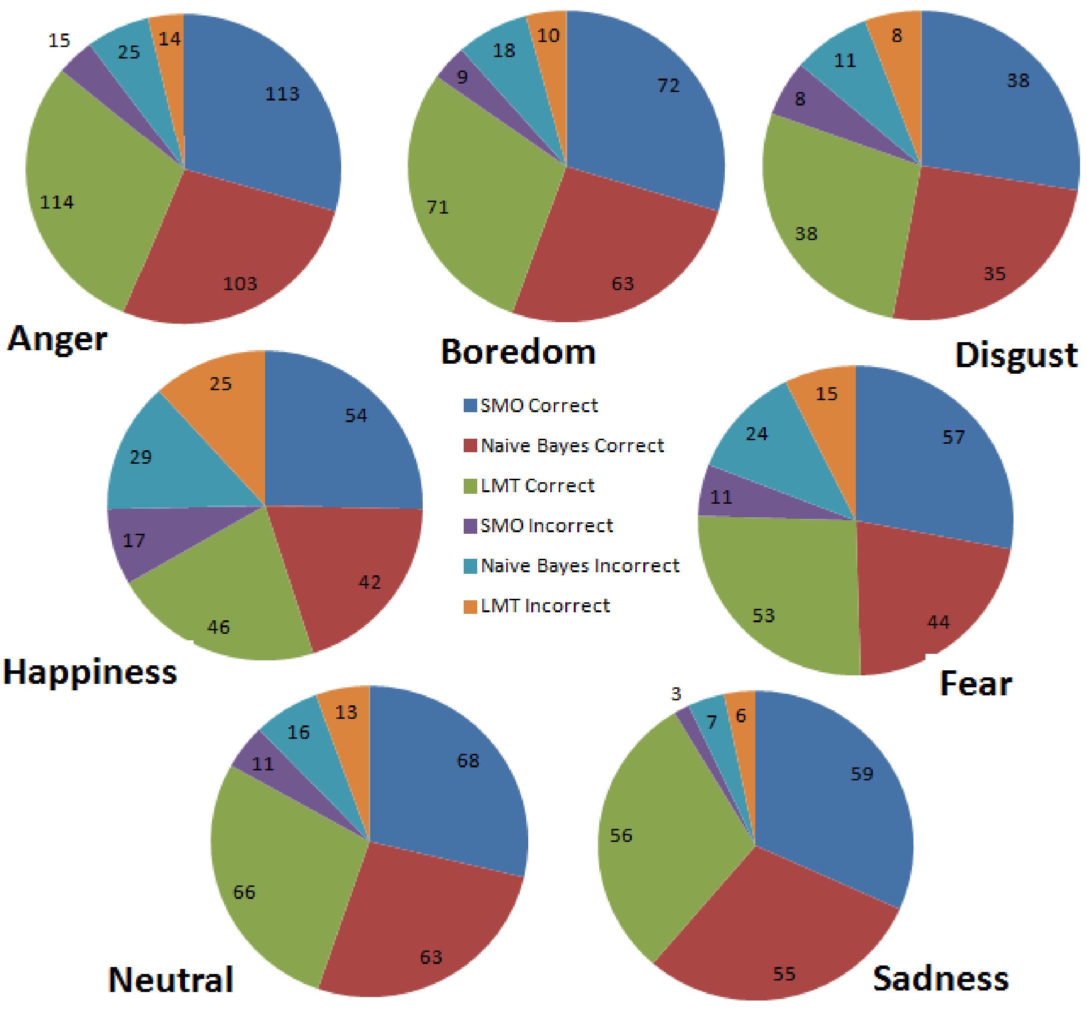


Figure 4.6: Specific instance distribution per each class for '*emobase2010*'

	SMO			Naïve Bayes			LMT		
	Correct	False Negatives	False Positives	Correct	False Negatives	False Positives	Correct	False Negatives	False Positives
Anger	113	15	19	103	25	24	114	14	23
Boredom	72	9	9	63	18	19	71	10	12
Disgust	38	8	2	35	11	5	38	8	6
Fear	57	11	11	44	24	22	53	15	13
Happiness	54	17	18	42	29	24	46	25	13
Neutral	68	11	10	63	16	27	66	13	14
Sadness	59	3	5	55	7	9	56	6	10

Table 4.18: Performance distribution for each class and classifier for '*emobase2010*' (absolute values)

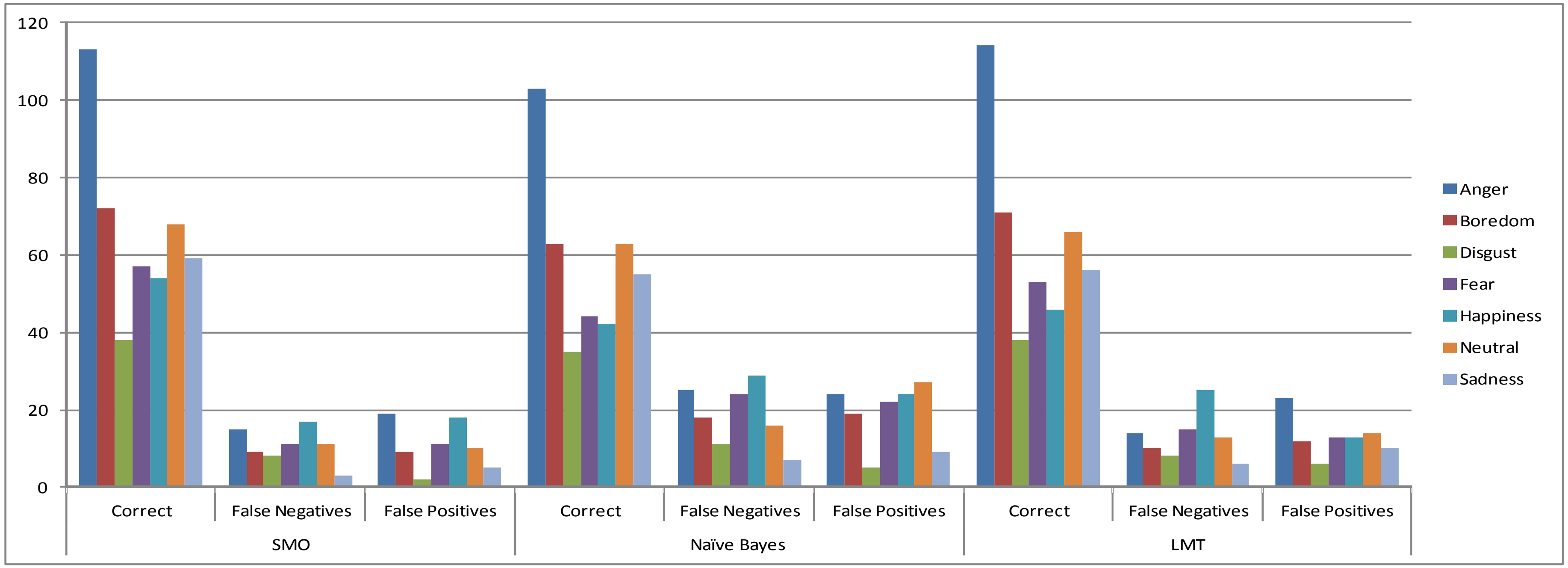


Figure 4.7: Graphical overview of correct and incorrect classified instance for each classifier and '*emobase2010*' feature set

4.3. Experiment 3

The next experiment will be focused on the feature selection techniques discussed on section 3.2.3. As a quick reminder, its main focus was to reduce the number of necessary acoustic features for emotion recognition while trying to , at least, attain a similar rate of performance and accuracy. As we have seen, our feature extraction sets work with quite a large number of features: 386 the lightest and 1430 the heaviest. Despite such difference in numbers, the performance results did not always justify the use of the most recent, up-to-date configuration set. For instance, results with the SMO or LMT classifiers where very similar and proved that the three sets did a very good job on the topic of affect recognition. Does that mean that, among all attributes, there are some that are unnecessary or some that may hinder the overall results? If so, could we be able to build an optimal list of features that provide similar or even better results than its original set? The goal of this experiment is to try to further advance on this topic and find an answer to such questions. Because, needless to say, working with a small, close to optimal list of acoustic features would result in lower computational and processing times, hence getting closer to real-time automatic affect recognition.

To this purpose, we decided to work with the same files that we created during the second experiment, that is the three feature extracted files belonging to each of three configuration sets of openSMILE. What will differ from the last experiment will be the use of WEKA. As graphically described in figure 4, our modus operandi will embody all the elements of this diagram. So, a new step will be introduced before the data classification algorithms: the feature selection procedure. As explained in section 3.3.4, we will be using the '*Filter*' button available within the '*Preprocess*' tab of Weka Explorer. The chosen classification methods once the new, and hopefully lightest, feature list has been computed are the already familiar SMO and naïve Bayes.

However, after several experiments, we saw that the original list of features got reduced to the order of one hundred in the worst case. So, despite the used feature set, the final list of attributes had greatly decreased. That is why we decided to work with the largest, most reliable set we have at our hands in order to ensure that our source of data was the optimal from the very beginning: '*emobase2010*' .

4.3.1. Principal Components Analysis (PCA)

We will begin by applying this transformation technique to our feature set, combining it with the Ranker search method – the only one allowed by Weka – and discuss the obtained results. After that, we will perform the already familiar classifiers SMO and naïve Bayes. Notice that PCA returns a modified list of attributes, where not only there will be a lot less than the original set, but they will be linear combinations between them and will have different values than the original attributes. This will not happen with our other feature selection technique, where the only notable change will be the decrease of the original features: the remaining ones where also present in the original set. A key parameter that needs to be manually specified is the '*variance coverage*', which will set the PC attributes to be accounted for the final list. Since this technique delivers an ordered list of transformed attributes by maximum to minimum variance, this parameter will do the function of threshold and discern those attributes that do not match its criteria. Setting the recommended value of 0.95 (95 %) delivered the best classification results, with a total of 49 attributes from the originally 1.430 and a total time of 172 seconds.

In the summary log created by Weka, all the steps necessary of such feature transformation technique can be seen. First, the whole correlation matrix is computed. Then, eigenvalues are computed and the threshold condition regarding the variance is applied, returning only the best 49 values. After that, their 49 eigenvectors are calculated using the 1.430 original attributes and are ordered accordingly. The final attributes are the result of a linear combination between 5 different features of the original set, thus obtaining the 49 final transformed features. Refer to appendix C.1 for the full list of these attributes.

<u>Performance of the feature set under SMO classifier</u>

Working with the SMV nature classifier delivered more than correct overall performance, with a slight decrease of 3% in overall performance with respect to the classification under the full set of features – experiment 2. The number of correctly classified instances is now of 445 (83,18 %), while the remaining 90 are errors. We will now present the confusion matrix where each element is the difference between this run and the one its counterpart from experiment two. That way it will be very easy to see which classes were more and less effectively guessed. The *'='* elements represent unchanged but non-zero values.

Anger	**Boredom**	**Disgust**	**Fear_Anxiety**	**Happiness**	**Neutral**	**Sadness**	
+2	0	+1	=	-3	0	0	**Anger**
0	**-4**	+2	+1	0	=	+1	**Boredom**
=	-1	**+2**	=	0	=	-1	**Disgust**
-1	=	0	**-3**	+2	+1	+1	**Fear_Anxiety**
+2	0	-1	+4	**-8**	+3	0	**Happiness**
0	+1	-1	-1	+1	**-1**	+1	**Neutral**
0	+3	0	+1	0	=	**-4**	**Sadness**

Table 4.19: SMO: Relative confusion matrix between original and PCA reduced set

The first worth mentioning is the worse performance of 5 of the 7 emotion classes, specially acute in terms of the 'happiness', 'boredom' and 'sadness', while classes 'anger' and 'disgust' present slightly better classification. Secondly, is the fact that for the first time in our research, we found a set of features that does not suffer so sharply of the mutual harmful effect between the classes 'anger' and 'happiness'. Relative numbers show us that up to 8 less true positives (TP) do not belong to the 'happiness' class any more, but the majority of them are not caused by ambiguity with the 'anger' class. Instead, this mutual harm effect has spread with the 'fear' (+4 FN) and 'neutral' (+3 FN) classes, which is even worse that our original problem with SMO performance. The Anger emotion, however, seems to do not suffer of such degradation, but even presents better classified instances, with 3 less FN that previously fell under the 'happiness' class. And third and finally, also interesting to see is that the best classified class over our whole research, Sadness, now presents worse numbers: -4 TP instances and the same FP than before, which makes the Anger emotion the best classified now.

Performance under the naïve Bayes classifier

The results with the Bayesian nature classifier deliver just one less correctly classified than in second experiment: a total of 404 correct, 131 incorrect. We can conclude that the overall effectiveness is the same, but with only 49 attributes (3,5 % of the original set). Let us see, however, if this overall data do not mask an undesired effect regarding some specific class classification.

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
+5	0	+2	-3	-3	-1	0	**Anger**
0	**-1**	+1	+2	0	-3	+1	**Boredom**
-1	=	**-1**	-1	+1	+1	+1	**Disgust**
+3	+1	=	**+1**	-2	-2	-1	**Fear_Anxiety**
+1	0	=	+1	**+1**	-3	0	**Happiness**
0	+1	0	=	+1	**-2**	0	**Neutral**
0	+2	0	+3	0	-1	**-4**	**Sadness**

Table 4.20: N-B: Relative confusion matrix between original and PCA selected set

By examining these matrix, we can say that 5 classes behave with no remarkable changes, while the anger and sadness classes show antagonistic results. The first, with 5 more TP instances, mainly due to better classification of their instances under the predicted 'fear' and 'happiness' (-6 in total) is the most benefited of the chosen PC features. But it presents 3 more FP instances, which is quite undesired. The second one, sadness, is quite the contrary: 4 more FN values, and 1 more FP value hinder its performance. A similar phenomenon occurred with our PCA and SMO classification.

4.3.2. Correlation-redundancy feature set analysis (CfsSubsetEval)

Here we will present the results of our research while applying this feature selection technique in combination with the three different search methods discussed in 3.2.3, and then evaluating them with SMO and Bayesian classifiers. Notice that Weka does not allow us to use the Ranker search option while applying this method.

4.3.2.1. Performance results with the *Best First* selection method

Applying this technique in the '*Filter*' button of the pre-process tab of Weka environment returned a list of 134 attributes – refer to appendix C.2 – , and it took 87 seconds to be completed. Approximately 90% of the original instances have been neglected, and they did not pass the threshold configured by the Best First selection. Taking a look at the list, we can see that almost all the MFCC, LSP, and Mel-Frequency spectrum bands are included. Very few pitch and energy related features such as PCM, F0 and voicing are also present. The rest of them is not taken into account. Now, it is time to test the performance of these roughly 10% instances by classifying them the way we are already familiar with.

Performance of the feature set under SMO classifier

The final results even exceed our prior expectations. The number of correctly classified instances is 461, equal to its analogue of the second experiment – that is when running the '*emobase2010*' set over the SMO classifier. For now, we may attribute this phenomena as a pure coincidence. But the data obtainable from the confusion matrix and the in-depth class performance shows us that the results are not exactly the same, but the errors and valid instances are slightly differently distributed among the seven classes. We now present the confusion matrix where each element is the difference between this run and the one from experiment two, in a similar way than before.

Anger	**Boredom**	**Disgust**	**Fear_Anxiety**	**Happiness**	**Neutral**	**Sadness**	
+3	0	+1	+2	-6	0	0	**Anger**
0	**+2**	+1	0	0	-2	-1	**Boredom**
-1	-1	**+3**	-1	0	=	=	**Disgust**
+2	-1	+2	**-3**	=	=	0	**Fear_Anxiety**
+1	0	=	+2	**-4**	+1	0	**Happiness**
+1	+1	+1	-2	=	**-2**	+1	**Neutral**
0	-2	0	0	0	+1	**+1**	**Sadness**

Table 4.21: SMO: Relative confusion matrix between original and '*Best First*' feature selected set

As expected, that way we can instantly see that 4 classes improved their classification, specially important for 'anger' and 'disgust', while the remaining 3 suffered an impact, specially 'happiness' and 'fear'. Notable curious is the double effect of, on one hand, the decrease of false negatives (FN) instances of the anger class that fell under 'happiness' (-6), while at the same time, the FN of happiness instances towards anger went up (+1). But, generally speaking, this is very valuable data since it means that these 10 % of attributes are sufficient to obtain the same performance as the whole list of 1.430 features. Hence, thinking about automatic real-time speech recognition, working with such a small list of attributes would represent a big step forward. It would only be necessary to list them from the very start and only extract those from our audio database.

Performance under the naïve Bayes classifier

A considerable increase of almost 5 % has been achieved in this scenario, with 20 more correct classified instances than the case of second experiment (425) and a global accuracy of 79.44 %. So, if we were very optimistic in the last case, now we find ourselves with even better results, over performing the other experiment results held with the original list of attributes. We will now proceed to present the relative confusion matrix, since we believe is an easy way to sketch valid conclusions.

Anger	**Boredom**	**Disgust**	**Fear_Anxiety**	**Happiness**	**Neutral**	**Sadness**	
+5	0	0	=	-4	-1	0	**Anger**
0	**+3**	-1	=	0	=	-2	**Boredom**
-1	-3	**+3**	=	+1	-1	-1	**Disgust**
=	=	+1	**+6**	=	-6	-1	**Fear_Anxiety**
+2	0	=	+1	**-1**	-1	0	**Happiness**
0	-2	0	-2	=	**+3**	+1	**Neutral**
0	-1	0	0	0	=	**+1**	**Sadness**

Table 4.22: N-B: Relative confusion matrix between original and '*Best First*' feature selected set

As we can see, this improvement of 5% could have not affected better the original confusion matrix: the positive tendency has spread through 6 of the 7 classes, with special insight with anger and fear true positive (TP) values (+5 and +6). only the 'happiness' class experiments a very slight hindering, since it loses one correctly classified instance that, as we could have predicted, it falls under the 'anger' class aggravating the mutual harm effect widely seen across our research. Similarly to what happened before, the FN anger instances that fell under the happiness class are also decreased here (-4). Besides this common fact, a specially good performance under this circumstances is the decrease of false positives (FP) throughout all the matrix, particularly affecting the predicted 'neutral' (-9) and 'boredom' (-6) classes which experiment a positive change.

4.3.2.2. Performance with Linear Forward Selection (LFS) search method

This search method allowed us to manually select the *k* number of attributes that would be considered as the 'top' ones and through which the different sub-sets of features are compared, in order to build the final list of attributes. By a trial and error methodology, we fixed this number to 200 (about 15% of the whole attributes). With this configuration, and after 6 seconds, we obtained the best performance results, and a total of 66 instances presented in appendix C.3. The main differences regarding this final list are the number of MFCC and Mel-Frequency band related features, now about one third of the total number previously used by *Best First*.

Performance of the feature set under SMO classifier

The total of true positive ascends to 456, representing an overall accuracy of 85,25% and 79 incorrectly classified instances. Less than 1 % effectiveness drop with respect to the previous search method, but half the attributes needed to attain such results.

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
+9	0	0	=	-9	0	0	**Anger**
0	**+2**	0	0	0	=	-2	**Boredom**
-1	=	**+1**	=	0	=	=	**Disgust**
+3	-1	+2	**-6**	+1	=	+1	**Fear_Anxiety**
+9	0	=	+2	**-11**	0	0	**Happiness**
+1	-2	-1	=	=	**+1**	+1	**Neutral**
0	+1	0	0	0	=	**-1**	**Sadness**

Table 4.23: SMO: Relative confusion matrix between original and '*LFS*' feature selected set

Unlike before, LFS technique grants us a not that optimistic scenario once we have deeply analysed its performance. In fact, despite the very good overall results, we can now see that the pair of 'anger' and 'happiness' classes undergo an undesired transformation. On one hand, the 'anger' class gets benefited for these 9 FN instances that are now their TP. But on the other hand, 11 more FN of the 'happiness' class, of which 9 now fell under the predicted 'anger' class, lower drastically the performance of the mentioned class. In other words, we get a worse predicted anger class (much more FP instances) at expenses of a better guessed 'happiness' class. It is quite clear that several vital instances that were key towards correctly defining these two classes have been neglected by LFS. Besides that, the 'fear' class also experiences notable changes, while the remaining 4 classes are slightly better classified.

Performance of the feature set under naïve Bayes classifier

Working with the Bayesian nature classifier brings us similar results than before, achieving also 1% decrease in overall performance (418 correct instances, 117 incorrect) when compared with the Bayesian and best first search method, but an increase of 3 % when comparing results with its analogue of the second experiment (4.2.2.2).

Anger	Boredom	Disgust	Fear_Anxiety	Happiness	Neutral	Sadness	
+3	0	0	-2	=	-1	0	**Anger**
0	**+2**	+1	-1	0	=	-2	**Boredom**
-1	-2	**+2**	=	+1	+1	-1	**Disgust**
+1	+1	=	**+2**	+1	-5	=	**Fear_Anxiety**
+3	0	-1	-2	**+2**	-2	0	**Happiness**
0	=	0	-1	+1	**=**	0	**Neutral**
0	-1	0	0	0	-1	**+2**	**Sadness**

Table 4.24: N-B: Relative confusion matrix between original and '*LFS*' feature selected set

In this scenario, the use of LFS clearly proves to be a better choice. We can see a specific increase of each class correct instances, almost equally distributed among them. Only the 'happiness' and 'anger' class experiment an increase in their FP values, but nothing as important as in our last scenario. Of course, we don't have to forget that we are dealing with relative numbers, that means that whatever lack of performance intrinsic of this Bayesian classifier is maintained here. For instance, the FP values of the 'anger' class sum up to 28. what we mean by that is that, despite clearly improving our results in experiment 2, it still shows better overall performance than the other classifier.

4.3.2.3. Performance with Scatter Search method

The third and final search method that we decided to use performs an scatter search among the given set of data. The only configuring parameter worth mentioning is the labelled as '*combination*', with two available options: *greedy* and *reduced greedy*. We tried them both, referred as SS-GC and SS-RGC from this point, and two main differences were observed. As excepted, the computational time of the SS-GC was much bigger, more than five hours of data processed by Weka, but presented slightly better performance results (4,1 %).On the other hand, the reduced greedy combination took 124 seconds to be finished. The variance in time is due to the very core of each approach. While in SS each iteration of the combination method with a new subset of features generates new attributes that are to be added to the list of solutions, the RGC only adds the features with the highest accuracy performance from those that do not belong to the solution list. The GC compares them with the whole rest of attributes, which derives into a huge extra number of computations.

Since the main objective of this experiment is to retrieve the optimal list of features, we decided that time issues were not a problem here. That is why we present the results only with the greedy combination approach, which returned a total of 87 attributes fully listed under appendix C.4.

SS-GC feature set classified under SMO

The overall performance has been of 83,89%, with 449 correctly classified instances. Also experimenting a slight decrease in comparison with the full attribute set of 2,2 %. The RGC approach returned a drop of 6,4 % global accuracy. If we take a look at the specific parameters per class:

Anger	**Boredom**	**Disgust**	**Fear_Anxiety**	**Happiness**	**Neutral**	**Sadness**	
+1	0	0	-1	=	0	0	**Anger**
0	**=**	+2	=	0	-2	=	**Boredom**
-1	=	**-1**	=	+1	+1	=	**Disgust**
+1	=	-2	**-1**	-2	+2	+1	**Fear_Anxiety**
+8	0	-1	=	**-7**	0	0	**Happiness**
+2	+3	+1	0	-1	**-5**	0	**Neutral**
0	-1	-1	0	0	0	**+2**	**Sadness**

Table 4.25: SMO: Relative confusion matrix between original and '*GC-SS*' feature selected set

Very equally distributed differences under 5 of the 7 classes, and objectively worse performance under 'happiness' and 'neutral' classes. In particular, an increase of both FP and FN related to the latter class, while a the first one experiences an increase of its FN values but a decrease of its FP. As well as our other experiments, satisfying results overall. The only remarkable counterpart is the effect mentioned in the last lines.

SS-GC feature set classified under Bayesian logic

A total of 420 instances were correctly classified, 16 more than the performance with the full feature set which translates into an overall increase of effectiveness about 3 %. If we take a look into the usual relative confusion matrix, we will quickly see how are now each instance distributed.

Anger	**Boredom**	**Disgust**	**Fear_Anxiety**	**Happiness**	**Neutral**	**Sadness**	
+6	0	0	-4	-1	-1	0	**Anger**
0	**+2**	-1	=	0	-1	-1	**Boredom**
-2	-2	**+2**	-2	+3	+1	=	**Disgust**
-2	+1	=	**+1**	+1	-1	=	**Fear_Anxiety**
+1	0	-2	=	**+4**	-3	0	**Happiness**
0	+1	-1	+1	-1	**=**	0	**Neutral**
0	-4	0	0	0	+3	**+1**	**Sadness**

Table 4.26: N-B: Relative confusion matrix between original and '*GC-SS*' feature selected set

The overall increase of performance is reflected into an increase of TP values for each class, with the exception of the 'neutral' which retains the same orginal value but with less FP now. We can also see an specially sharp improvement of the mutual harmful effect between the Anger and Happiness emotions.

4.3.3. Conclusions

Up to this point, we have been very thorough with our objective of further analysing the effect of applying feature selection techniques to our original and large set of attributes extracted through the '*emobase2010*' configuration file. To that purpose, we have established two different techniques present in Weka package, PCA and CfsSubsetEval, as well as combining them with several attribute search methods that would finally return the definite and greatly reduced list of features. Once we had them in our hands, we applied two already familiar classifiers, SMO – SVM natured – and naïve Bayesian – obeying Bayesian logic , thus getting our end results that we could deeply analyse.

In the different sections of this third experiment, we have stated specific comments on their performance, but now it is time to treat them as a whole and deduce general conclusions that might answer the rhetorical questions that we wrote at the beginning of this section.

We will begin by presenting the overall performance parameters from all our experimentation in the table below.

Technique		**PCA**		**Cfs Subset Eval**					
Search method		**Ranker**		**Best first**		**LFS**		**Scatter Search**	
Nº features (% original set)		49 (3,5%)		134 (9,4%)		66 (4,6 %)		87 (6,1 %)	
Classifier		SMO	Bayes	SMO	Bayes	SMO	Bayes	SMO	Bayes
Overall accuracy (%)		83,18	75,51	86,17	79,44	85,25	78,13	83,89	77,52
Relative performance		-3%	+0,5%	=	+4,4%	-0,9 %	+3,1%	-2,2%	+3%
Relative True Positive Values	**Anger**	**+2**	**+5**	**+3**	**+5**	**+9**	**+3**	**+1**	**+6**
	Boredom	-4	-1	**+2**	**+3**	**+2**	**+2**	**0**	**+2**
	Disgust	**+2**	-1	**+3**	**+3**	**+1**	**+2**	-1	**+2**
	Fear	-3	**+1**	-3	**+6**	-6	**+2**	-1	**+1**
	Happiness	-8	**+1**	-4	-1	-11	**+2**	-7	**+4**
	Neutral	-1	-2	-2	**+3**	**+1**	**0**	-5	**0**
	Sadness	-4	-4	**+1**	**+1**	-1	**+2**	-2	**+1**

Table 4.27: Summary of key values retrieved during experimentation, based on the set '*emobase2010*'

Where the relative TP values follows our usual scheme while showing the final data analysis. Once again, we are showing the difference between specific TP values from the classification results using the full feature set and those obtained while working with the reduced ones. At the same time, we have highlighted all results that show a non deterioration of the original performance.

The first objective conclusion that we can extract is that applying feature selection techniques is a more than a justified key matter among researchers, and we can now get a glimpse on the reasons why so much human power and resources are being derived into this subject. We have seen that, in the worst case when just counting the absolute number of attributes, we ended up with just 9 % of the original features while obtaining equal and even better performance results than with the original set. This is the case of *CfsSubsetEval+Best First* technique, undoubtedly the best one result-wise. But even when working with just the 3 % of the original attribute set, the effectiveness only decayed a 3% in the worst case; this translates into 16 less correctly classified instances.

Needless to say, directly extracting the selected list of attributes from the audio signals would prove a huge leap forward into automatic real-time affect recognition, since the processing times are now almost negligible. However, we must not consider this viable when working with the PCA, since the final list contains transformed parameters that were not initially contemplated in the original set. Thus, directly extracting such features from audio signals is meaningless, since they would be dependent of the source audio file used. In other words, unlike in our other feature selection method, this time we do not have our hands into an invariable list of selected attributes, but rather one that has to be computed from an already extracted set of features.

The second conclusion would involve the specific overall performance of each of our classifiers. On the first hand, the improvement in the performance of the Bayesian classifier under each of our experimentation, with a maximum of 4,4 % (24 more correct instances) when using the best feature selection list result-wise: *CfsSubsetEval+BestFirst.* In fact, all three classification results regarding this selection technique are more than satisfying, being the other two (LFS and SS-GC) of approximately a 3 % increase. The same cannot be said when running the PCA, where the improvement over the original was practically negligible (+0,5%). On the other hand, we are faced with not that optimistic results with the data regarding the overall performance of selected feature sets and SMO classification. Worse results experimented under *PCA+Ranker* feature set (-3 %), while the relatively best performance was achieved under the *CfsSubsetEval+BestFirst,* once again. Here, the instances were slightly differently distributed, but ended up with the same total TP values: same overall accuracy results. LFS also performed very good, with a relative difference of -0,9% while SS-GC ended up quite near to the overall performance of PCA.

Our third and final conclusion from this experimentation will be focused on the specific class behaviour (classification) under the mentioned selection techniques. Figure 18, next page, accounts all this information. For instance, we can see how the Anger emotion is the only one that experiments better classification results regardless of the feature selection technique applied. The opposite behaviour is experimented by the 'happiness' class, specially acute when the SMO classifier is applied to the different feature sets. In fact, the mostly bipolar results are achieved under *CfsSubsetEval+LFS* method, where a very sharp improvement of 'anger' instances classification is shaded by a proportional worsening of the 'happiness' class. The 'boredom' and 'disgust' classes also attain very good overall results when the correlation-redundancy technique is applied, although not that good under PCA. The remaining three classes obtain mixed results, showing good improvement under specific feature list, but resulting harmed under others. Aside from that, and as we sketched in the paragraph above, it is easy to see the improvement experienced by Bayesian classification, even more if we discard the PCA related results. All classes experiment a positive change, except just one instance of 'happiness' with the *Best First* search method.

To sum up, we would conclude that our chosen feature selection techniques have proven to be very useful, ranging from a -3% to a +4,4% relative overall performance while working with less than 10 % of attributes from the original set. Thus, we have showed objective data that proves that there is a vast number of features that are highly redundant or even hinder overall results, and we have also obtained a more close to an optimal list of attributes from which to work in the future having an automatic, real-time, affective detection HCI. However, we must not forget the limitations of our emotion database, with only seven emotion classes represented. Ours is just theoretic data from an idealistic scenario where we only experience six emotive states and an emotionless one, which is very far from a realistic approach.

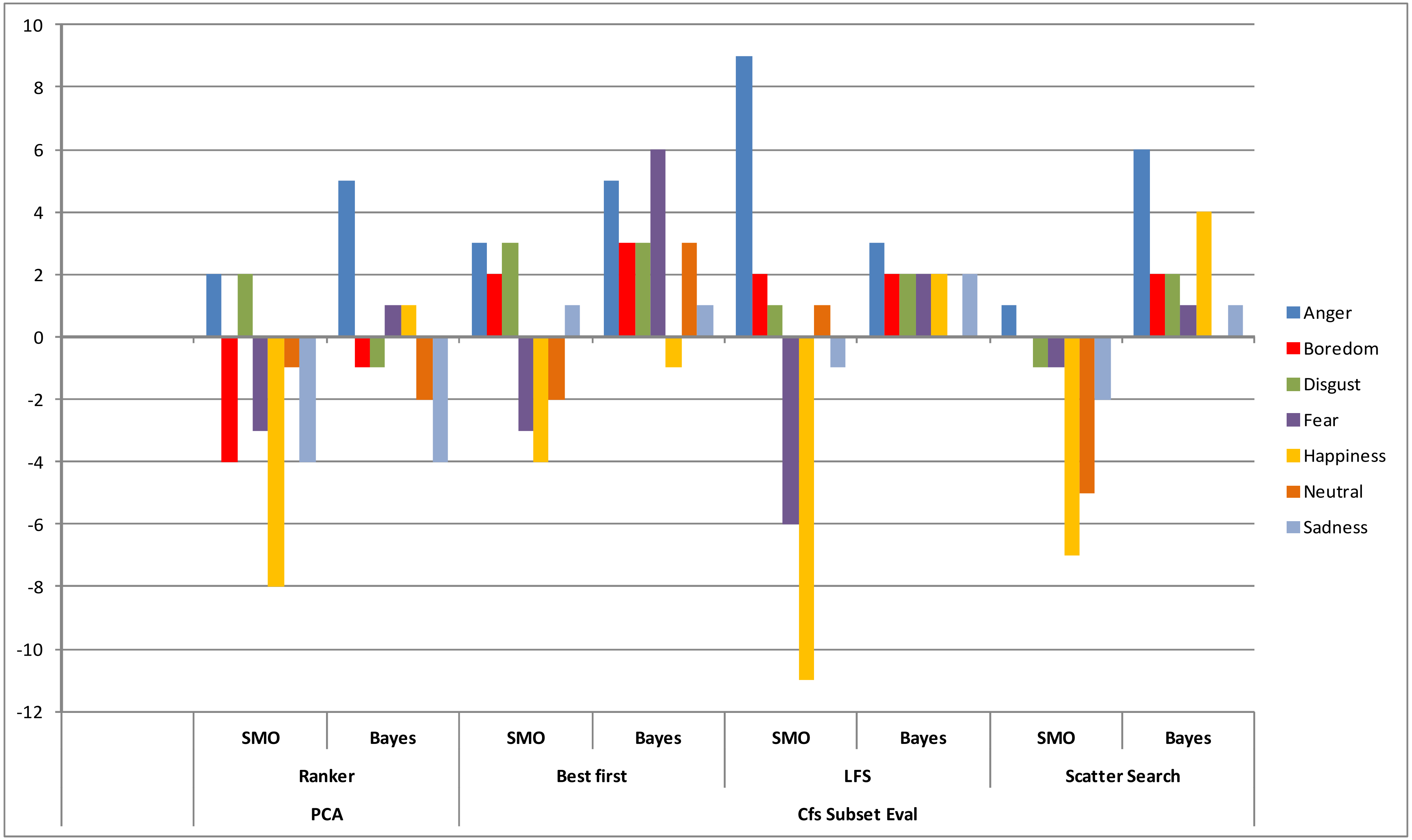


Figure 4.8: Relative values of specific emotion class performance: original '*emobase2010*' set and reduce feature set

5. DISCUSSION AND FINAL CONCLUSIONS

5.1. Summary

This thesis has provided a theoretical survey on the field of emotion recognition from speech signals supported by a comprehensive body of empirical data fruit of our experiments. First, we have provided a close to the current state-of-the-art general review, along with an extensive description of all the elements that have played an important role in our research. Second, the retrieved data from every experiment has been evaluated and specific conclusions have been derived.

Discussion on first experiment

Regarding the first experiment, it was initially intended to serve as a reference point that, at the same time, helped to better understand the procedure followed and the key parameters that our data-mining software returned. The basic difference among other experiments had to do with the distribution of the emotion instances retrieved from our database. We already knew that it was highly asymmetrical, with notable difference between the total number of instances (or audio files) from each emotion; hence, we decided to symmetrize it by manually selecting 40 files from each affection - which was an easier procedure. The classification results turned up to be very positive, proving that the SMO was a very suitable classifier. In particular, the 'sadness', 'anger' and 'disgust' classes obtained the best results, while the 'happiness' the worst. Specific peculiarities regarding the correlation between the emotions were also detected. For instance, the high number of false negatives within the 'happiness' class, specially acute effect when misclassifying them as instances from either 'fear' or 'anger' classes. Some degree of misleading effect due to the 'neutral' class was also seen, deriving in big values of either false positives and false negatives related to this class.

Discussion on the second experiment

The main idea behind the second experiment, the most extensive one, was to obtain a wide and rich group of data that could be used to obtain rather objective conclusions regarding both the intrinsic relationship between the used emotions and the performance of the chosen classifiers. To the purpose of obtaining a wide range of results, the whole German database was used, setting a total of 535 instances. Our extraction tool would then deduce three different feature sets based on three different configuration files used. Once these attribute sets were created, they were ran under three nature distinct classifiers, returning very interesting data from each combination. For instance, general misclassification between the 'anger', 'happiness' and 'fear' classes emerged again (specially high between the first two), proving their high intrinsic correlation, big enough to confuse the three classifiers. On the other hand, the classes 'sadness' and 'disgust' obtained the best general results, similarly to what happened in our first experiment.

It was also easy to identify which classifier proved to be the most worthy of them: the SMO. Followed by LMT, though with the huge counterpart of having, by far, the biggest processing time of them. A fatal disadvantage since we have always had in our mind the objective of getting closer to a real-time, automatic emotion recognition system. The Bayesian classifier attained mixed results, showing a more than acceptable performance under the '*emobase2010*' feature set, but mediocre effectiveness validating the data from the lightest feature set. This variation contributed to proof the improvement on the research field in one year lapse. Much worse was its performance with the '*emobase'* achieving results similar to what a random decision system would obtain (very close to 50% accuracy). Worth mentioning is also the most modern, but also heaviest, feature set, '*emobase2010*'. Best overall classification results were obtained when working under this set, and this was the main reason why such configuration file would be later used as the basis of the third and final experiment.

Discussion on the third experiment

So, boarding the third and final experiment comments, we could say that its objective was to test several feature selection techniques that had already been discussed over the theoretical survey of the field. With undeniable benefits described before hand, we managed to retrieve data that showed that it was possible to achieve equal or similar performance than with the use of the original feature sets was the true goal of this experiment. Since now it was important to start with the best possible original attribute set, we chose to work with the '*emobase2010*'. As for the classification stage, we chose to work with SMO and naïve Bayes classifiers, discarding the LMT due to its incompatibility with theoretical real-time operations. Our results turned out to be rather positive, with a range of relative effectiveness that did not exceed a -3% in the worse case, and achieved an increase of +4,4% in the best case. Without forgetting that such results were obtained with feature lists that weighted less than 10% of the original set, being even less than 4% under a specific technique. Worth mentioning was the positive effect of such selection techniques and the Bayesian classifier results, ranging up to a +4,4% accuracy leap. On the other hand, the SMO working under the reduced featured lists only managed to perform equally in the best scenario, with a drop of 3,1% in the worst case. However, we were always achieving overall accuracies above 80%, which may be already be practically non-improvable.

5.2. Problems encountered

Gaining access to a reliable database of emotion supposed our first notorious complication. Needless to say, all our research would have been meaningless without at least one, and though we found information about the most widely used [36] , the vast majority of them were not allowed to be publicly accessed. We even tried to add the Maribor speech database, but it ended up being a dead end. Finally, only the Berlin emotion database was our source of affective data.

Being unable to run the debugger function from QtOctave represented another important issue. The implementation of the scrip code needed to be very accurate since we were calling Octave from a scrip file and, even more, running openSMILE from there. The correct setting of the long command line that called and configured our feature extraction tool presented the biggest problem of those related to our impossibility to automatically debug the scrip code. Other programming related issues, like successfully controlling iterative loops, or keeping all variables within a correct range, were also amplified due to the mentioned lack of debugging. That being said, the cause of this problem is most likely related with the installation of the program source code via Ubuntu, or some software incompatibility.

5.3. Further research topics

Different environments

It would have been very interesting to be able to extend our results and experimental processes to databases recorded in other languages. Not only that, but also to have a wider range of emotional expression (total number of speakers, sex, age differences) from such databases: Berlin Emotional Database only comprises ten different speakers, therefore the range is rather thin. Comparing performance results between different language databases would have given us key information about the degree of universality of our acoustic features.

In fact, it would have been even more challenging to work with natural emotion speech databases. We have already discussed these non-acted database through our theoretical survey, and we have pointed that they are the final target of emotion recognition systems. Characteristics like having to deal with an unbounded range of emotions, or even a much greater degree of variation to express each of them, make them much harder to analyse than acted speech. Even more, aside from these intrinsic factors, the samples of natural speech are much more contaminated by noise [37] than the ones generated under laboratory conditions, which is our case.

Optimisation of the feature set

We mentioned that we were dealing with a rather close to the current state-of-the-art attribute set, the most recent being based on the '*Interspeech 2010 paralinguistic challenge*' from September of that year. We began the experimentation roughly one year after that, and we present the results approximately five months later. It is undeniable that it conforms a more than valid reference set of features, but it is also worth noticing that no update belonging to the year 2011 has been uploaded in the openSMILE developers website. It would have been interesting to prove the performance leap during this year, in a similar way that we have seen the improvements done when comparing the results from 2009 feature set with the mentioned one from late 2010.

That being said, another comment will be made regarding the feature selection techniques that we were able to work with. We were indeed limited to those that were included in the Weka package, but during our survey we mention other widely used algorithms, like the already mentioned Linear Discriminant Analysis (LDA, see 3.2.4) or Sequential Floating Forward Search (SFFS) [38] [39], that we have not been able to test since they are not included in Weka.

6. LITERATURE

[1] R. Nakatsu, J. Nicholson, and N. Tosa, , "Emotion recognition and its application to computer agents with spontaneous interactive capabilities", 2000

[2] C.Min Lee, S. Narayanan, "Toward Detecting Emotions in Spoken Dialogs", 2005

[3] C. Williams and K. Stevens, "Emotions and Speech: Some Acoustic Correlates", 1972

[4] M. Suwa, N. Sugie, K. Fujimora, "A Preliminary Note on Pattern Recognition of Human Emotional Expression", 1978

[5] L. Chen, T.S. Huang, T. Miyasato, R. Nakatsu, "Multi-modal Human Emotion / Expression Recognition", 1998

[6] J.A. Russell, J. Bachorowski, J. Fernandez-Dols , "Facial and Vocal Expressions of Emotion", 2003

[7] P. Ekman, "Emotions in the Human Faces ", 1982

[8] James A. Russell, A circumplex model of affect, 1980

[9] J.A. Russell, J. Bachorowski, J. Fernandez-Dols, "Facial and Vocal Expressions of Emotion", 2003

[10] O.W. Kwon, K. Chan, J. Hao, T.W. Lee, "Emotion Recognition by Speech Signals", 2003

[11] Prof. Sendlmeier, Dr. A. Bartels, Dr. M. Rolfes , Dr. F. Burkhardt, Web EMO-db, http://pascal.kgw.tu-berlin.de/emodb/index-1024.html , accessed 2/11/2011

[12] F. Burkhardt, A. Paeschke, M. Rolfes, W. Sendlmeier, B. Weiss, "A Database of German Emotional Speech", 2001

[13] D. Ververidis, C. Kotropoulos, "Emotional speech recognition: Resources, features, and methods", 2006

[14] Z. Hachkar, B. Mounir, A. Farchi, , "Comparison of MFCC and PLP Parametrization in pattern recognition"

[15] B.Schuller, A. Batliner, S. Steidl, D. Seppi, "Recognising realistic emotions and affect in speech: State of the artand lessons learnt from the first challenge", 2011

[16] S. Wang; X. Ling; F. Zhang; J. Tong, "Speech Emotion Recognition Based on Principal Component Analysis and Back Propagation Neural Network", 2010

[17] , Hill-climbing techniques, http://en.wikipedia.org/wiki/Hill_climbing, accessed 5/3/2012

[18] M. Gütlein, E. Frank, M. Hall, A. Karwath, "Large-scale attribute selection using wrappers",

[19] F. García, M. García, B. Melián, J. A. Moreno, J. M. Moreno-Vega, "Solving feature subset selection problem by a Parallel Scatter Search", 2004

[20] N. Landwher, M. Hall, E. Frank, "Logistic Model Trees (LMT)", 2004

[21] J. C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines", 1998

[22] P. Refailzadeh, L. Tang, H. Liu, "Cross-Validation", 2008

[23] "Performance Measures for Machine Learning: Confusion Matrix and ROC curve",

[24] F. Eyben, M. Wöllmer, B. Schuller , "openSMILE - The Munich Versatile and Fast Open-Source Audio Feature Extractor", 2010

[25] F. Eyben, M. Woellmer, B. Schuller, openSMILE Web site, http://opensmile.sourceforge.net/ , accessed 28/10/2011

[26] F. Eyben, M. Woellmer, B. Schuller, "openSMILE book v.1.0.1", 2010

[27] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, "HTK Book", 2000 (v3)

[28] A. Batliner, S. Steidl, B.Schuller, D. Seppi, K. Laskowski,T. Vogt, L. Devillers, L. Vidrascu,N.Amir, L. Kessous, V.Aharonson, "Combining Efforts for Improving Automatic Classification of Emotional User States", 2006

[29] J. Psutka, L. Müller, J.V. Psutka, "Comparison of MFCC and PLP parametrizations in the speaker speech recognition task", 2001

[30] B. Schuller, S. Steidl, A. Batliner, F. Burkhardt, L. Devillers, C. Müller, S. Narayanan, "The INTERSPEECH 2010 Paralinguistic Challenge", 2010

[31] B. Schuller, S. Steidl, A. Batliner, "The INTERSPEECH 2009 Emotion Challenge", 2009

[32] GNU OCTAVE, (v.3.4.3), http://www.gnu.org/software/octave/ , accessed 5/11/2011

[33] P. L. Lucas, QTOCTAVE: A front-end for Octave, 2006-2011, http://qtoctave.wordpress.com/ , accessed 15/11/2011

[34] WEKA 3: Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/ , accessed at 21/10/2011

[35] J. Davis, M. Goadrich, "The Relationship Between Precision-Recall and ROC Curves", 2006

[36] D. Ververidis C. Kotropoulos, "A State of the Art Review on Emotional Speech Databases", 2006

[37] D. Neiberg, K. Elenius, I. Karlsson, K. Laskowski, "Emotion Recognition in Spontaneous Speech", 2006

[38] P. Pudil, J. Novovicova, J. Kittler, "Floating search methods in feature selection", 1994

[39] D. Ververidis, C. Kotropoulos, "Emotional Speech ClassificationUsing Gaussian Mixture Models and the SequentialFloating Forward Selection Algorithm", 2005

[40] Boersma, Paul & Weenink, David , "Praat: doing phonetics by computer ", 2001-2012, http://www.fon.hum.uva.nl/praat/ , accessed 17/03/2012

7. APPENDIX

Appendix A: Display of specific characteristics from our audio files

We selected one male speaker (26 years) from our database reading the same sentence with each of the database's emotions. The sentence is:

"Ich will das eben wegbringen und dann mit Karl was trinken gehen."

Translation:

"I will just discard this and then go for a drink with Karl."

The goal of this sub-section is to get a graphical view of several of the acoustic features that our software extracting tool is using. Seven plots will be presented in the following pages, each of them with specific information about each emotion (6 basic emotion plus the neutral state). Everything has been done with Praat software tool [40].The heading information of each graph is common:

- Upper plot is the time (seconds) VS amplitude (normalized) form of the audio file. It is immediate to identity the voiced and unvoiced segments.
- Lower plot is the spectrogram (time VS frequency – 0 to 5 kHz) of the audio file, in a grey-scale colour. The darkest parts show higher density of energy. Sharing the same graph zone, two more parameters are showed:
    - The yellow continuous waveform corresponds to the intensity, presented on a logarithmic scale. Needless to say, the comments of such curves will always be referred to the voiced segments.
    - The blue discontinuous curve is the pitch estimation of the audio file. The discontinuities are linked to the unvoiced segments of the file (small pauses between words or clauses) since no value is returned by the Praat algorithm when in such segments
- Since time is common in both X-axis plots, it is shown at lowest part of the graph (labelled as 'visible part'). As for Y-axis scales, the spectrogram one is shown on the left side of its corresponding plot. The pitch frequency values are shown in the rightmost part (75 to 500 Hz), and the intensity values (50 to 100 dB) just left of the pitch scale.

Anger

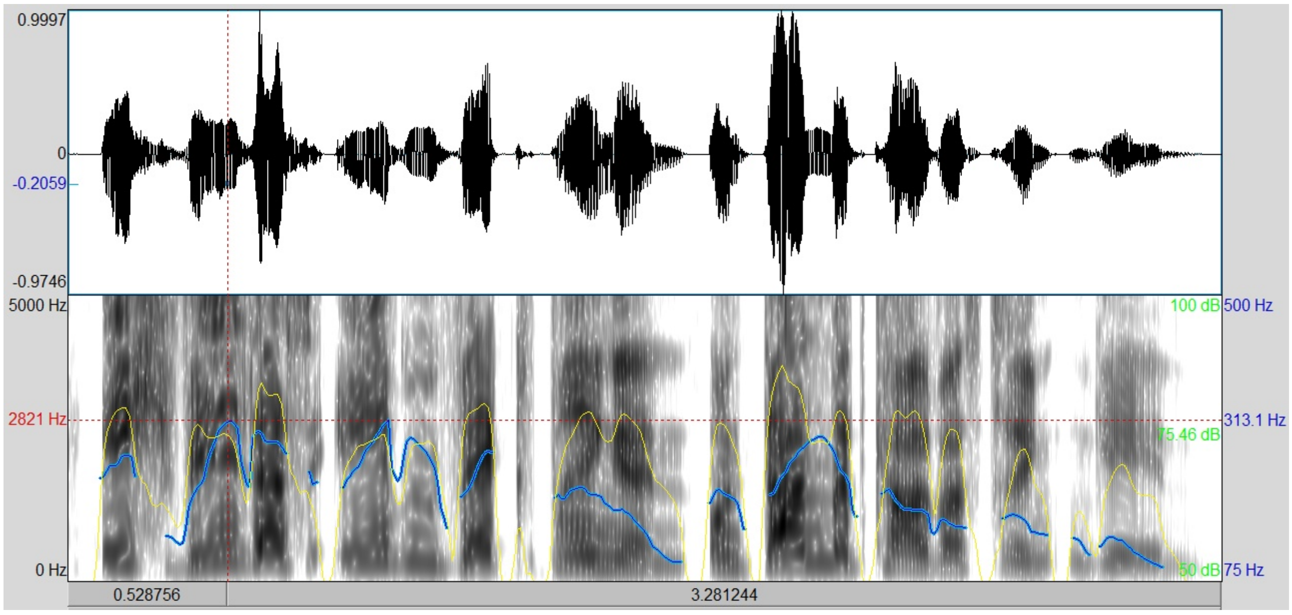


Figure A.1: Anger class acoustic features

- High variance of pitch, with its maximum peak of 313 Hz (see specific point in figure A.1) and several relative maximums near to that high frequency value. The overall estimation presents the highest frequency values for pitch, as we will see.
- Abrupt changes of intensity during the voiced segments of the audio sentence, with multiple relative maximums.

Boredom

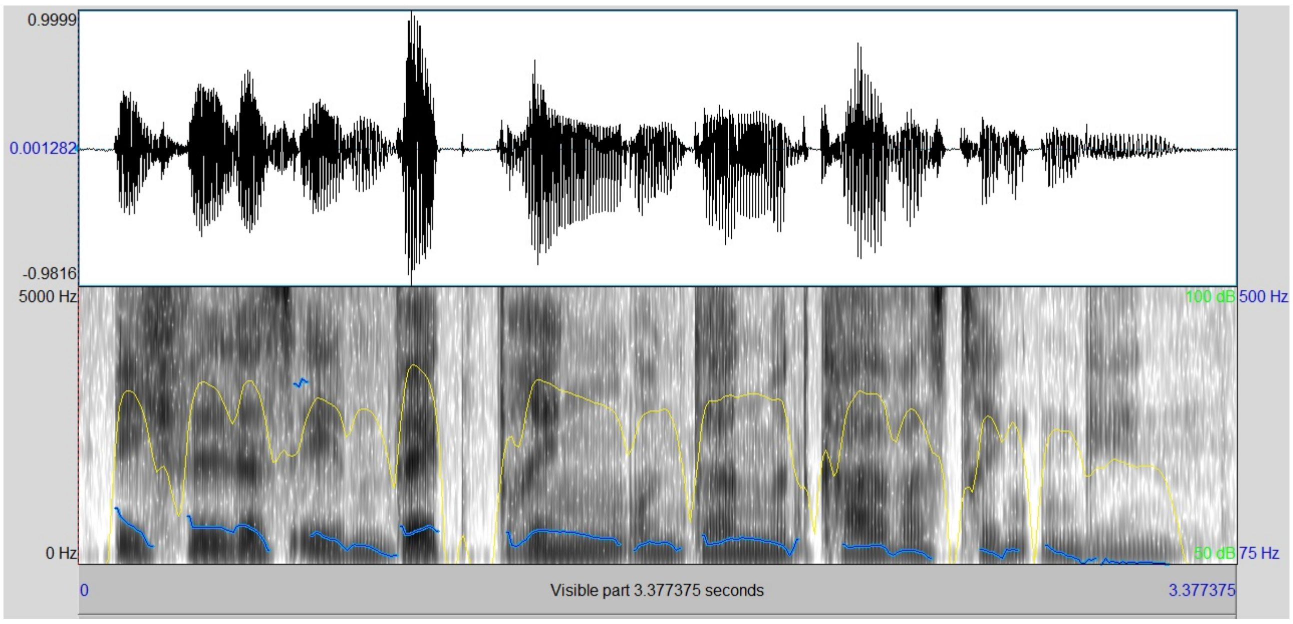


Figure A.2: Boredom class acoustic features

- Much more monotonous, low frequency pitch values that do not exceed 120 Hz.
- The intensity wave has more smooth variations, specially in the middle section (between the seconds 1 and 2 approximately).

Disgust

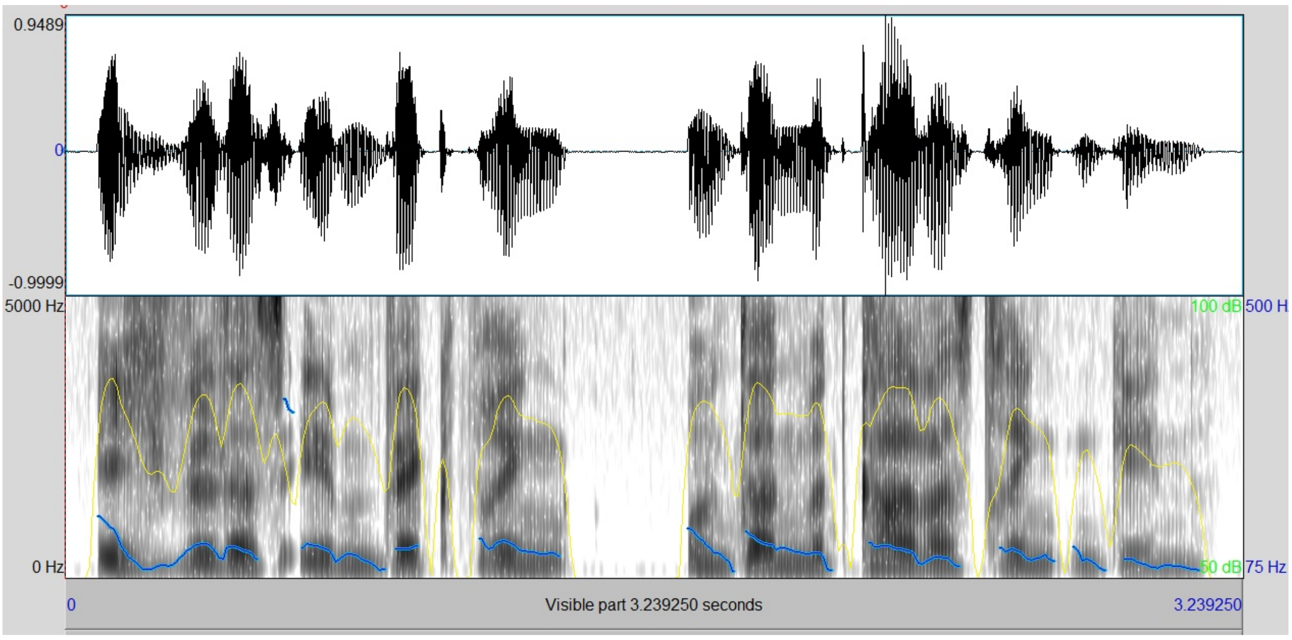


Figure A.3: Disgust class acoustic features

- Very similar pitch shape to the Boredom class, contained under an estimate range of 80 to 100 Hz.
- Intensity waveform presents also sharp changes, showing a common behaviour to the Anger class.

Fear

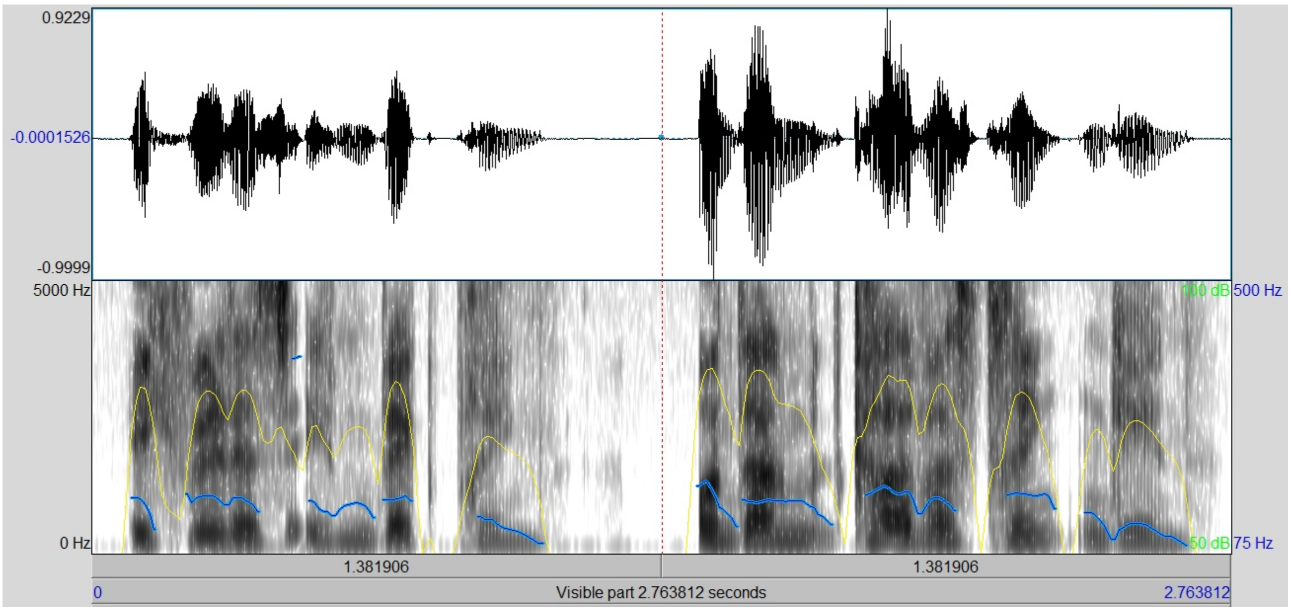


Figure A.4: Fear class acoustic features

- Generally higher pitch frequency values than the last two cases, slightly less monotone shape.
- Very similar waveform for intensity than in the Disgust emotion.

Happiness

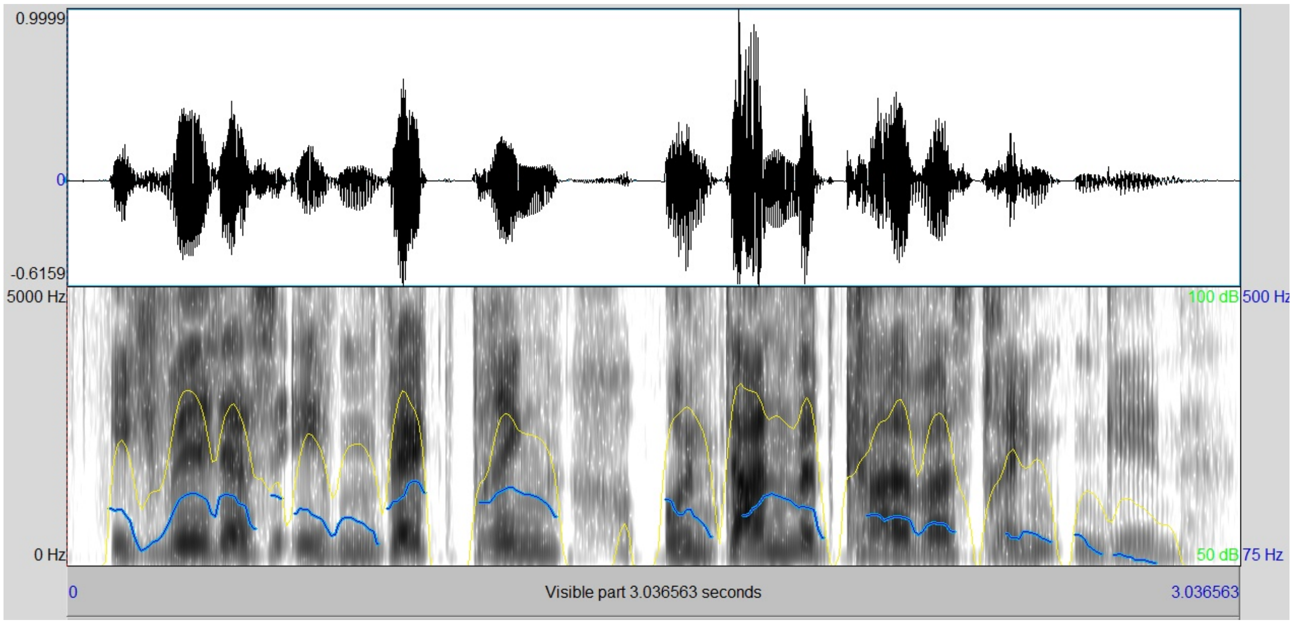


Figure A.5: Happiness class acoustic features

- Higher values of pitch estimation, with more sharply sketched changes in this case.
- We can also allocate the shape of the intensity waveform in the group of Fear and Disgust

Neutral

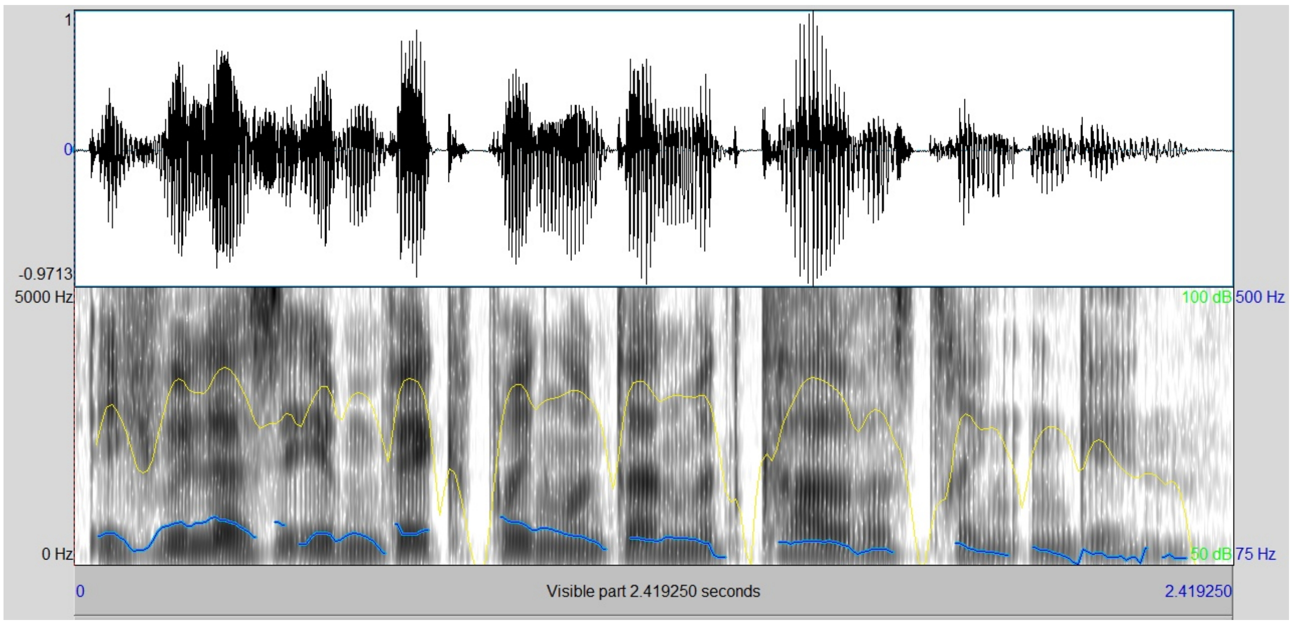


Figure A.6: Neutral class acoustic features

- Its intensity waveform is undoubtedly the one with less variance during the voiced segments, which is logical being this a non-affective state simulation.
- The pitch shape seems to follow this logic, being very smooth and always on the lowest frequency values.

Sadness

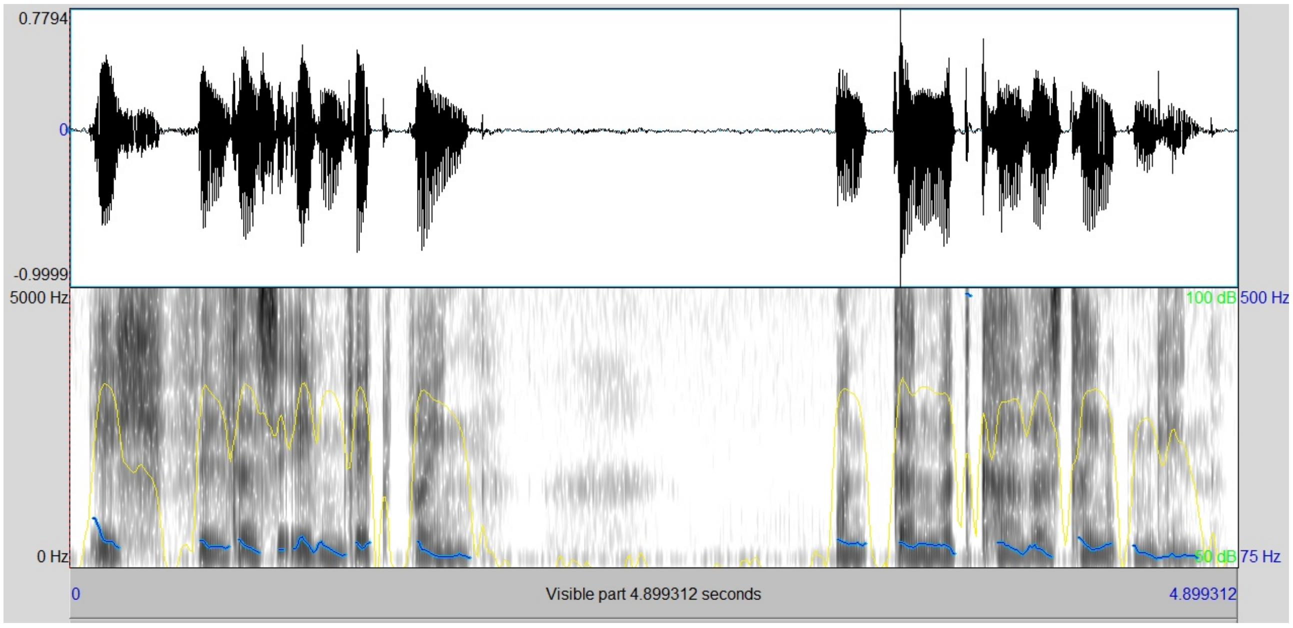


Figure A.7: Sadness class acoustic features

- As a curiosity, it has the longest unvoiced segments of our selected audio files, and thus, is the longest voice file with almost 5 seconds of duration.
- The intensity is most likely the one with the most variance, an effect specially displayed in the first part of the two main voiced segments.
- The pitch is also remarkable, being very similar to the Neutral state (figure A.6).

Appendix B: Script codes used for experiments

B.1. Experiment 1: QtOctave script code

```
% ----------------------------------

% SET FILE NAME

expKey= 'AggregateARFF';
timeOct = int2str(clock);
timeStamp = strrep (timeOct, ' ', '_');
fileName = strcat (timeStamp,'_',expKey,'.arff');

% SET REQUIRED FOLDERS PATHS & CLASS INSTANCES

openSmilePath = ' /home/scallone/00-xBed_VoiceEmotions/03-Tools/00-openSMILE/';
inputFileFolder = ' /home/scallone/00-xBed_VoiceEmotions/02-Data/02-germanWAV/';
resultFolder = ' /home/scallone/00-xBed_VoiceEmotions/02-Data/00-
ExperimentalResults/00-firstExperiment/';
aggFilePath=strcat(resultFolder,fileName);
openSmileConfSubPath = ' ~/00-xBed_VoiceEmotions/03-Tools/00-openSMILE/config/';
configFileName = 'emobase2010.conf';
clsNames={" Anger",  " Boredom", " Disgust", " Fear_Anxiety", " Happiness", "
Neutral", " Sadness"}
chdir (strtrim(resultFolder));

% ----------------------MAIN CODE

for (jj=1:7)
      workingFolder=strcat(inputFileFolder,strtrim(clsNames{jj}),'/');
      chdir (strtrim(workingFolder));
      inputFiles=dir();
      numOfFiles=size(inputFiles,1);
      for (ii=1:42)
            tmp=inputFiles(ii).name;
            % COMPARE FILE FORMAT, DISCARDING DIR FILES '.' and '..'
            if index(tmp,'wav') != 0
                  cmdStr = strcat(openSmilePath, 'SMILExtract', ' -C  ',
      openSmileConfSubPath, configFileName,' -I',workingFolder, tmp, ' -O',
resultFolder,     fileName, ' -instname', clsNames{jj},' -classes {Anger,Boredom,
Disgust,    Fear_Anxiety,     Happiness,Neutral,Sadness} -classlabel',
clsNames{jj});
                  chdir (strtrim(resultFolder));
                  % EXECUTING COMMAND LINE SPECIFIED in cmdStr
                  system(cmdStr);
                  end;
      end;
end;
disp('FINISHED');
% ----------------------------------
```

B.2. Experiment 2: QtOctave script code

```
% ----------------------------------
% SET FILE NAME

expKey= 'emobase2010';   % WILL BE CHANGED ACCORDING TO THE USED FEATURE SET
timeOct = int2str(clock);
timeStamp = strrep (timeOct, ' ', '_');
fileName = strcat (timeStamp,'_FULL_DATABASE_',expKey,'.arff');

% SET REQUIRED FOLDERS PATHS & CLASS INSTANCES

openSmilePath = ' /home/scallone/00-xBed_VoiceEmotions/03-Tools/00-openSMILE/';
inputFileFolder = ' /home/scallone/00-xBed_VoiceEmotions/02-Data/02-germanWAV/';
resultFolder = ' /home/scallone/00-xBed_VoiceEmotions/02-Data/00-
ExperimentalResults/01-secondExperiment/';
aggFilePath=strcat(resultFolder,fileName);
openSmileConfSubPath = ' ~/00-xBed_VoiceEmotions/03-Tools/00-openSMILE/config/';
configFileName = 'emobase2010.conf';    % DIFFERENT FOR EVERY EXPERIMENT
clsNames={" Anger",  " Boredom", " Disgust", " Fear_Anxiety", " Happiness", "
Neutral", " Sadness"}
chdir (strtrim(resultFolder));

% ----------------------MAIN CODE

for (jj=1:7)
      workingFolder=strcat(inputFileFolder,strtrim(clsNames{jj}),'/');
      chdir (strtrim(workingFolder));
      inputFiles=dir();
      numOfFiles=size(inputFiles,1);
      for (ii=1:numOfFiles)
            tmp=inputFiles(ii).name;
            % COMPARE FILE FORMAT, DISCARDING DIR FILES '.' and '..'
            if index(tmp,'wav') != 0
                  cmdStr = strcat(openSmilePath, 'SMILExtract', ' -C  ',
      openSmileConfSubPath, configFileName,' -I',workingFolder, tmp, ' -O',
resultFolder,     fileName, ' -instname', clsNames{jj},' -classes {Anger,Boredom,
Disgust,    Fear_Anxiety,     Happiness,Neutral,Sadness} -classlabel',
clsNames{jj});
                  chdir (strtrim(resultFolder));
                  % EXECUTING COMMAND LINE SPECIFIED in cmdStr
                  system(cmdStr);
                  end;
      end;
end;
disp('FINISHED');
% ----------------------------------
```

Appendix C: Reduced feature sets used in experiment 3

C.1. Feature list for PCA+Ranker method

49-Ranked attributes:

1. *-0.056logMelFreqBand_sma_de[2]_linregerrA – 0.055logMelFreqBand_sma_de[2]_stddev-0.055logMelFreqBand_sma_de[4]_linregerrA – 0.055mfcc_sma_de[12]_linregerrA - 0.055mfcc_sma_de[12]_stddev*

2. *0.081mfcc_sma_de[1]_linregerrA+0.08 mfcc_sma_de[1]_stddev + 0.079mfcc_sma_de[1]_linregerrQ + 0.074lspFreq_sma_de[7]_linregerrA + 0.074lspFreq_sma_de[7]_iqr1-3*

3. *0.099logMelFreqBand_sma[1]_quartile1 + 0.098logMelFreqBand_sma[1]_amean + 0.097mfcc_sma[0]_quartile1 + 0.096mfcc_sma[0]_amean + 0.095logMelFreqBand_sma[0]_quartile1*

4. *0.099lspFreq_sma[1]_stddev + 0.098lspFreq_sma[1]_linregerrQ + 0.097lspFreq_sma[1]_linregerrA + 0.09 lspFreq_sma_de[1]_linregerrQ + 0.09 lspFreq_sma_de[1]_stddev*

5. *-0.114lspFreq_sma[3]_quartile1 – 0.111lspFreq_sma[3]_amean – 0.104lspFreq_sma[1]_amean-0.101lspFreq_sma[3]_percentile1.0 - 0.101lspFreq_sma[1]_quartile1...*

6. *0.102logMelFreqBand_sma_de[1]_linregc1 + 0.101logMelFreqBand_sma_de[2]_linregc1 + 0.101mfcc_sma_de[0]_linregc1 + 0.1 logMelFreqBand_sma_de[4]_linregc1 + 0.1logMelFreqBand_sma_de[3]_linregc1*

7. *-0.085mfcc_sma_de[4]_stddev – 0.085mfcc_sma_de[4]_linregerrA – 0.084mfcc_sma_de[4]_linregerrQ + 0.08 lspFreq_sma[5]_iqr1-3 + 0.079logMelFreqBand_sma[5]_linregc1*

8. *-0.084logMelFreqBand_sma[3]_skewness – 0.08logMelFreqBand_sma[6]_skewness + 0.079logMelFreqBand_sma[0]_pctlrange0-1 – 0.079mfcc_sma[0]_skewness - 0.078logMelFreqBand_sma[4]_skewness*

9. *0.117lspFreq_sma[1]_linregc1 – 0.106mfcc_sma[1]_linregc1 + 0.104lspFreq_sma[4]_linregerrQ + 0.103lspFreq_sma[4]_linregerrA - 0.102logMelFreqBand_sma[0]_linregc1*

10. *0.088lspFreq_sma[0]_kurtosis + 0.087mfcc_sma[7]_stddev + 0.084mfcc_sma[7]_iqr1-3 + 0.084lspFreq_sma[0]_skewness + 0.082mfcc_sma[7]_linregerrA*

11. *0.145logMelFreqBand_sma[3]_linregc1 + 0.128logMelFreqBand_sma[2]_linregc1 + 0.124logMelFreqBand_sma_de[3]_amean + 0.123mfcc_sma[0]_linregc1 + 0.123logMelFreqBand_sma[4]_linregc1*

12. *0.115lspFreq_sma[4]_kurtosis – 0.106lspFreq_sma[4]_iqr1-2 – 0.104lspFreq_sma[4]_iqr1-3 – 0.103lspFreq_sma[4]_upleveltime75 + 0.102mfcc_sma[1]_kurtosis*

13. *-0.105mfcc_sma[10]_quartile3 – 0.105mfcc_sma[10]_amean – 0.096mfcc_sma[10]_quartile2 – 0.092mfcc_sma[10]_quartile1 - 0.083mfcc_sma[10]_percentile99.0*

14. *0.11 logMelFreqBand_sma_de[3]_minPos + 0.106mfcc_sma_de[0]_minPos – 0.104logMelFreqBand_sma[2]_skewness + 0.103logMelFreqBand_sma_de[2]_minPos + 0.1logMelFreqBand_sma_de[4]_minPos*

15. *0.111pcm_loudness_sma_linregerrA + 0.104pcm_loudness_sma_linregerrQ + 0.103pcm_loudness_sma_iqr1-3 + 0.102logMelFreqBand_sma_de[1]_quartile2 + 0.101pcm_loudness_sma_stddev*

16. *-0.099logMelFreqBand_sma_de[6]_skewness – 0.098logMelFreqBand_sma_de[7]_skewness*

– 0.095lspFreq_sma[2]_linregc1 – 0.095mfcc_sma_de[0]_skewness - 0.092logMelFreqBand_sma_de[5]_skewness

***17.** -0.097logMelFreqBand_sma[3]_kurtosis – 0.094mfcc_sma[13]_quartile3 – 0.093mfcc_sma[13]_linregc2 – 0.087logMelFreqBand_sma[2]_kurtosis - 0.083mfcc_sma[13]_amean*

***18.** -0.096mfcc_sma_de[0]_skewness – 0.093logMelFreqBand_sma_de[3]_percentile99.0 – 0.089logMelFreqBand_sma_de[4]_percentile99.0 – 0.087logMelFreqBand_sma_de[4]_skewness - 0.087mfcc_sma_de[0]_percentile99.0*

***19.** 0.109lspFreq_sma[5]_skewness + 0.092mfcc_sma[12]_amean + 0.092mfcc_sma[12]_quartile2 + 0.091mfcc_sma[12]_quartile1 + 0.091lspFreq_sma[2]_linregerrQ*

***20.** 0.105mfcc_sma[9]_linregerrQ + 0.105mfcc_sma[9]_linregerrA + 0.102mfcc_sma_de[9]_pctlrange0-1 + 0.098mfcc_sma_de[9]_percentile99.0 + 0.098mfcc_sma[9]_stddev*

***21.** -0.102mfcc_sma[7]_quartile1 – 0.098mfcc_sma[7]_amean – 0.096lspFreq_sma_de[1]_linregc1 – 0.094mfcc_sma[7]_quartile2 - 0.088mfcc_sma[7]_quartile3*

***22.** -0.093logMelFreqBand_sma_de[2]_minPos – 0.088logMelFreqBand_sma_de[3]_minPos + 0.085mfcc_sma[8]_linregc2 – 0.082mfcc_sma_de[0]_minPos - 0.079logMelFreqBand_sma_de[1]_minPos...*

***23.** 0.099mfcc_sma[10]_linregc1 – 0.088lspFreq_sma_de[2]_linregc1 + 0.087lspFreq_sma_de[2]_linregc2 + 0.085logMelFreqBand_sma_de[5]_maxPos + 0.084mfcc_sma[5]_stddev*

***24.** 0.111F0finEnv_sma_kurtosis – 0.106F0finEnv_sma_skewness – 0.094mfcc_sma[10]_quartile3 - 0.09mfcc_sma[10]_amean - 0.089mfcc_sma[10]_quartile2...*

***25.** -0.146mfcc_sma[9]_stddev – 0.144mfcc_sma[9]_linregerrQ – 0.142mfcc_sma[9]_linregerrA- 0.139mfcc_sma[9]_iqr1-3 - 0.125mfcc_sma[9]_iqr1-2*

***26.** 0.1 mfcc_sma_de[1]_skewness + 0.099lspFreq_sma[5]_percentile99.0 – 0.089logMelFreqBand_sma[4]_iqr1-2 + 0.086lspFreq_sma[4]_percentile99.0 + 0.084mfcc_sma[11]_iqr1-2*

***27.** 0.097mfcc_sma_de[14]_skewness – 0.096lspFreq_sma_de[5]_linregc2 + 0.09 logMelFreqBand_sma_de[1]_skewness + 0.088logMelFreqBand_sma_de[0]_percentile1.0 - 0.087lspFreq_sma[2]_linregc1*

***28.** 0.116mfcc_sma_de[1]_skewness + 0.097lspFreq_sma_de[4]_linregc1 – 0.096mfcc_sma_de[1]_quartile2 – 0.094lspFreq_sma_de[4]_linregc2 - 0.091mfcc_sma_de[0]_maxPos*

***29.** -0.092lspFreq_sma[7]_kurtosis – 0.084lspFreq_sma_de[7]_kurtosis + 0.082lspFreq_sma[4]_percentile99.0 – 0.077lspFreq_sma[2]_linregc1 + 0.077logMelFreqBand_sma_de[4]_skewness*

***30.** 0.101logMelFreqBand_sma_de[3]_quartile2 + 0.094logMelFreqBand_sma[7]_kurtosis + 0.088logMelFreqBand_sma_de[2]_quartile2 – 0.088mfcc_sma_de[3]_quartile2 - 0.086lspFreq_sma[5]_linregc1*

***31.** -0.139lspFreq_sma[6]_upleveltime75 + 0.132lspFreq_sma[6]_skewness – 0.126lspFreq_sma[6]_upleveltime90 – 0.117lspFreq_sma[6]_kurtosis + 0.107mfcc_sma[5]_upleveltime75*

***32.** 0.095mfcc_sma_de[3]_pctlrange0-1 + 0.086mfcc_sma[3]_stddev – 0.084lspFreq_sma_de[3]_linregc2 + 0.083voicingFinalUnclipped_sma_de_pctlrange0-1 + 0.081mfcc_sma_de[3]_percentile99.0*

***33.** 0.117mfcc_sma[9]_percentile99.0 + 0.087logMelFreqBand_sma[6]_percentile99.0 +*

0.083logMelFreqBand_sma[7]_iqr2-3 + 0.076mfcc_sma_de[6]_linregerrQ + 0.075mfcc_sma[9]_quartile3

***34.** 0.098mfcc_sma[8]_kurtosis + 0.097pcm_loudness_sma_kurtosis – 0.096mfcc_sma[11]_minPos +0.095pcm_loudness_sma_skewness -0.093mfcc_sma[8]_skewness*

***35.** -0.091mfcc_sma_de[0]_maxPos – 0.089mfcc_sma_de[7]_skewness + 0.084mfcc_sma_de[7]_quartile2 + 0.083mfcc_sma[10]_iqr1-3 - 0.083lspFreq_sma_de[0]_linregc1*

***36.** -0.123lspFreq_sma_de[1]_amean + 0.109mfcc_sma_de[1]_amean – 0.1lspFreq_sma_de[4]_amean –0.097mfcc_sma_de[11]_amean -0.096lspFreq_sma_de[5]_amean*

***37.** 0.156lspFreq_sma_de[2]_amean -0.142mfcc_sma_de[1]_amean +0.116lspFreq_sma_de[3]_amean +0.115lspFreq_sma_de[1]_amean +0.09 logMelFreqBand_sma_de[5]_maxPos*

***38.** 0.124logMelFreqBand_sma_de[4]_maxPos +0.118mfcc_sma_de[0]_maxPos +0.115logMelFreqBand_sma_de[2]_maxPos +0.113logMelFreqBand_sma_de[3]_maxPos - 0.109lspFreq_sma_de[0]_linregc1*

***39.** -0.108lspFreq_sma_de[3]_amean +0.093mfcc_sma_de[1]_amean - 0.093mfcc_sma_de[6]_amean - 0.09lspFreq_sma_de[7]_upleveltime75 -0.089lspFreq_sma_de[5]_upleveltime75*

***40.** 0.106lspFreq_sma_de[7]_skewness -0.104mfcc_sma[12]_iqr1-2 + 0.096mfcc_sma_de[5]_skewness -0.093mfcc_sma[12]_iqr1-3 + 0.091mfcc_sma[10]_kurtosis*

***41.** -0.093mfcc_sma[9]_kurtosis – 0.092mfcc_sma_de[12]_upleveltime75 – 0.088pcm_loudness_sma_de_kurtosis -0.086lspFreq_sma_de[1]_linregc2 + 0.086lspFreq_sma_de[1]_linregc1*

***42.** 0.112mfcc_sma[14]_skewness +0.11 lspFreq_sma_de[2]_maxPos +0.106lspFreq_sma_de[4]_amean +0.094mfcc_sma[9]_skewness +0.092mfcc_sma_de[8]_iqr2-3*

***43.** -0.111F0finEnv_sma_skewness +0.103F0finEnv_sma_upleveltime75 - 0.093mfcc_sma_de[6]_skewness +0.088lspFreq_sma_de[6]_skewness + 0.088mfcc_sma_de[4]_maxPos*

***44.** 0.098mfcc_sma[10]_upleveltime75 – 0.089lspFreq_sma_de[6]_upleveltime75 + 0.089lspFreq_sma_de[7]_skewness +0.089mfcc_sma[14]_upleveltime75 - 0.087logMelFreqBand_sma_de[4]_quartile2*

***45.** 0.111mfcc_sma[3]_iqr2-3 +0.11 lspFreq_sma_de[6]_quartile2 – 0.097mfcc_sma_de[13]_skewness + 0.096lspFreq_sma_de[7]_quartile2 -0.094mfcc_sma[13]_linregc1*

***46.** -0.126logMelFreqBand_sma_de[7]_upleveltime75 +0.11 lspFreq_sma_de[0]_skewness + 0.108lspFreq_sma[6]_linregc1 +0.106lspFreq_sma[7]_linregc1 - 0.104lspFreq_sma_de[0]_upleveltime75*

***47.** -0.132lspFreq_sma[7]_linregc1 – 0.12lspFreq_sma[6]_linregc1 +0.116mfcc_sma_de[8]_skewness -0.095mfcc_sma_de[7]_kurtosis - 0.094pcm_loudness_sma_de_upleveltime75*

***48.** -0.128mfcc_sma_de[11]_quartile2 -0.111mfcc_sma_de[8]_kurtosis – 0.102mfcc_sma_de[12]_kurtosis +0.1 mfcc_sma_de[4]_amean +0.088mfcc_sma_de[3]_linregc1*

***49.** -0.107mfcc_sma_de[10]_upleveltime75 +0.104mfcc_sma_de[10]_skewness +0.093lspFreq_sma[1]_iqr1-2 -0.092lspFreq_sma_de[6]_quartile2 - 0.088mfcc_sma_de[4]_skewness*

C.2. Feature list for technique 'CfsSubsetEval + Best first ' search method

Selected attributes from original '*emobase2010*' set: **134 (total)**

pcm_loudness_sma_linregc1
pcm_loudness_sma_linregc2
pcm_loudness_sma_skewness
mfcc_sma[0]_maxPos
mfcc_sma[1]_amean
mfcc_sma[1]_skewness
mfcc_sma[1]_quartile1
mfcc_sma[1]_quartile2
mfcc_sma[1]_quartile3
mfcc_sma[1]_iqr2-3
mfcc_sma[1]_percentile99.0
mfcc_sma[1]_upleveltime75
mfcc_sma[1]_upleveltime90
mfcc_sma[2]_amean
mfcc_sma[2]_percentile1.0
mfcc_sma[3]_quartile2
mfcc_sma[5]_stddev
mfcc_sma[5]_quartile1
mfcc_sma[5]_percentile1.0
mfcc_sma[5]_percentile99.0
mfcc_sma[6]_percentile99.0
mfcc_sma[9]_stddev
mfcc_sma[10]_quartile2
mfcc_sma[10]_percentile99.0
mfcc_sma[10]_upleveltime75
mfcc_sma[12]_quartile3
mfcc_sma[14]_linregc1
logMelFreqBand_sma[0]_maxPos
logMelFreqBand_sma[0]_linregc1
logMelFreqBand_sma[0]_linregc2
logMelFreqBand_sma[0]_quartile3
logMelFreqBand_sma[1]_linregc1
logMelFreqBand_sma[1]_linregc2
logMelFreqBand_sma[1]_quartile3
logMelFreqBand_sma[1]_percentile99.0
logMelFreqBand_sma[2]_linregc1
logMelFreqBand_sma[2]_pctlrange0-1
logMelFreqBand_sma[3]_linregc1
logMelFreqBand_sma[3]_linregerrQ
logMelFreqBand_sma[3]_kurtosis
logMelFreqBand_sma[4]_linregerrA
logMelFreqBand_sma[4]_linregerrQ
logMelFreqBand_sma[4]_pctlrange0-1
logMelFreqBand_sma[5]_quartile3
logMelFreqBand_sma[5]_iqr2-3
logMelFreqBand_sma[5]_iqr1-3
logMelFreqBand_sma[5]_pctlrange0-1
logMelFreqBand_sma[6]_linregc1
logMelFreqBand_sma[6]_iqr2-3
logMelFreqBand_sma[7]_amean
logMelFreqBand_sma[7]_linregc1
logMelFreqBand_sma[7]_skewness
logMelFreqBand_sma[7]_iqr1-2
logMelFreqBand_sma[7]_percentile99.0
lspFreq_sma[0]_quartile1
lspFreq_sma[0]_quartile3
lspFreq_sma[0]_percentile1.0
lspFreq_sma[1]_skewness
lspFreq_sma[1]_kurtosis
lspFreq_sma[1]_quartile2
lspFreq_sma[1]_iqr1-2
lspFreq_sma[1]_iqr2-3
lspFreq_sma[6]_upleveltime75
F0finEnv_sma_maxPos
F0finEnv_sma_linregc1
F0finEnv_sma_iqr2-3
F0finEnv_sma_iqr1-3
F0finEnv_sma_upleveltime75
F0finEnv_sma_upleveltime90
voicingFinalUnclipped_sma_maxPos
voicingFinalUnclipped_sma_amean
voicingFinalUnclipped_sma_linregc2
voicingFinalUnclipped_sma_skewness
voicingFinalUnclipped_sma_kurtosis
voicingFinalUnclipped_sma_iqr1-2
voicingFinalUnclipped_sma_pctlrange0-1
voicingFinalUnclipped_sma_upleveltime90
mfcc_sma_de[0]_quartile1
mfcc_sma_de[0]_quartile3
mfcc_sma_de[0]_iqr1-3
mfcc_sma_de[1]_linregc1
mfcc_sma_de[1]_linregc2
mfcc_sma_de[1]_iqr2-3
mfcc_sma_de[2]_linregerrQ
mfcc_sma_de[2]_stddev
mfcc_sma_de[2]_percentile99.0
mfcc_sma_de[3]_linregerrA
mfcc_sma_de[3]_quartile1
mfcc_sma_de[4]_linregerrA
mfcc_sma_de[4]_iqr1-2
mfcc_sma_de[5]_stddev
mfcc_sma_de[5]_quartile3
mfcc_sma_de[6]_upleveltime75
mfcc_sma_de[8]_linregerrA
mfcc_sma_de[9]_linregerrA
mfcc_sma_de[9]_stddev

mfcc_sma_de[9]_quartile1
mfcc_sma_de[9]_iqr2-3
mfcc_sma_de[10]_linregc2
mfcc_sma_de[11]_quartile3
mfcc_sma_de[12]_linregerrQ
mfcc_sma_de[12]_iqr1-3
mfcc_sma_de[14]_linregerrQ
logMelFreqBand_sma_de[0]_quartile3
logMelFreqBand_sma_de[0]_iqr1-3
logMelFreqBand_sma_de[2]_quartile1
logMelFreqBand_sma_de[2]_iqr2-3
logMelFreqBand_sma_de[2]_iqr1-3
logMelFreqBand_sma_de[2]_uplveltime75
logMelFreqBand_sma_de[3]_skewness
logMelFreqBand_sma_de[4]_iqr2-3
logMelFreqBand_sma_de[4]_percentile99
logMelFreqBand_sma_de[5]_amean
logMelFreqBand_sma_de[5]_iqr2-3
logMelFreqBand_sma_de[5]_iqr1-3
logMelFreqBand_sma_de[6]_amean
lspFreq_sma_de[0]_kurtosis
lspFreq_sma_de[1]_linregc2
lspFreq_sma_de[1]_linregerrA
lspFreq_sma_de[1]_kurtosis
lspFreq_sma_de[1]_quartile1
lspFreq_sma_de[2]_linregc1
lspFreq_sma_de[2]_linregc2
lspFreq_sma_de[2]_linregerrQ
lspFreq_sma_de[2]_quartile3
lspFreq_sma_de[3]_iqr2-3
lspFreq_sma_de[5]_quartile1
lspFreq_sma_de[6]_kurtosis
lspFreq_sma_de[6]_iqr1-3
lspFreq_sma_de[7]_kurtosis
F0finEnv_sma_de_skewness
F0finEnv_sma_de_kurtosis
F0finEnv_sma_de_percentile1.0
voicingFinalUnclipped_sma_de_iqr1-3

C.3. Feature list for technique 'CfsSubsetEval + LFS' search method

Selected attributes: **66 (TOTAL)**

pcm_loudness_sma_linregc2
mfcc_sma[1]_amean
mfcc_sma[1]_quartile2
mfcc_sma[1]_quartile3
mfcc_sma[1]_iqr2-3
mfcc_sma[1]_upleveltime75
mfcc_sma[2]_amean
mfcc_sma[2]_percentile1.0
mfcc_sma[3]_quartile1
mfcc_sma[5]_quartile1
mfcc_sma[5]_percentile1.0
mfcc_sma[6]_linregc1
mfcc_sma[9]_linregc1
mfcc_sma[9]_quartile1
mfcc_sma[10]_quartile2
mfcc_sma[10]_upleveltime75
mfcc_sma[14]_linregc1
logMelFreqBand_sma[0]_maxPos
logMelFreqBand_sma[0]_quartile3
logMelFreqBand_sma[1]_quartile3
logMelFreqBand_sma[2]_linregc1
logMelFreqBand_sma[4]_linregerrA
logMelFreqBand_sma[4]_stddev
logMelFreqBand_sma[4]_pctlrange0-1
logMelFreqBand_sma[5]_quartile1
logMelFreqBand_sma[5]_quartile3
logMelFreqBand_sma[7]_skewness
lspFreq_sma[0]_quartile1
lspFreq_sma[0]_quartile3
lspFreq_sma[1]_iqr1-2
lspFreq_sma[1]_iqr2-3
F0finEnv_sma_linregc1
F0finEnv_sma_iqr1-3
voicingFinalUnclipped_sma_maxPos
voicingFinalUnclipped_sma_amean
voicingFinalUnclipped_sma_skewness
voicingFinalUnclipped_sma_quartile1
voicingFinalUnclipped_sma_iqr2-3
mfcc_sma_de[0]_quartile1
mfcc_sma_de[0]_iqr1-3
mfcc_sma_de[1]_linregc1
mfcc_sma_de[2]_stddev
mfcc_sma_de[5]_stddev
mfcc_sma_de[5]_quartile3
mfcc_sma_de[6]_upleveltime75
mfcc_sma_de[8]_linregerrA
mfcc_sma_de[9]_linregerrA
mfcc_sma_de[9]_stddev
mfcc_sma_de[11]_iqr1-3
mfcc_sma_de[12]_linregerrQ
mfcc_sma_de[12]_iqr1-3
logMelFreqBand_sma_de[1]_quartile1
logMelFreqBand_sma_de[2]_iqr2-3
logMelFreqBand_sma_de[2]_iqr1-3
logMelFreqBand_sma_de[3]_skewness
logMelFreqBand_sma_de[4]_iqr2-3
logMelFreqBand_sma_de[5]_amean
logMelFreqBand_sma_de[5]_iqr1-3
logMelFreqBand_sma_de[7]_iqr1-3
lspFreq_sma_de[0]_kurtosis
lspFreq_sma_de[1]_quartile1
lspFreq_sma_de[2]_quartile3
lspFreq_sma_de[6]_kurtosis
F0finEnv_sma_de_skewness
F0finEnv_sma_de_kurtosis
F0finEnv_sma_de_percentile1.0

C.4 Feature list for technique 'CfsSubsetEval + SS-GC' method

Selected attributes from original '*emobase2010*' set: **87 (TOTAL)**

pcm_loudness_sma_linregc2
mfcc_sma[1]_quartile2
mfcc_sma[1]_iqr2-3
mfcc_sma[2]_percentile1.0
mfcc_sma[4]_iqr1-3
mfcc_sma[5]_quartile1
mfcc_sma[6]_linregc1
mfcc_sma[6]_percentile99.0
mfcc_sma[9]_linregc1
mfcc_sma[10]_upleveltime75
mfcc_sma[10]_quartile2
mfcc_sma[10]_percentile99.0
mfcc_sma[10]_upleveltime75
mfcc_sma[12]_quartile3
mfcc_sma[14]_linregc1
logMelFreqBand_sma[0]_maxPos
logMelFreqBand_sma[0]_linregc1
logMelFreqBand_sma[0]_maxPos
logMelFreqBand_sma[0]_quartile2
logMelFreqBand_sma[2]_linregc1
logMelFreqBand_sma[3]_kurtosis
logMelFreqBand_sma[4]_iqr1-3
logMelFreqBand_sma[4]_pctlrange0-1
logMelFreqBand_sma[5]_quartile3
logMelFreqBand_sma[5]_iqr1-3
logMelFreqBand_sma[5]_pctlrange0-1
logMelFreqBand_sma[6]_linregc1
logMelFreqBand_sma[6]_iqr2-3
logMelFreqBand_sma[7]_amean
logMelFreqBand_sma[7]_linregc1
logMelFreqBand_sma[7]_skewness
logMelFreqBand_sma[7]_iqr1-2
logMelFreqBand_sma[7]_percentile99.0
lspFreq_sma[0]_quartile1
lspFreq_sma[0]_quartile3
lspFreq_sma[0]_percentile1.0
lspFreq_sma[1]_linregerrA
lspFreq_sma[1]_skewness
lspFreq_sma[1]_kurtosis
F0finEnv_sma_linregc1
F0finEnv_sma_quartile3
voicingFinalUnclipped_sma_maxPos
voicingFinalUnclipped_sma_amean
mfcc_sma_de[0]_quartile3
mfcc_sma_de[0]_iqr1-3
mfcc_sma_de[1]_linregc2
mfcc_sma_de[1]_iqr2-3
mfcc_sma_de[2]_linregerrQ
mfcc_sma_de[2]_stddev
mfcc_sma_de[2]_percentile99.0
mfcc_sma_de[3]_linregerrA
mfcc_sma_de[3]_quartile1
mfcc_sma_de[4]_linregerrA
mfcc_sma_de[4]_iqr1-2
mfcc_sma_de[5]_stddev
mfcc_sma_de[5]_quartile3
mfcc_sma_de[6]_upleveltime75
mfcc_sma_de[8]_linregerrA
mfcc_sma_de[9]_linregerrA
mfcc_sma_de[9]_stddev
mfcc_sma_de[9]_quartile1
mfcc_sma_de[9]_iqr2-3
mfcc_sma_de[10]_linregc2
logMelFreqBand_sma_de[0]_quartile3
logMelFreqBand_sma_de[2]_skewness
logMelFreqBand_sma_de[2]_iqr1-3
logMelFreqBand_sma_de[5]_amean
logMelFreqBand_sma_de[5]_iqr1-3
logMelFreqBand_sma_de[7]_upleveltime7
lspFreq_sma_de[0]_iqr1-3
lspFreq_sma_de[1]_quartile1
lspFreq_sma_de[2]_quartile3
lspFreq_sma_de[5]_quartile1
lspFreq_sma_de[6]_kurtosis
lspFreq_sma_de[6]_percentile1.0
F0finEnv_sma_de_skewness
voicingFinalUnclipped_sma_linregc2
voicingFinalUnclipped_sma_skewness
voicingFinalUnclipped_sma_kurtosis
voicingFinalUnclipped_sma_iqr1-2
voicingFinalUnclipped_sma_pctlrange0-1