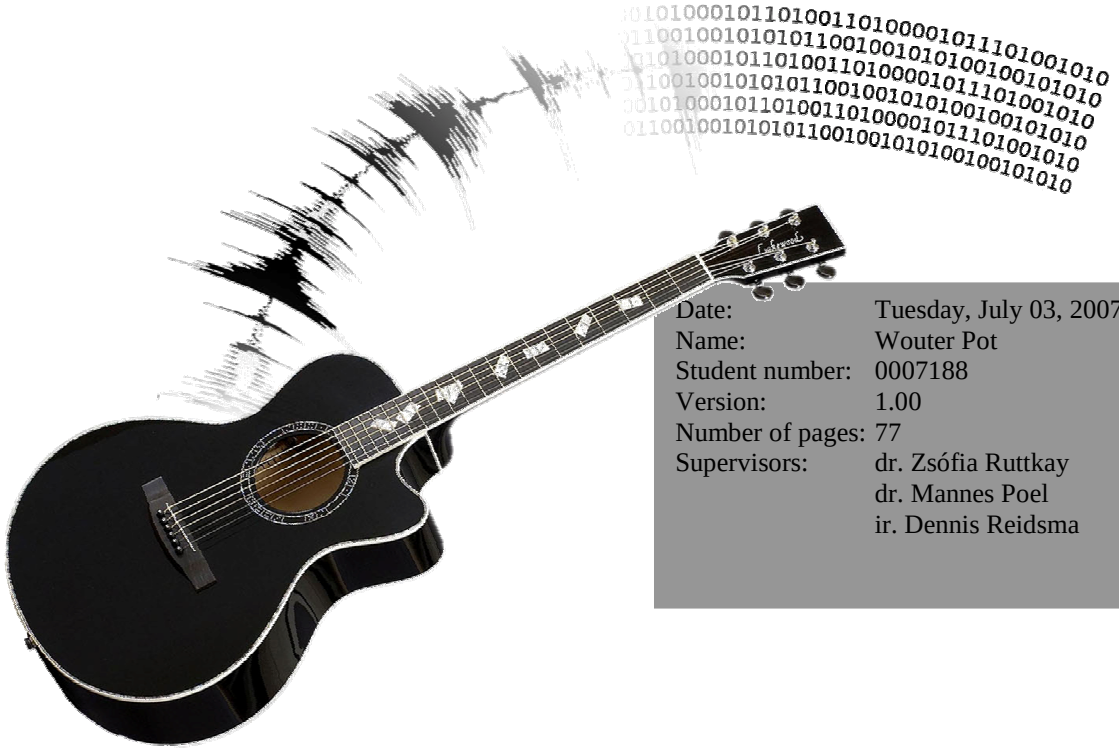
A machine learning approach for generating expressive musical transcriptions

Master thesis



Date:	Tuesday, July 03, 2007
Name:	Wouter Pot
Student number:	0007188
Version:	1.00
Number of pages:	77
Supervisors:	dr. Zsófia Ruttkay
	dr. Mannes Poel
	ir. Dennis Reidsma

Human Media Interaction

University of Twente
Enschede - The Netherlands

Table of contents

Acknowledgements

First of all I would like to thank my supervisors for their patience and willingness to provide constructive feedback at all time. Their positive attitudes helped me keeping progress at moments when I was stuck. Besides my supervisors, I owe much to my parents who never forced things and were understanding throughout the course of this thesis; my friends and family – my young niece and nephew in particular – for the vital distraction.

1. Introduction

Everybody is familiar with digital synthesized music in general. Well known are MIDI-files, one or more tracks with sequenced notes that can be played on a computer or keyboard. Characteristics of music that is played by and synthesized on a computer are that it sounds mechanical and artificial. The striking difference that exists between a human and a machine performance is because of the computer's inability to put expression into music. Expression is the quality that accounts for emotion in music and makes it interesting to listen to. This expression is realized by using the appropriate phrasing, style, interpretation and by applying variations in dynamics and tempo [1]. Although some of these factors can be realized by means of MIDI controls such as the aftertouch (force by which a key is struck), legato, release time (the time it takes for a tone to fade out) or attack time (the time it takes for a tone to fade in) these controls need to be programmed or recorded explicitly. The common MIDI sequencers offer no logic that deduces settings for these controls on basis of the musical structure for example.

The mechanical nature of synthesized sound inspired many researchers the last decades to study expressive performance by quantitative or computational modeling. Computational modeling is an attempt at formulating hypotheses concerning expressive performance in such a precise way that they can be empirically verified (or disproved) on real measured performance data, as defined in [2]. In this article, 4 approaches of computational music modeling were compared: *analysis-by-synthesis* (evaluation of rules brought forward by researchers by professional musicians), *analysis-by-measurement* (obtaining empirical evidence directly from measurements of human performances), applying *mathematical musical music theory* and the *machine learning model* (discovering significant regularities in musical performances via inductive machine learning and data mining techniques). The last model was adopted for this thesis as a way of obtaining an expressive knowledge base.

In order to realize models that can synthesize expressive performances, three stages are distinguished:

1. Capture low-level expressive characteristics of a large set of human performances
2. Develop higher level rules and models on basis of the knowledge base from step 1
3. Use these models to synthesize expressive performances

In this thesis an experimental environment (*Exprimulator*) is presented that realizes the goal of step 1. It aids researchers with capturing these expressive characteristics. This thesis addresses the design, implementation and testing of this piece of software. Exprimulator was tested with guitar performances, that were transcribed (by capturing note onset, note duration and pitch) and annotated on a note-level scale with expressive labels. These expressive labels could in principle denote any playing technique a musician used to convey expression in the performance. However, in order to limit and focus the scope of this thesis one particular expressive dimension (the scale between two complementary playing techniques in which a musician can vary) was chosen as a proof of concept, namely the Right Hand Playing Position (RHPP). Variation in the striking position of the right hand is an important and commonly used form of expression for guitarists. It can effectuate any sound color between sharp and soft tones.

The annotation phase is the main task of Exprimulator, which is globally achieved by training classifiers that can recognize different playing modes in an expressive dimension. By annotating songs performed by human musicians, an image is obtained how well classifiers perform in relation to each other. Through this comparison a selection of the best classifiers is obtained. For this subset it is determined if this performance dominance would also go for the recognition of other expressive dimensions in general.

Describing audio data has a wide range of applications and purposes. A very popular and common application for meta-data are online multi-media querying systems that enables finding musical or video content by using content related query commands (for example searching on genre [3, 4] or query by humming [5, 6]), so that multimedia data can be searched though by a meaningful way for humans. These applications mostly involve high scale descriptors such as the genre of a musical piece or the used instruments. In this thesis, lower level annotation is computed from musical performances, such as the sequence of played pitches and note-level expressive annotation (RHPP).

In contrast to the numerous video annotators, musical annotators are scarce. The most important are: Timeliner [7], the Acousmograph[1], MiXA [8], Marsyas [9], the CLAM Annotator [10] and Mucosa [11]. Timeliner is a pedagogical visualization and annotation tool for a musical library, whereas Acousmograph, Marsyas and the CLAM annotator do not offer (semi)-automatic annotation. Only MiXA and Mucosa incorporate automatic annotation, and aim towards collaborative annotation of existing music. The systems are mainly built in order to create online databases with musical transcriptions that can be queried with high-level descriptors of several dimensions (genre, melody, tonality, instrumentation, style, etc.). This contrasts with the note-level annotation that Exprimulator provides. An annotation tool that is more focused on the annotation on note-level (event level in the percussive context), is described in *Exploration of techniques for automatic labeling of audio drum tracks' instruments* [12]. In this project it has been attempted to create a drum track from a performance that denotes the appropriate percussive instrument for a certain percussive event. The major difference between Exprimulator and the discussed annotators is however that outputted transcription is not the sole goal; it is part of a bigger context – namely that of creating expressive synthesized performances.

In the major part of literature concerning extraction of expressive content or synthesis of expressive music (for example [13-17]) variation in dynamics, timing and melodic structure characteristics are regarded as the constituting elements of musical expression:

- Rhythmical variation (tempo): Notes can be deliberately played too soon or late in order to express a certain feeling (*rubato*), or one can slow down (*ritardando*) or speed up (*accelerando*).
- Volume variation (dynamics): Volume increase or decrease can build up certain suspense or release it. Variation in volume is indicated with for instance *crescendo* (play gradually louder) or *diminuendo* (play gradually softer).
- Pitch variation (frequency): A musician can deliberately deviate from the prescribed pitch from the score, by means of a *vibrato* or *glissando*, for example. This form of expression can only be produced by a limited class of instruments, among which the string instruments.

Albeit the importance of these factors is not denied, this thesis focuses more on the expressive modes that can be affected by applying different timbral playing techniques. This means that annotation takes the form of describing the concrete playing technique that is used, say *sul tasto* or *appoyando*. In this way emotive and subjective expressive labels such as *tender, aggressive, sad, joyful, calm* and *restless* [16] are avoided. The same applies to ambient labels such as *heavy, soft, hard, bright* and *dark* [18]. Concrete labels show unambiguously how a certain tone was played and how it should be played in future. By definition, concrete labels specify a playing mode in one expressive dimension. On the other hand an emotive or ambient label gives a musician freedom to use any of the available expressive dimensions.

In this thesis a method to obtain musical transcriptions annotated with expressive labels by means of machine learners is presented. To reduce the complexity of this problem the scope of the project has been narrowed and concretized to transcribing guitar performances. The transcription has been labeled with expressive labels concerning the right hand playing position (RHPP) of the guitarist. This proof of concept case has been executed in a dedicated environment called Exprimulator that facilitates several tasks such as creating optimal classifiers, transcribing, annotating and visualizing resulting (intermediary) data.

1.1. Research questions

In this thesis the following global question are answered:

- Can scores be annotated with expressive annotation concerning the right hand playing position using a machine learning approach?
- Is the achieved accuracy of annotating the RHPP dimension satisfying enough to make the presented methodology applicable for the recognition of other expressive dimensions?

[1] http://www.ina.fr/grm/outils_dev/acousmographe/

More detailed sub questions that are related to our specific test case of creating annotated transcriptions of guitar performances are stated below.

- What expressive dimensions can be discerned on a guitar, and how can they be divided into classes (equivalent with playing techniques) so that recognition by machine learners is possible? (chapter 4)
- Which audio features can be used as classifier input to determine the playing technique?
- Given a set of classifiers, which one annotates musical performances most accurately and consistently?

A question related to applicability of Exprimulator:

- Does Exprimulator provide enough general freedom for researchers to develop sound classifiers for the annotation of musical transcriptions?

1.2. Applications

Exprimulator is designed as a general tool for sound classification and musical annotation. Therefore, it can serve as an initiation to a variety of future applications.

1.2.1. Experimental environment

Exprimulator can serve as a general environment for exploring feature data and different classifiers. Studies performed with the help of Exprimulator can be compared with each other, because they are similar in execution (same note segmentation, same classifier implementation, etc.). In this way Exprimulator serves as a test environment for carrying out musical sound classification experiments and eases re-execution of these experiments and storage of experimental results.

1.2.2. Annotation of musical transcriptions

Exprimulator can serve as a tool to make more descriptive transcriptions, than regular scores or MIDI files. Besides the basic annotation such as note duration, pitch and onset, Exprimulator provides additional information about the playing technique. This application can have its benefits for existing scores, as these often lack annotation concerning playing techniques. Furthermore, annotation can be provided for any expressive dimension, as long as playing modes defined within the dimension are recognizable by an appropriate combination of feature vectors and classifiers. The annotated transcriptions could be used for musicians who need more guidance in learning a score, or for online databases to enable searching on expressive characteristics.

1.2.3. Initiation to expressive music synthesis

Exprimulator can be used to create a large corpus of annotated musical guitar performances, from which expressive rules can be inferred. These rules can relate musical structure to playing techniques. For this application, a skilled guitarist has to perform a significant amount of songs that are to be transcribed and annotated. The now obtained corpus of songs can serve as a knowledge base to deduce rules that relate musical structural quantities (characteristic note sequences, characteristic sequences of note durations and/or dynamics) to expressive quantities. The intention of these rules is to expose ‘universal’ playing conventions that guitarists share. It is also likely that these rules can capture more specific trends such as the playing style, musical genre, the unique style of a musician or the mood of the musician.

1.2.4. Tutoring guitar students

Exprimulator can function as a tutor that guides a guitar student with playing songs with the appropriate playing technique. With help of classifiers that are trained with different playing modes that were performed by a skilled guitarist, feedback can be given to a student if a tone or a phrase was performed correctly. This feedback is given by means of a prescribed transcription that a pupil has to perform in front of a microphone. From this audio signal appropriate audio features are calculated and provided as input to the classifier which can recognize the playing technique the pupil played. This performed playing technique is then compared with the prescribed technique to provide feedback.

2. Approach

The main goal of this thesis is to be able to recognize different expressive playing modes within an expressive dimension. This chapter outlines the steps taken to realize expression recognition.

Recognition of expressive playing modes is valuable for the different applications proposed in the previous chapter, such as acquiring transcriptions annotated with expressive markup, deducing expressive rules or tutoring guitar students. Several machine learning approaches were chosen to classify a continuous expressive dimension into a discrete set of classes. For testing purposes, these machine learners had to recognize the expressive dimension RHPP. Corpora were recorded that contained sufficient repetitions of tones per distinguished class. In order to train a classifier to recognize different playing modes within the RHPP dimension suitable audio descriptors had to be calculated from the corpus' tones. These audio descriptors were arranged in feature vectors that were used to train classifiers with different machine learning approaches. The accuracy of the classifiers was evaluated and compared by calculating the annotation accuracy over a song corpus consisting of a representative amount of human performances.

Literature study formed an important part of this thesis to determine the ways how guitarists can perform musical expression. This survey was used to pick out a representative dimension for our research, in our case RHPP. Literature study was also conducted in order to obtain suitable audio descriptors and machine learners.

2.1. Goals

Below we list the necessary goals to answer the main research questions. These goals are evaluated in the experiments chapter 8 and the conclusions chapter 9.

- Determining expressive playing modes and dimensions for a guitarist
- Obtaining an optimal training corpus
- Obtaining a selection of best performing classifiers
- Obtaining optimal classifier configurations for the best performing classifiers
- Obtaining suitable audio descriptors and feature vectors
- Proving possibility of expressive playing mode recognition
- Proving possibility of transcribing guitar performances and using expression recognition to annotate them
- Obtaining transcriptions with an annotation performance of at least 80%

2.2. Exprimulator

The experiments (creating corpora, calculating feature vectors, transcribing and annotating guitar performances) have been conducted in a dedicated environment called Exprimulator. This environment has been developed in Matlab and combines the strengths and ease of the scripting language of Matlab with the wide selection of machine learning methodologies that WEKA offers and the audio descriptor calculation features of the MPEG-7 toolbox Matlab-XM.

We used an easy configurable interface (the classifier GUI) to be able to visualize the classifier's training corpus and feature vectors. This gives immediate manual insight how well a certain feature vector separates a class-configuration. Besides that, Exprimulator consists of a transcriber that transcribes and annotates songs using the classifiers of the classifier manager. To make matters more concrete, Exprimulator supports the following tasks:

1. Creation of a sound corpus consisting of tones played with different playing techniques (one realization of a specific playing technique equals a class in this occasion).
2. Calculation of feature vectors for a configurable fraction of each tone.
3. Creation of several types of classifiers that can be trained with the feature data from the preceding step
4. Creating transcriptions (containing note onsets, durations and pitches) of guitar performances.
5. Annotation of transcriptions with the classifiers of the third step. With help of this feature classifier performances can be mutually compared, if ascertained that expressive annotation is possible/satisfying at all (related to the main research question of section 1.1).

These features have been implemented with the research questions and goal setting in mind. Configurability is the key aspect to realize the majority of the goals of the previous section. Configurable GUI controls have to be implemented to investigate the influence of different corpora, feature vectors and classifier settings on the annotation performance. An optimal experimental set-up for future work is expected to be found by varying these dimensions.

In the following sections the sequential tasks Exprimulator performs in order to get to annotated transcriptions are described in more detail.

2.3. Creating corpora

In order to create a classifier that is able to distinguish nominal playing modes on a continuous expressive dimension, training material that exemplifies these classes have to be presented as input. Corpora have been created by playing tones with a certain expressive mode repetitively with a certain amount of jitter to reflect the natural inaccurateness of a guitarist. Subsequently, these recorded waveforms are segmented into individual tones and grouped in accordance with the class arrangement in the expressive dimension. The resulting corpus has to be easily adjustable by excluding tones that were played incorrectly. This manual revision of the corpus is realized by means of the possibility to visually and audibly review the individual corpus tones.

2.4. Calculating feature vectors

In order to derive sufficiently small feature vectors from the corpus' tones that can serve as input for classifiers, appropriate audio descriptor algorithms have to be implemented. Literature on sound classification (preferably timbral sound classification [12]) has been explored to find candidate descriptors. After suitable ones are found they have to be either manually implemented or integrated from existing packages. Once audio descriptors are found that provided adequate class separation, calculation parameters have to be ascertained. Amongst these are the window size, fractions of tones over which the descriptors are calculated, hopsize, window type, etc. These parameters need to be accessible via the GUI of Exprimulator. With the parameters set, the feature vectors of all the tones of the corpus are calculated in one run.

The resulting feature vectors are input in the classifiers as training material together with an index specifying the desired class. These feature vectors can be reviewed as a whole by plotting them grouped per class in different colors. Another way is to review individual feature attributes by creating a scatter plot wherein all the instances are plotted in a domain spanned by two selected feature attribute dimensions. These evaluation methods are used to judge the degree of separation power of the audio descriptors and to recalculate them with different settings if necessary.

2.5. Training & evaluation of classifiers

A majority of the classifiers the WEKA toolbox offers are presented in Exprimulator's GUI to be trained by the created training corpus of the previous section. Various WEKA settings which are unique for each classifier have to be configurable via Exprimulator's interface. Exprimulator has to provide the freedom to compose a unique corpus for each classifier (by including or excluding tones, or by changing the class

configuration). For comparison considerations, the corpora have been kept constant throughout the experiments though.

The trained classifiers can be reviewed visually or statistically. Visual review provides insight in the trained model by plotting its structure, weights, or parameters. Statistical review means the prediction of the tones present in the song corpus by the trained classifiers, after which the correct prediction fraction is calculated. These measures are used as a guideline to optimize a classifier by fine-tuning its WEKA settings. The prediction performance is also used to obtain better performing classifiers. By defining a quantitative exclusion rule for the classifiers with the lowest performance, a reduced set of classifiers has been determined. This reduction spares evaluation time for future experiments.

2.6. Transcribing & annotating song corpus

A number of monophonic guitar performances have been recorded to create a song corpus like mentioned in the previous section. The songs are well-known melodies and contain different expressive playing modes that are also present in the training corpus. The playing modes that are subsequently used in the performances are annotated manually for comparison with the classifiers' automatically predicted labels.

In order to create a transcription the songs have been segmented at the note onsets to begin with; secondly the pitches of these segments (tones) have been calculated. For the individual tones the same feature vectors (section 2.4) are calculated as for the training corpus (2.3) after which they are presented to a trained classifier. The prediction outcome of a certain classifier is used to annotate the transcription. By annotating the song corpus with different classifiers the annotation performances are compared in order to reduce the set of classifiers.

3. Literature review

The extensive amount of literature on sound classification has been used to make choices for the design of the machine learning methodology. Reviewing this literature helps to narrow down the possible ways to conduct this research and helps finding common approaches for sound classification. Finding a common approach comes down to the selection of common classifiers and audio descriptors primarily. Research on sound classification often has the character that a set of classifiers are compared by using some performance measure. Also feature selection methods are applied to obtain optimal audio descriptors. The results of such experiments can be used to the advantage of this project to exclude classifiers and audio descriptors that have been proved unsuccessful. In like manner, extra attention can be paid to classifiers which have been proven more successful.

3.1. Audio descriptors

Due to the fast growing body of multimedia content on the internet, the necessity for adequate storage with descriptive meta-data rises. This meta-data is necessary for humans to be able to retrieve media with content describing queries, instead of just a filename. The idea is to make the web just as searchable for multimedia content as it is for text. From the calculation of lower level acoustic properties of media content, higher level descriptors that are understandable and relevant for humans can be derived.

In the scope of audio retrieval the need for these descriptors is just as urgent as in the fields of image and video retrieval. Examples of recent audio retrieval methods are query by genre, by humming, by spoken/sung content or by rhythm. Besides retrieval applications, audio descriptors are used for other tasks such as score following/aligning or the sole purpose of providing musical annotation, as in this thesis. For the latter purpose, classifiers are used to 'convert' low level descriptors to higher level annotation that has an expressive musical meaning. In order to find good candidates for audio descriptors, an investigation of the musical characteristics that need to be derived is necessary. As has been stated before, the forms of musical expression that this thesis focuses on are timbre related, as is the case with RHPP. Analysis of musical audio by timbre recognition is an approach that has been applied often in the scope of musical instrument [19], playing technique [20, 21] and genre [3] recognition. These applications are considered related to the RHPP recognition problem in principle, and are therefore scanned for commonly used audio descriptors.

By scanning the relevant timbre related literature, in a lot of occasions the MPEG-7 standard have been used, a description framework in which several commonly used higher and lower descriptors are implemented. An important benefit of using a standard approach is the compatibility with other software that uses MPEG-7 and the uniformity by which the audio description algorithms are defined and implemented. This compatibility is ensured by the fact that MPEG-7 uses the widespread XML-scheme for describing the multimedia content. MPEG-7 has been used in several applications that are closely related to the ones of this thesis, namely the classification of musical instruments [22-26], musical sound recognition within one instrument [21], genre recognition [27], environmental sounds recognition [28-30] and music recommendation [31]. Although not all these applications are strictly related to expression recognition, the MPEG-7 descriptors are designed to be universal in their application. The following section outlines the descriptive strength of the MPEG-7 audio descriptors.

3.1.1. MPEG-7 audio description

MPEG-7, informally known as the *Multimedia Content Description Interface*, is an attempt to standardize meta-data provision for multimedia. The MPEG-7 standard contains the following sub tools: MPEG-7 Visual – for describing video content; MPEG-7 Audio – for describing audio content; MPEG-7 Multimedia Description Schemes – description tools that deal with generic features and multimedia descriptions; and MPEG-7 Descriptions Definition Language – the language that defines the syntax of the former description tools. The relevance and importance of MPEG-7 is stressed by summing up some of application areas and accompanying examples: broadcast media selection – for constituting custom radio or TV channels; journalism – searching speeches for politicians using their face or voice; or remote sensing – cartography, ecology, etc.

The MPEG-7 unit that is relevant in the light of timbre recognition is MPEG-7 Audio. The audio descriptors that can be calculated with this toolbox are divided in low-level descriptors (LLD's), such as spectral, parametric or temporal features of the signal, and high-level Description Schemes (DS). The LLD's describe sound characteristics such as harmonics, sharpness, pitch and timbre. They are applied to short time frames of the sound signal and yield either scalars or vectors, which can be aggregated using mathematical operators such as minimum, maximum, mean, and variance. The Description Schemes describe sound at a higher level and are more related to the way humans describe audio. Examples of these description schemes are *Musical Instrument Timbre Description Tools*, *Melody Description Tools* and *Spoken Content Description Tools*. The LLD's are depicted in Figure 3.1, and grouped in seven categories. In the following, the MPEG-7 descriptors are outlined per category.

Basic Descriptors

The basic descriptors basically represent the shape of the waveform of an audio signal. The *AudioWaveform* is a representation of the waveform's envelope. The *AudioPower* descriptor computes the average square of the waveform in a certain time frame, thereby describing the power of a signal over time.

Basic Spectral

This group describes basic properties of the audio signal's spectrum. The *AudioSpectrumEnvelope* (ASE) is a logarithmic representation of the short-term power spectrum in frequency bands. The *AudioSpectrumCentroid* is the center of gravity of the ASE spectrum. The *AudioSpectrumSpread* describes the deviation of the ASE *from* its centroid. The last descriptor, *AudioSpectrumFlatness*, is a fingerprinting algorithm for determining the similarity between two sounds.

Spectral basis

The spectral basis descriptor group contains the *AudioSpectrumBasis* and *AudioSpectrumProjection* descriptor. The former transforms a signal's spectrum to a lower dimensional representation, while the latter does the same for the ASE. Both descriptors aim at representing the spectra while preserving the maximum amount of information.

Signal parameters

This group contains the *AudioFundamentalFrequency* and the *AudioHarmonicity* descriptors. The first descriptor calculates the fundamental frequency curve of an audio signal. The second calculates the harmonic ratio (varying between 0 for noise to 1 for purely harmonic) as well as the upper limit of harmonicity.

Timbral temporal

The *LogAttackTime* descriptor calculates the time from the onset of a sound to the point where it reaches its maximum. The *TemporalCentroid* is the location in the signal where most of the energy is concentrated.

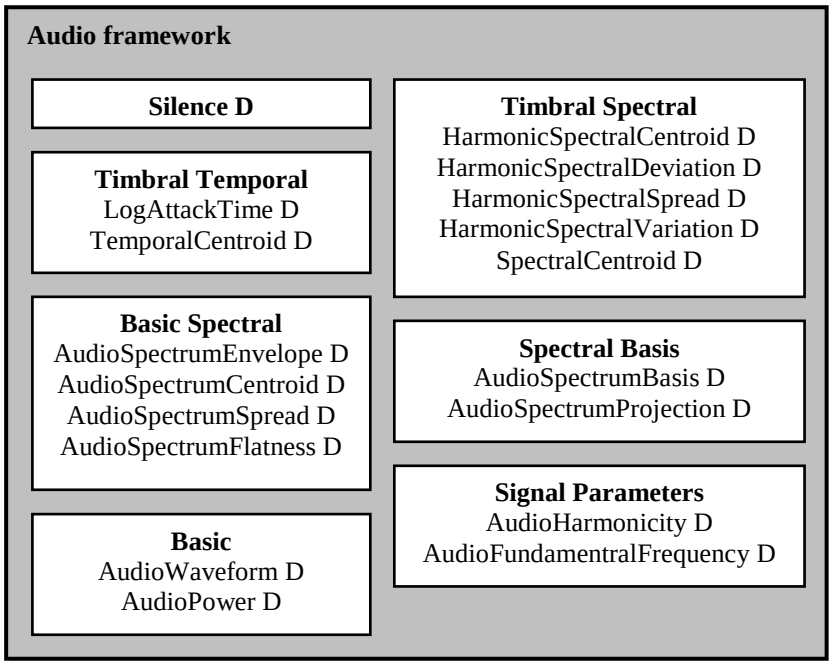


Figure 3.1: Low-level MPEG-7 audio descriptors

As one might expect, in the discussed literature from section 3.1 concerning timbre recognition, the *Timbral Temporal* and *Timbral Spectral* categories play an important role. These categories describe the perceptual features that make two sounds having the same pitch and loudness sound different. The descriptors can be related to human notions such as 'attack', 'brightness' or 'richness'. Because of these reasons these LLD's would seem obvious candidates for the application of recognizing RHPP.

3.2. Machine learning for sound classification

Machine learning is a commonly applied methodology in the field of sound recognition and classification. The applications vary from environment recognition for PDA's [29] and musical instrument recognition [25] to emotion recognition [32]. The more closely related literature on timbre recognition from the previous section was scanned for common machine learning approaches.

Most of the literature discusses around 6 different classifier approaches that fall in one the following categories: decision trees, Neural Networks, Bayesian learning, lazy learning, rule learning or Support Vector Machines (SVMs). The different classifiers are evaluated on test material so that their performances can be compared. It is worth noticing that in most of the projects no conclusions have been drawn about the best performing classifier, and results of mutual projects seem to contradict in case the same classifiers have been used. However, in general the used set of classifiers and experimental approach differs too much to compare one study with the other.

A considerable amount of studies use the classifiers implemented in Waikato Environment for Knowledge Analysis (WEKA, see appendix B). Compatibility, standardization and robustness reasons motivate incorporation of WEKA classifiers within Exprimulator. Section 6.3 outlines in greater detail what classifiers are generally addressed in literature, along with the degree of similarity with this thesis.

4. Musical theory

In order to start with recognition of musical expression, it is important to determine what dimensions of expression exist and – within the expressive dimensions (ED) – the playing modes that can be identified. This chapter starts with a general explanation of musical expression. Section 4.3 briefly summarizes per instrument category what the most important expressive dimensions are. The subsequent section outlines for the guitar what static and dynamic parameters influence the timbre of the guitar sound. Finally, in section 4.6 several EDs are defined by the scale between two complementary playing techniques. The playing modes that are a result of dividing the EDs are to be recognized by human ear or a machine learning methodology.

4.1. Musical expression

Expression can be defined as a quality that accounts for the specific emotional effect of music. When a musician plays with expression, the listener (1) may praise his musical sensitivity or that the musician has a keen sense of how the passage should be played. One can also say (2) that an expressive performance is one that recognizably embodies a particular emotion, and may cause alike emotional response in the listener[2].

4.1.1. Explicit and implicit characteristics

Musical performances contain a lot of implicit characteristics (characteristics which are not the result of the explicit annotation of Table 4.1) beside explicit characteristics. The implicit characteristics are the result of the musician's unique playing style, implicit musical style conventions and the acoustic properties of the guitar. Partly, these implicit characteristics are beyond the ability of the musician to control, such as the acoustical properties of an instrument. Besides that, from the characteristics that are controllable by the player, a large part is applied unconsciously. The part that is applied consciously can be regarded as musical expression. This project tries to capture the expressive part of implicit characteristics, by extracting them from recorded human performances. Combined with the explicit markup already present in a score file, rules can be obtained by finding relations between the explicit markup and the implicit characteristics extracted from the human performances.

Explicit characteristics	**Implicit characteristics**
Note duration	Note length deviation
Note pitch	Different timing
Dynamics	Timbral variation
Vibrato	Plucking style
Plucking style	

Table 4.1: Explicit and implicit characteristics

The distinction between implicit and explicit characteristic is not disjoint, i.e. most of the explicitly annotated characteristics (such as vibrato or portamento) can also be applied implicitly (i.e. applied to the player's insight, when there is no explicit annotation present that tells the player to do so). Vibrato is a good example of an expression that is used a lot among guitarists and violinists in more places than those prescribed by explicit annotation. Very often it is applied unintentionally and/or unconsciously.

[2] http://www.people.carleton.edu/~jlondon/musical_expression_and_mus.htm

Explicit annotation can be (un)consciously deviated from or supplemented as follows:

Consciously deviating	Deviating from annotation from score due to musician's preferences
Consciously adding	Adding expression because of musician's preferences or gaps in annotation
Unconsciously deviating	Deviation because of technical imperfection, misinterpretation of annotation or personal playing style
Unconsciously adding	Addition of expression because of musician's personal style that is imprinted by years of practicing and therefore applied unconsciously

Table 4.2: Deviation or suppletion of explicit annotation

4.1.2. Relation between expression and playing techniques

Musical expression is a subjective notion related to conveying an underlying emotional message of a song and bringing listeners into that desired emotional state. The subjective emotional message is not directly interpretable by a computer. Therefore the concept of musical expression is concretized to analyzable quantities such as playing techniques that are presented in section 4.6. Although the application of playing techniques that this project focuses on does not comprise the full arsenal a musician has at his disposal for transmitting an emotional message, it forms an important part. Playing techniques affect to a large extent the timbre that is produced when playing a song. Timbre, on its turn, is a concept that is strictly related to musical expression because a tone's timbre is often explained in terms with an emotional load or terms related to the taste or touch sense. Examples of such terms are *soft*, *harsh* or *muddy*, which are commonly heard when music is described orally. Although the interpretation of these descriptions are not determinative either, timbre can be described in terms of measurable acoustic quantities equally well (see section 3.1, amongst others). This measurability enables computational analysis and annotation of performances with timbral markup.

4.2. Communicating expression with annotation

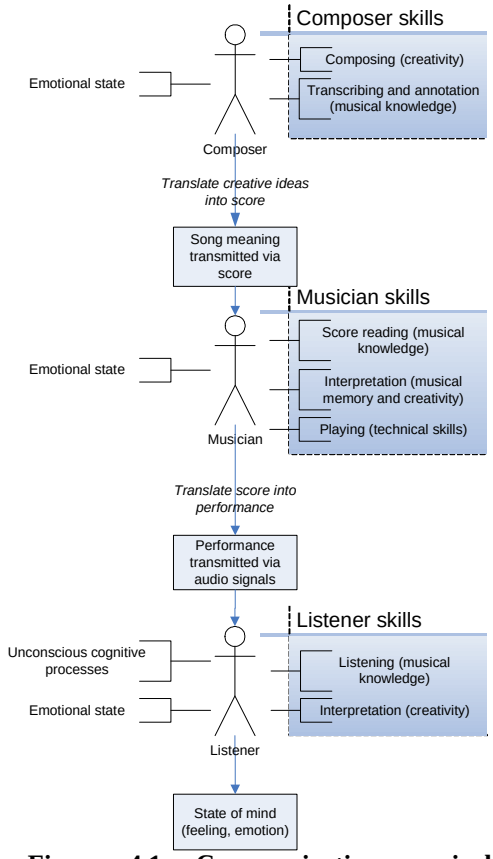


Figure 4.1: Communicating musical message from composer to listener

In expressing music, there are a lot of ambiguities that can limit the clarity by which a composer can convey his message. A composer initially starts with a musical piece with a certain message attached to it that should be conveyed through the score. At several stages, however, this conveying can be hampered. Firstly, the song meaning must be transcribed in musical notation, with restricted freedom of expression. A composer can choose to incorporate annotation with the following increasing levels of concreteness:

- Emotive annotation (e.g. *amoroso* – loving): with this kind of annotation, the composer makes clear what emotional character a musical phrase should bear. The resulting musical performance should convey this specific emotion. This is the vaguest category of transcription.
- Relative annotation (e.g. *crescendo, ritardando)*: describes a change in tempo or volume in relation to the current playing conditions. Examples of these relative annotators are *crescendo* which means to play louder, or *ritardando,* referring to a slow down. It is not always clearly denoted what the pre- and post playing conditions are.
- Technical annotation (e.g. *appoyando, vibrato*): describes mostly with fixed musical terms the technique that the musician has to apply. This technical annotation is limiting in the way that it is

not detailed enough: a score mostly provides only the keyword '*vibrato*' under the music bar, without any further clues for the musician how to fill in the unknown parameters, such as the speed and the amplitude of the modulation.

- Absolute annotation: can be interpreted unambiguously, because it involves numeric quantities. Examples are metrical indicators such as the time signature (4/4 for example) or the tempo (in BPM).

Figure 4.1 depicts the way through which a musical message or song meaning is normally transmitted from composer to listener through the musician and the media *score* and *performance*. It also depicts the actions that the actors have to undertake in order to come to the intended intermediate states and final state of mind. These actions require unique personal skills (the most important are depicted between the parentheses) that are the reason why the musical communication course is never the same. Within this thesis the computer performs the listener's role to extract the initial score from the beginning of the course. In this way the communication course is executed in the reverse direction, without intervention from the musician. The intended characteristics for the computer to capture are quantitative parameters, but do not necessarily reconstruct the original score.

4.3. Expressiveness per instrumental category

For this thesis, expression recognition is demonstrated by using a guitar. The ultimate goal would be the realization of universal expression recognition for any musical instrument, but the ways to realize expression per instrument are in most cases too unique and specific. While the role of the guitar in this thesis is merely a demonstrating than a motivated choice, there are some benefits of this instrument. String instruments in general offer more expressive freedom than other categories mainly because both the pitch and the onset of a tone are directly controlled by human hand without mechanical intervention. This mechanical intervention is present for instance in a piano, where a hammer touches the string at a fixed location, thereby narrowing the control over the timbre.

Other instrumental categories offer control over other expressive dimensions. For instance, with the family of wind instruments the envelope of a tone can be controlled to a great extent, thereby allowing a crescendo within a tone. This is an ability that only the bowed string instruments are capable of within the family of string instruments. For the other 'plucking' instruments, the sustain follows invariably a certain exponential decay function. An expressive mode that is important for the percussive instruments is the striking location. Whereas pitch is of no concern for the majority of percussionists, their attention is mostly directed towards the realization of different timbres by striking a cymbal, tom or snare drum on a unique location.

4.4. Guitar anatomy

Before proceeding to the discussion about expression in relation to the guitar, the guitar's anatomy is displayed in Figure 4.2 to illustrate the naming of the different parts of the guitar. Through the course of this thesis these terms are frequently used.

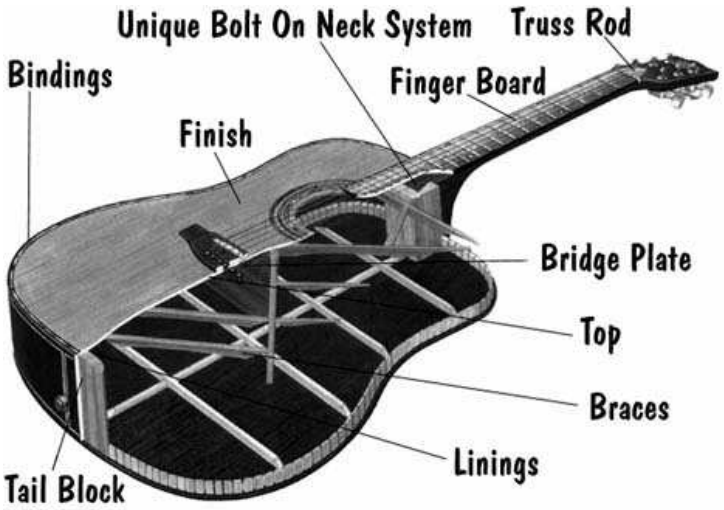


Figure 4.2: Anatomy of an acoustic guitar[3]

4.5. Timbre and guitar

The concept of timbre is hard to define, in comparison to other psycho acoustical features such as pitch, duration and intensity. These latter can be scaled and organized hierarchically and notated, but timbre has a very complex definition and is multi dimensional in the time and spectral domain [33]. According to the ANSI definition timbre is that attribute of sensation in terms of which a listener can judge that two sounds having the same loudness and pitch are dissimilar. There are five timbral parameters to distinguish [34]:

1. The range between tonal and noise like character.
2. The spectral envelope.
3. The time envelope in terms of rise, duration, and decay.
4. The changes both of spectral envelope (formant-glide) and fundamental frequency (micro-intonation).
5. The prefix, an onset of a sound quite dissimilar to the ensuing lasting vibration.

In order determine how the timbre of a guitar is constituted, a distinction is made between the unique static timbral properties of a guitar, and the dynamical timbral differences that can be effected by a musician. The next two subsections outline the most attributive components of static and dynamic timbre.

4.5.1. Static timbral properties of the guitar (physical characteristics) [4,5,6]

Guitar builders can recognize the limitless factors that influence the sound of a guitar. While a musician can influence the sound of guitar by playing differently, every guitar also has from itself a unique sound color. The acoustic properties that define this static timbre are largely defined when building a guitar. It is important to be aware of these characteristics, as static sound differences between two guitars can make general dynamic timbre recognition impossible. Below the most important properties of a guitar are outlined that define its sound.

[3] http://www.phys.unsw.edu.au/music/guitaracoustics/anatomy.html

[4] http://www.phys.unsw.edu.au/music/guitaracoustics/anatomy.html

[5] http://www.frets.com/FRETSPages/Musician/Strings/Strings/strings01.html

[6] http://www.wikipedia.org

Material composition

The materials that are used for the construction of a guitar have direct consequences for its acoustic characteristics. The part that defines the guitar's sound most dramatically is the body, especially the top plate. The majority of the guitar's sound is namely produced when the vibrating energy from the strings are transferred via the bridge to this plate, causing it to resonate. The detail by which the top plate resonates is a key defining element for the guitar's timbre. The traditional and up to date most commonly used material is wood, mostly tonewood like spruce, red cedar or mahogany. A common problem with the application of wood is that two plates of wood are never the same, due to natural variation. With this knowledge it is impossible to create two guitars with the same timbre.

A variety of synthesized materials have been explored to imitate or replace existing woods. Examples of these materials are fiberglass, carbon fibre and polymers. The most important arguments for research of these materials are that synthetics are cheaper and believed to exhibit less variation. The latter assumption has not been validated yet, as the synthetics expose as much acoustic variation as wood. Furthermore the characteristics of the synthetics still cannot mimic those of wood, so the timbre of wood cannot even be approximated with synthetics.

The strings, the initiators of the sound, exist in many variants. The choice of the string is a subjective one which is up to the musician. The most important difference exists between Spanish guitars and steel-string guitars. The top three strings of Spanish guitar are nylon strings, whereas the lower three are composite strings made of a silk fibre core wrapped with silver or gold-plated copper wire. Steel string guitars are equipped with plain steel string for the three highest strings and steel strings wounded with a wire of some metal alloy. The most commonly used alloys are bronze, phosphor bronze, nickel or silver plated copper. Sometimes the core is wrapped with silk filaments before wound by the steel wire. The used alloy has much effect on the sound and durability of the strings.

Guitar's anatomy

When constructing a guitar there are many choices to be made that have large effect on the timbre. The most important part with respect to construction is – again – the guitar's body. The back of the top plate of the body is supported by a grid of wooden strips, called braces (see Figure 4.2), which can be arranged in different geometries. The bracing technique is acoustically critical: it alters the stiffness-to-mass ratios and elastic moduli dramatically, thereby defining the guitar's sound radiation. These braces are necessary to support the top plate, because it is relatively thin in comparison with its surface. The guitar's sound hole, through which sound is transmitted to the exterior of the guitar, can vary in shape and size. This geometry is most important for the transmission of lower frequencies. Lower frequencies are namely produced by the 'coupling' between successively the strings, bridge, sound-board, ribs (sides of the guitar), back plate and ultimately the air cavity in the guitar's body. If these elements have a strong interaction, the guitar is said to be *strongly coupled*. The higher frequencies are mainly realized by the coupling between the strings, bridge and top plate. The interaction strength can be tuned by a luthier according to taste: a certain amount is necessary for the transmission of vibration, but too much coupling causes harsh tones.

4.5.2. Dynamic timbre control

This paragraph describes how a musician can control the timbre by applying playing techniques with his hands (dynamic timbre control). Through investigation of these techniques, the expressive dimensions of section 4.6 can be defined.

4.5.2.1. Right hand playing position

By changing the plucking position of a string, the guitar can produce a rich palette of different timbres varying from a thin, metallic sound (near the bridge) to a thicker and smoother sound (above the fret board); see Table 4.3. Skilled musicians never play statically at one location on the string, but apply variations conforming the score's annotation, prescribed dynamics or the mood of a phrase instead. Inherent to the technique is that all the tones of a polyphonic performance are affected. Varying the plucking position is a technique that is applied on a relatively high scale, meaning rather on sequences of notes than per note. The technique is commonly applied per musical sentence (sequence of notes forming

a melodically whole, marked by pauses). In this way, the distinction between sentences is emphasized and effects such as question-answering can be realized. Below is displayed how the playing style is related to psycho acoustic properties and physical properties.

Playing style	Psycho acoustic properties	Physical properties
At bridge	Metallic, sharp	Stronger higher harmonics
On fret board	Smooth, round	Fundamental frequency stronger, less upper harmonic content

Table 4.3: Right hand playing positions and resulting sound properties

4.5.2.2. Right hand finger configuration

Timbre variation can also be realized with a fixed right hand position, but then varying the finger configuration. A couple of variables are stated below that can change the sound [35]:

1. **The angle of contact between finger and the string:** A slight rotation of the wrist can drastically affect the angle between the finger and the string. A guitarist can achieve a range of timbres in between striking the string with the right or left side of the finger nail. The sound is the smoothest at the edges and the most metallic when the nail is in line with the string.
2. **The amount of weight transmitted to the string via the finger:** Involves channeling the weight of the entire arm to one finger, ranging from very light (the string barely flexes before it is released) to as much weight as possible without creating a distorted sound.
3. **The firmness of the first joint in the striking finger:** This variable determines the firmness of the sound. As the first joint is loosened, the finger becomes less perpendicular to the string as the stroke is made. With more firmness on the other hand, the effected sound is more metallic, without having to move the right hand towards the bridge.
4. **The ratio of fingernail to flesh touching the string at release time:** This ratio determines the degree of harmonic overtones the vibrating string produces. An attack with pure nail results in a lot of depth in the higher harmonic overtones, while a combination of flesh and nail produces a more full sound, with a wider range of overtones including the lower overtones and sub harmonics.
5. **The direction from where the string is touched:** The string can be struck from below (*tirando*) or from above (*appoyando*). The *appoyando* technique realizes a louder and a fuller sound.

4.5.2.3. Left hand finger configuration

The left hand is not a common way to create different timbres with. A professional guitarist would attempt to find the optimal position to place a finger within a fret, in order to create the clearest sound. This means placing the finger as close as possible to the fret without extending it. Any placement further from the fret results in a buzzing sound, which is not considered as a way of intentionally realizing timbre within this thesis.

However, sometimes tones are intentionally played in a certain fashion to create harmonics, which sounds radically different than an ordinary guitar tone. Basically, when playing harmonics, the fundamental frequency and some of its overtones are eliminated. This effect is obtained by slightly touching instead of pressing the string above the fret with the left hand. This can only be realized in certain fret positions, namely the ones that divide the string into halves, thirds or fourths. They double, triple or quadruple the pitch of the string's base frequency respectively. One does not come across harmonics regularly in scores, and it is mostly used for aesthetic reasons or for special occasions (comparable with the scarce use of *pizzicato*). For this reason, it is hard if not impossible to induce expressive rules for the use of harmonics.

4.6. Overview of expressive dimensions

As a result of the preceding outline of dynamic timbre control, some dimensions are proposed on which different playing modes can be arranged. The dimensions are necessary to define because it is used as a guideline for the creation of a class distribution that defines the outputs of the classifiers in chapter 6.

These outputs are used on their turn as eventual annotation for the transcription. This means that the transcription contains annotation relating to the applied playing technique.

The following are only a selection of the total amount of timbre comprising dimensions. A convenient property of these 5 defined dimensions is however, that they can be applied simultaneously. The dimensions are defined by two complementary playing techniques, indicated between the parentheses. The first four are continuous dimensions, meaning that there are an unlimited number of intermediate playing modes between the two extreme playing techniques. *Fingering* on the other hand is a discrete dimension.

Degree of palm muting (*Pizzicato – non-pizzicato*)

The degree of muting is a measure for the extent by which a guitar's string is damped by slightly resting the side of the right hand on the strings near the bridge. This results in a buzzing tone that has a shorter sustain, less overtones and is softer. The amount of muting can be increased by applying more pressure on the strings, or by moving the side of the hand farther from the bridge.

Plucking direction of fingers (*Appoyando-Tirando*)

The playing techniques *appoyando* (rest stroke) and *tirando* (free stroke) refer to the direction the strings are pulled to. With *tirando* a bended finger is positioned below a string before it is plucked, thereby raising the string. With *appoyando* the opposite technique the string is 'pushed' downward before released, after which the striking finger rests on the adjacent string. *Appoyando* generally results in a fuller sound containing more harmonics, while a *tirando* tone is more metallic.

Right hand playing position (RHPP) (*sul tasto – sul ponticello*)

The Italian annotation *sul tasto* and *sul ponticello* refer to the playing position of the right hand, respectively on the fret board or at the bridge. *Sul ponticello* produces a characteristically glassy sound, which indicates that the tone contains more higher harmonics at the expense of fundamental harmonics. Below the position of the techniques in relation to the sound hole are indicated together with the unofficial intermediate label *neutral*.

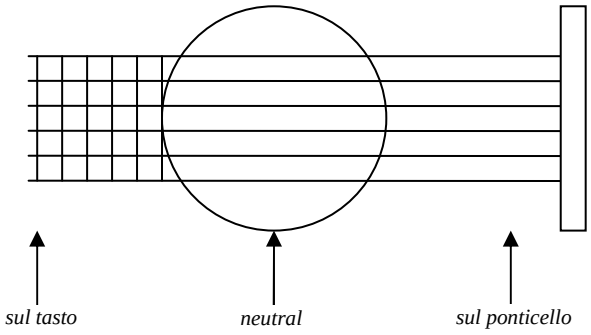


Figure 4.3: attack position of the string

Dynamics (*piano – forte*)

The timbre defining role of the dynamics-dimension is debatable. Volume, after all, is not regarded as a timbral dimension. However, in the case of a guitar the timbre of loud tones vary from soft ones in more dimensions than just the volume. This is because loud tones result in different resonance of the guitar's body and sometimes a buzzing sound (a noise component). Moreover the preceding finger movement is larger for a tone played *forte*, thereby irrevocably affecting the timbre.

Fingering (discrete dimension: *thumb*, *index*, *middle* or *ring*)

The finger used for a certain tone is often prescribed when applying a strumming technique that requires finger alteration or if the score incorporates fingering annotation. Sometimes the fingering configuration is fixed because a chord needs to be played. Therefore in most occasions the fingering is more a technical convention than a free expressive choice. However, every finger has its unique mass, texture, shape,

striking angle and power, thereby creating a subtly unique timbre. Recognizing this dimension would therefore be most useful in the application of the tutoring system (section 1.2.4), when for example finger alteration is to be tutored.

From the presented EDs, Exprimulator is tested with the RHPP. Variation in this ED results in distinctive classes of sound which can easily be recognized by ear. Therefore, manual evaluation is easier.

5. Design of Exprimulator

The experimental environment called Exprimulator has been designed, to meet the research goals and answer the research questions. While the tasks and responsibilities of this system already have been outlined briefly in chapter 2, this chapter goes more deeply into the underlying design and program requirements.

In Figure 5.1 the event structure of the two components (the classifier manager and the transcriber) that comprise Exprimulator are illustrated. The scheme clearly shows the responsibilities of each component. The logical operation procedure prescribes the creation of a training corpus initially. This training corpus is a two dimensional wavetable, in which every training instance is categorized per class according to the class labels defined by the user. These class labels are used by the transcriber as annotation labels in a later stage. From every instance of this wavetable feature vectors can be calculated. Such a feature calculation function is denoted by $\theta(Tone_{a,i})$. The composition of this function in terms of calculated audio descriptors can be configured by the user. Once for every tone in the wavetable the feature vector is calculated, classifiers based on several machine learning approaches can be created and trained to recognize the defined class distribution. The trained classifiers are used in a later stage by the transcriber for the annotation of transcriptions. The transcriber starts by loading one or more recorded song performances of which a transcription can be created by tone segmentation and pitch calculation. This note segmentation also enables separate feature calculation of every tone. The composition of these feature vectors needs to correspond with the classifier that is intended to be used for annotation. The classifiers that are trained with an identical feature vector composition can now be used to provide annotation by predicting the class within an expressive dimension.

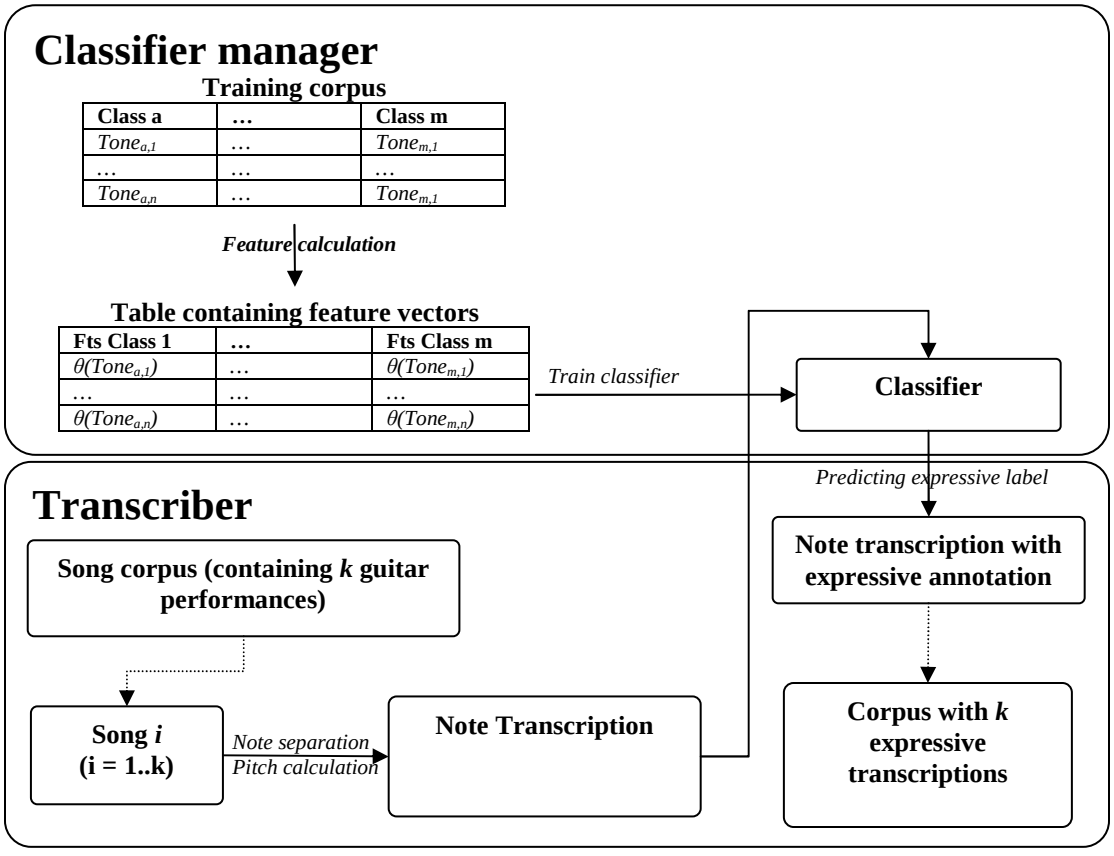


Figure 5.1: Structure of *Exprimulator*

In Figure 5.2 the procession of data structures managed by the classifier manager and the transcriber is illustrated. The user and processors can affect state transitions of these data structures which are denoted by dashed arrows.

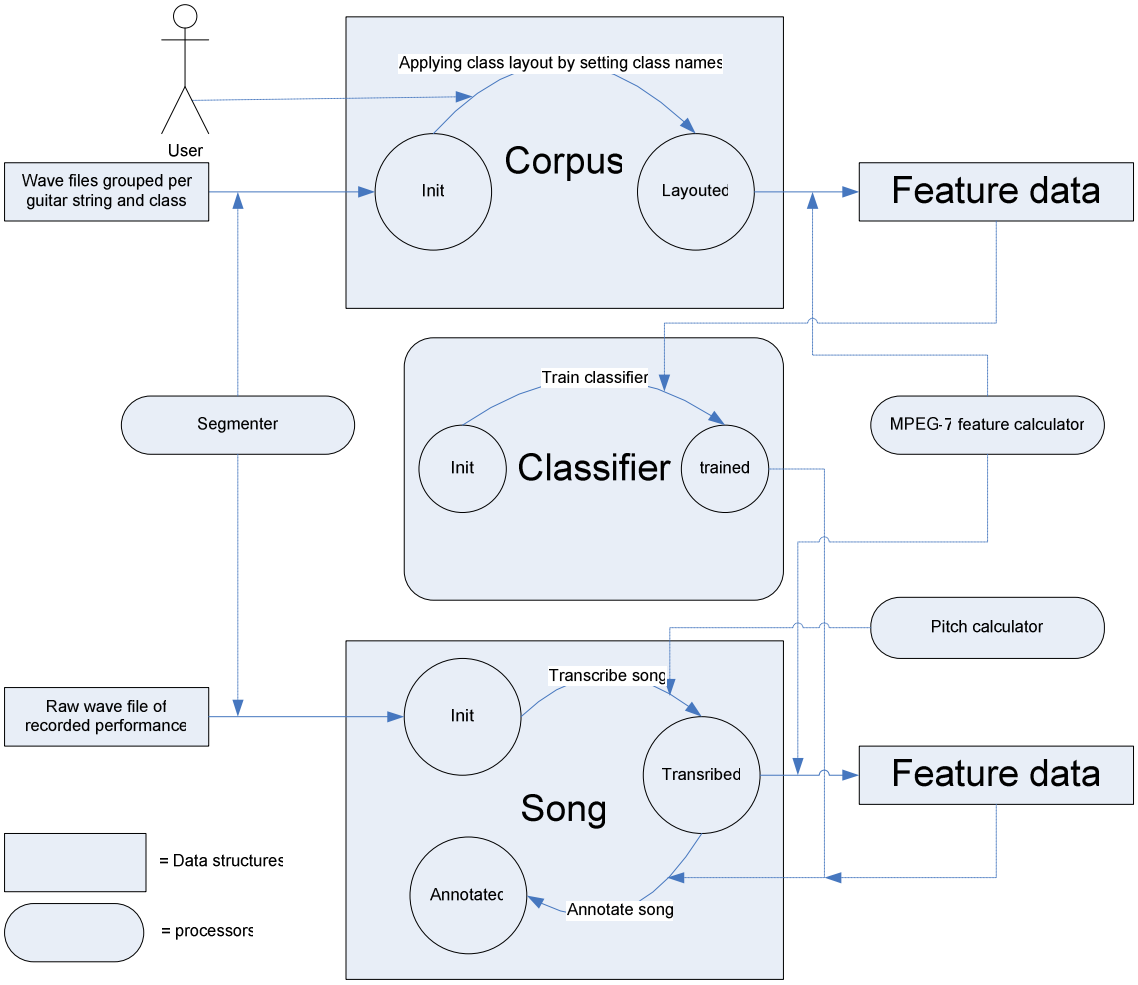


Figure 5.2: Interaction between entities and transitions

5.1. Requirements

The goals and approach presented in chapter 2 give rise to the following requirements for the classifier manager and transcriber component of Exprimulator. The functional requirements that are necessary to come to annotated transcriptions are presented in section 5.1.1, whereas the next subsection states the functional requirements for Exprimulator's to serve as an experimental tool. Section 5.1.3 presents some measures to ensure easy operation. At last non-functional requirements are given that have to ensure durability and extendibility of Exprimulator.

5.1.1. Functional requirements for producing annotated transcriptions

The final version of Exprimulator must be capable of performing the following tasks:

- Creating training corpora by automatically splitting wave files containing repetitions of training tones into separate tones.
- Ordering tones present in corpus in classes. This class layout is intended to be learned by the classifiers.
- Creating, parameterizing and training classifiers.
- Configurable calculation of feature vectors for tones present in training corpus.
- Transcribing songs, i.e. detect note onsets and pitches. These detection algorithms must be parameterizable via Exprimulator's GUI.
- Calculating feature vectors for tones present in songs. Therefore the user has to be capable of specifying the intended classifiers by which annotating the song, in order to calculate the correct feature vectors.
- Annotating songs with a trained classifier.
- Setting desired annotation for notes of song, if this information is available for the recordings.
- Displaying classifier performance by calculating percentage of correctly annotated notes.

5.1.2. Functional requirements regarding experimenting and evaluation

- Visualizing waveforms of corpus tones in order to be able to check wrong onset calculation.
- Providing means to correct miscalculated or missed onsets by splitting and appending tones, or by repositioning the onset.
- Flexibly including or excluding misplayed tones for the training of classifiers.
- Providing insight into feature data by visualizing feature vectors grouped by class.
- Providing insight into underlying model of classifiers by visualizing rules, statistics, decision trees or networks.
- Visualizing annotated transcriptions in a representation resembling a MIDI sequence with textual annotation referring to the predicted classes.
- Visualizing performance statistics per classifier that annotated a song (number of notes, correctly classified notes, annotation performance).
- The waveforms of performed songs must be viewable together with the calculated note onsets. In this way falsely calculated note onsets can be detected.
- Select notes in the waveform of a song performance. In this way certain fragments can be evaluated by listening or by observing the respective note statistics.
- Evaluating and comparing the performance of classifiers.

5.1.3. Usability

- Carrying out repetitive tasks efficiently for large amounts of data: tone segmentation, pitch calculation and calculation of feature descriptors of multiple tones of a training corpus, constructing and training multiple classifiers and annotation of multiple songs with multiple classifiers.
- Sequential tasks must be automatized. For example, a common sequence of actions within the transcriber is *transcribing → calculating feature vectors → annotating*. This sequence has to be aggregated in order to ease operation.

5.1.4. Non-functional requirements

- Exprimulator has to be general in its design so it can be applied to any musical note-level sound classification problem. The sound content that the classifier manager is intended to process must not be restricted to guitar sound. Therefore general purpose audio descriptors have to be implemented, so that in theory any set of sound classes can be recognized, provided that they are acoustically separable. The same applies to the transcriber: it has to be capable of transcribing any musical performance and not only guitar performances.
- Exprimulator must be easily extendable and modifiable by consistent programming.

6. Design machine learning methodology

6.1. Design of training corpus

The initial step for obtaining machine learners that recognize musical expression is to create a training corpus. This step is a crucial one as the class configuration embedded in the corpus is learned by classifiers when trained. This learned class configuration is hoped to represent the more general difference between playing techniques. Two training corpora were created to be able to realize playing technique recognition, and to test Exprimulator's supporting features for realizing this goal. One initial training corpus with less training material was created to get a provisory idea about the adequacy of the corpus and ideas for improvements of the extensive corpus. The extended corpus was used for final experiments and ought to be representative enough for classifiers to annotate a larger amount of song performances. A more detailed description of these two corpora is given in section 7.2. The design of the corpora raises some primary questions. Important considerations for answering these questions are discussed below, whereas definitive answers are given in section 7.2.

- **What expressive dimension is to be recognized?**

  Exprimulator allows for creation of corpora with any imaginable sound class configuration. However, this does not guarantee accurate recognition by a classifier of these sound classes in a later stage. Differences between the sound classes can be simply too subtle to recognize. A good initial guideline within our context is to check whether the sound classes can be distinguished by ear. If not, the chances of successful recognition are naturally smaller. It is also debatable if recognizing inaudible sound differences (if they exist at all) is desirable for annotating musical transcriptions.

  Furthermore, the context of recognizing musical expression narrows the possible sound class configurations. The sound classes need to have some relation with actual playing techniques that can be realized on a guitar. Moreover, the playing techniques need to be known and accepted by musicians to realize expression, and they should be arrangeable on one single expressive dimension.

  With respect to the goals of this thesis, it is necessary to find an expressive dimension that demonstrates the possibility of expression recognition. Therefore it is important to not choose a dimension that is too easy to classify, as this does not guarantee succession of more difficult recognition tasks.

- **How are classes distributed over the expressive dimensions?**

  Dimensions are defined as the range of possible playing configurations between two complementary playing techniques; hence a training corpus for a playing dimension should contain at least the two classes of complementary playing techniques. In case there is a lot of timbral variation possible in between the two complementary playing techniques, intermediate classes can be created. In this manner the classifier can interpolate between the two extreme classes more smoothly. However, caution should be taken that these inner classes do contribute to a more accurate classifier, and that the distances between the classes are large enough to recognize.

- **What interfering dimensions should be incorporated in the training corpus to reflect real performances as much as possible?**

  For this paragraph, we define an interfering dimension as any other dimension than the expressive dimension which has to be classified, and might disrupt correct expression recognition. This is formalized as follows:

  - $A = \{a_1, ..., a_m\}, B = \{b_1, ..., b_n\}$: *A* and *B* are playing dimensions, defined by a set of labels denoting the playing modes that comprise the dimension. Dimension *A* and *B* can be applied simultaneously.

- $tone_a$ denotes a tone realized by applying playing mode $a \in A$, while $tone_{a \times b}$ denotes a tone by applying playing mode $a \in A$ simultaneously with playing mode $b \in B$
- $pred_A(c_j, \theta(tone_a))$: The predicted playing mode within dimension A, by classifier c_j given feature vector $\theta(tone_a)$. Correct prediction should yield label $des_A(tone_a) = a$, a manually defined supervision function.

Using the preceding definitions, a playing dimension *B* is defined as interfering, if the following statement is true:

$$\exists a \in A, \exists b \in B : pred_A(c_j, \theta(tone_a)) = des_A(tone_a) \wedge pred_A(c_j, \theta(tone_{a \times b})) \neq des_A(tone_a)$$

It is important to decide which interfering dimensions should be incorporated in the corpus to enable a classifier to generalize over these dimensions when test data is presented. The idea of this incorporation is to create a training corpus Φ that contains variation in these dimensions, as follows:

$\Phi = \{tone_{a \times b \times c} \mid a \leftarrow A, b \leftarrow B, c \leftarrow C\}$, presuming that playing dimensions *B* and *C* are interfering dimensions. Examples of possible candidate interfering dimensions that a classifier irrevocably has to deal with are pitch, loudness, note duration, etc. An important issue is what ranges of these dimensions have to be incorporated in order for a classifier to generalize. Another point of interest is that the number of instances in the corpus grows rapidly if more playing dimensions are added ($|\Phi| = |A| \cdot |B| \cdot |C|$)

- **How many instances per class?**

The number of instances in a corpus is important for the classifier to be able to generalize, for the computation time and for the risk of overfitting. The idea behind incorporating more instances in a corpus is to improve the classifier's ability to generalize over the natural variation and noise that test data bears in comparison with the training data. However, a corpus that is too big could bring along unnecessary practical work (human factors like recording and creation of corpus), long training times, unwanted effects such as overfitting and the trained models could become too complicated. In case one has a large enough corpus at his disposal, several decreasingly smaller fractions of the corpus can be created to discover an optimal size. In case the resulting performance curve still rises when extrapolated, it follows that the full reference corpus of which the fractions were formed was not big enough.

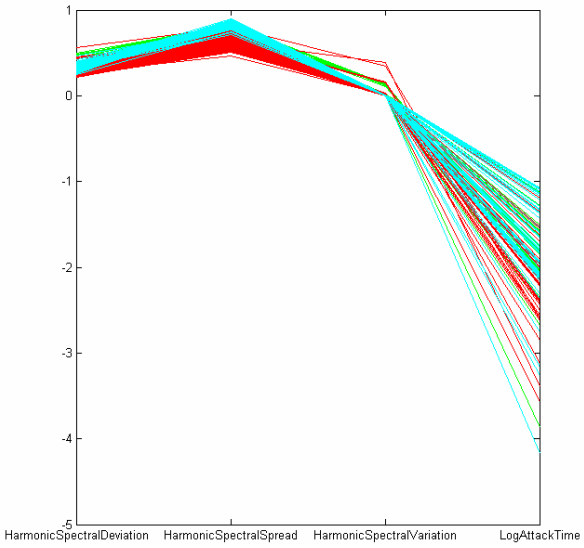


Figure 6.1: Selection of MPEG-7 descriptors of initial training corpus

6.2. Feature vectors

In chapter 3 the MPEG-7 standard was introduced as an elegant framework for audio description, because of its accurateness, compatibility, widespread use within audio content description and the unified implementation. However, not all the descriptors within the MPEG-7 toolbox are regarded useful within the context of this particular project, i.e. RHPP recognition. The most obvious set of descriptors to start with, would be a combination of Timbral Spectral and Timbral Temporal descriptors (see Figure 3.1). These are *Log Attack Time*, *Temporal Centroid*, *Harmonic Spectral Centroid*, *Harmonic Spectral Deviation*, *Harmonic Spectral Spread*, *Harmonic Spectral Variation* and *Spectral Centroid*. These descriptors help to describe notions as *attack*, *brightness* and *richness* in a quantitative way.

To verify the suitability of these descriptors for the class configuration of the RHPP, the *HarmonicSpectralDeviation*, *HarmonicSpectralSpread*, *HarmonicSpectralVariation* and *LogAttackTime* descriptors were calculated from the initial training corpus of section 7.2.1. A glance at Figure 6.1 shows that a selection of these descriptors already yields delimited clusters of feature vectors. In this plot, the classes are grouped by colors; green depicts *neutral* instances, red *sul ponticello* and blue *sul tasto*. It must be kept in mind that this is only an informal evaluation of the audio descriptors, and no final conclusions can be drawn upon these results. Visually analyzing feature vectors shows at any rate that the MPEG-7 vectors cluster better than vectors with audio descriptors implemented by the author on basis of formulae from literature.

6.2.1. Calculation of feature vectors

Feature vectors consist of temporal and spectral MPEG-7 descriptors. These temporal descriptors are global descriptors calculated from a certain range (the beginning 229 ms) of a tone. For the spectral descriptors, the tone is segmented initially in blocks of 30ms (the standard in MPEG-7) of which spectra are calculated. Once for every block these descriptors have been obtained, the mean is calculated in order to have one value per descriptor that can be inserted in the feature vector (see Figure 6.2).

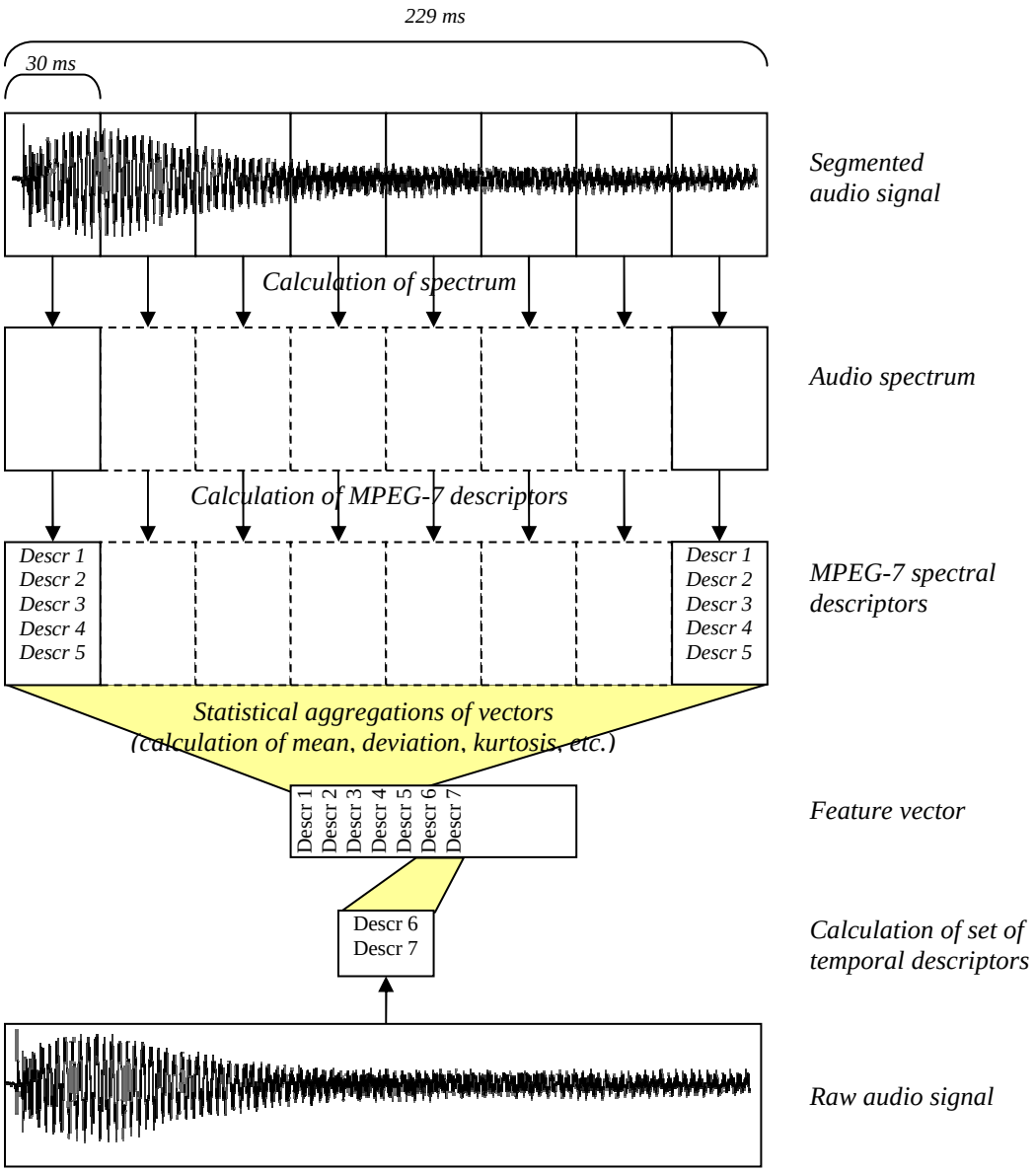


Figure 6.2: Construction of feature vector

6.3. Classifiers

In this section available classifier categories are outlined in order to determine which ones are the most appropriate for playing technique recognition. This ordering is analogue to the categories identified by the WEKA-toolbox. Once the general idea of the functioning and characteristics of these different machine learning approaches are clear, an overview of related work on sound classification is presented in section 6.3.2. The related projects share the property that performances of different classifiers are compared for each sound classification task, together with an evaluation of the best performing classifier.

6.3.1. Overview of classifier categories

We start with outlining the characteristics, applications, disadvantages and advantages of a major selection of the WEKA classifiers. For this section the WEKA API [36] as well as Wikipedia [37] has been consulted. The general information provided in this section is not determinative for the WEKA

classifiers that are evaluated, but they can give an idea how well their characteristics match with the particular sound classification problem of this project.

6.3.1.1. Decision trees

Decision trees are hierarchical classifiers which arrange decisions in a tree configuration where nodes represent a decision (often based on a Boolean function on a selected feature attribute) that can result in several outcomes (mostly two outcomes, in case of binary decision trees), corresponding with the branches.

Decision trees are built by starting with the entire set of training instances, and splitting these according to the most optimal partition according to some heuristic. These splits are represented by the tree's branches. This process is recursively repeated for the subsets of the partitions, until either a subset is a subset of the target class, or there are a minimum number of instances in the subset. A drawback of decision trees is that they are tended to overfit training data, especially when the training data is noisy. A tree is overfitted if there exists some other tree that fits the training data less, but performs better on test instances. Overfitting a tree can be overcome by stopping to expand the tree earlier, or by removing nodes of a full grown tree if this does not result in decreased accuracy over a test set (pruning).

Decision trees have several advantages: they are easy to interpret, and can be easily converted to a set of production rules. Decision trees are also general in the sense that no a priori assumptions are made on the nature of the data. Furthermore, little data preparation (like normalization, removing blank attribute values) is required. The disadvantages of decision trees are their sensitivity for noise in the training data. Slight variations can result in a selection of different attributes at the nodes of the trees, which can have significant influence on the descendant subtrees if this node is on a high level in the tree. Besides this, trees can become quite complex since the splits of the numeric training instances are binary.

One commonly used decision tree in the field of machine learning and pattern recognition is the J48-tree [3, 38-40], which is a supervised classification algorithm. It is also used in the more related field of instrument recognition [26]. Logistic model trees combine the benefits of logistic regression models with decision trees, by integrating these logistic regression models at the leaves of the tree. The logistic models in the leaves are the result of refinement of the logistic models in the higher level nodes of the tree. In some cases, better performance can be achieved by constituting a forest consisting of several decision trees. The RandomForest classifier follows this approach, and consists of more than one RandomTree (RandomTree is a classifier that considers K random features at each node).

6.3.1.2. Support Vector Machines (SVMs)

While they were originally proposed by Boser, Guyon and Vapnik in 1992, SVMs gained popularity in the late 1990s. Support vector machines are based on *structural risk minimization* paradigm, which implies finding a hypothesis which guarantees the lowest true error. An SVM performs classification by building an *N*-dimensional hyper plane that separates instances in two categories most optimal, which means that the hyper plane is located at maximum distance of the training instances. This is why SVMs are also called maximum margin classifiers.

Support vector machines are used for different pattern recognition tasks, such as speaker identification, face detection, text recognition [41], and genomic data. In various fields, SVMs are the best performing classifiers.

SVMs can classify problems with a high feature dimension, and they are also protected against overfitting. Besides feature vectors, more complex data structures can serve as an input for SVMs, such as graphs, sequences and relational data. Drawbacks of SVMs are that they do not provide insight in the underlying model and they classify linearly. Furthermore, the appropriate kernel-function (that can be polynomial, radial basis or Gaussian radial) and parameters are hard to determine, meaning that this is often performed in a try-and-see manner.

6.3.1.3. Neural networks

A Neural Network is a configuration of interconnected neurons, which can approximate a particular function. It is inspired by biological nervous systems. Neural networks consist of several layers, the input,

output and one or more hidden layers that contain neurons with a certain bias function. The neurons of a particular layer can be connected to the neurons of the adjacent layer with weights that are determined in the training phase.

Neural networks are successfully applied in speech recognition, image analysis (for example hand writing recognition) and adaptive control, and are used for the construction of software agents (in computer and video games) or autonomous robots.

Neural networks have the advantage that through the parallel computation, they can realize fast prediction. On the other hand, there is no insight in the trained model, and the training process is relatively slow. Like SVMs, Neural Networks are hard to fine-tune, and there are no common methodologies to determine the number of layers and neurons. Unlike with SVMs, there is an increased risk of overfitting.

6.3.1.4. Bayesian learning

Bayesian networks are a widely used statistical classification approach, in the fields of speech recognition, sound- and text classification. In Bayesian networks, variables are typically arranged as nodes in a graph structure, where the edges specify the conditional dependencies between the variables.

A simpler and more generic approach of the Bayes' theorem is the Naive Bayes classifier. It assumes that the individual attributes of the feature vectors are independent (hence the adjective Naive). It predicts instances on basis of maximum likelihood. Despite its simplicity and the naive independence assumption, it tends work well and even outperforms other classifiers in some occasions. In cases the independence assumption is violated, it often predicts the correct maximum-probability class. It also is robust to noise as it is not focused on completely fitting the training data. This has also the drawback that there is no guaranteed consistency with the training data.

6.3.1.5. Lazy learning

Lazy learning is a technique also known as instance based learning that does not constitute a model on basis of training data, but predicts training instances on basis of the distance or similarity on the training instances. This distance is mostly measured by the Euclidian distance function. As follows, lazy classifiers do not incorporate an actual training phase, as they just memorize the instances. This implicates that in the prediction phase the actual distance computation occurs, resulting in a slower classification of test instances. Amongst classification applications, instance based learning can also be applied in case-based planning and case-based reasoning in law and business.

K* is an instance based learner that uses an entropic distance measure for determining the similarity between test and training instances. It is a very simple method, while remaining accurate at the same time. However, the K* classifier treats all attributes as equally important, which might skew classification results. Another drawback is that there is no model which can be evaluated.

The k-Nearest Neighbour algorithm (k-NN) is a pattern recognition algorithm that assumes that instances generally are located in close proximity to other instances of the same class. It classifies instances on basis of their distance between *k* other training examples. The classifier is suitable for small amounts of training data and has a strong consistency: the algorithm is guaranteed to give an error rate no worse than twice the Bayes error rate. The algorithm is sensitive for noise however, as query-instances are compared with training instances. This effect of noise is reduced as the value for *k* is increased.

6.3.1.6. Rule learning

Closely related to decision trees are rule learners, which are applied to extract implication and correlation rules from large data sets. Rules can be obtained in numerous ways, for example by conversion of decision trees or Neural Networks. Decision trees can be converted to a set of rules by creating a rule for all the paths from the root to the leaf, which is followed by post-pruning where unnecessary antecedents are removed. Another approach is sequential covering, in which rules are learned one at a time, each rule covering a set of instances belonging to one class. This process is repeated for the other instances, until all instances are covered. Rules can be learned following the bottom-up or top-down approach. With the

bottom-up approach, a specific rule is used as a starting point in order to obtain a more generic one, as opposed to the top-down approach.

The rules generated by a rule classifier are easily interpretable by humans. This is why rule learners are frequently used in data mining problems where understandable patterns need to be discovered in large amounts of data. It is commonly applied in linguistics (text classification, webpage classification) and game learning. It has also been used recently for the classification of chemical compounds as cancer causing, based on their molecular structure and chemical characteristics.

Many existing rule learners are computationally expensive, especially on noisy data. Like decision trees, rule learners are sensitive for overfitting, which can also be overcome by pruning.

6.3.2. Literature review of classifiers used for sound classifying problems

In projects that are concerned with classifying sound on basis of timbre, several classifiers have been evaluated on their performance on large sets of training data. In the paper *Audio-based gender identification using bootstrapping* [38], a methodology for gender identification has been presented. Gender identification is useful for the improvement of speech recognition as well as video indexing. From the following six classifying approaches; Naive Bayes, Nearest Neighbor, Backpropagation Neural Network, Decision Tree, Support Vector Machine and Logistic Regression, the Neural Network consistently performed the best. Another speech related project, *Robust Recognition of Emotion from Speech* [32], evaluated the major WEKA-classifiers on classification of emotion from speech. Emotion has been classified on a rough level as positive or negative. On a low-level scale the positive category has been split in delight or flow (confident, encouragement), and the negative category in confusion or frustration. Because too little training was used, no significant performance differences between classifiers were found.

In *Retrieval of percussion gestures using timbre classification techniques* [20] it has been attempted to recognize five distinct classes of drum timbres, realized by six different striking positions of a snare drum: rimshot, brush stroke, center, near-center, halfway, near-edge and edge. Class recognition has been attempted with three different classifiers: k-Nearest Neighbor decision trees, Support Vector Machines (SVMs) and Neural Networks. Amongst the two others, the decision tree performed consistently the best in several test cases, whereas the Neural Network achieved the highest performance in a specific test case. The recognition of percussion gestures is considered closely related to the recognition of the RHPP of this project. Another drum sound classification project is described in *Automatic Classification of Drum Sounds: A Comparison of Feature Selection Methods and Classification Techniques* [21]. Drum sounds brought forth by different types of kick- and snare drums, toms, hihats and cymbals were classified. For this task, a 1-Nearest Neighbour tree, K* instance-based classifier, C4.5-tree, PART rule learner and Canonical Discriminant Analysis were explored. For detailed sub-category classification K* appeared to be the best choice, whereas CDA performed better than the others for super-category classification.

Besides the specific task of recognizing drum sounds, more generic classification has been explored in various projects relating instrument classification. In *Blind Signal Separation of Similar Pitches and Instruments in a Noisy Polyphonic Domain* [39], separation of two harmonic signals of different instruments was attempted, in a noisy environment. Four classifiers were compared to achieve this task: Tree J48, Logistic Regression Model, Bayesian Network, and Locally Weighted Learning. Locally Weighted Learning outperformed the other classifiers. Another instrument recognition project, *Differentiated harmonic feature analysis on music information retrieval for instrument recognition* [26], focused on the classification of the instrument families woodwind and string instruments. In the following stage, four woodwind instruments and four string instruments were classified with different classifiers. Bayesian Networks, Logistic Regression Model, Decision Tree J-48 and Locally weighted learning were compared, resulting in the best performance for the Logistic Regression Model for the instrument families, whereas the J48-tree performed equally well as the Logistic Regression Model for the recognition within the two instrument families.

The recognition of musical genre can be regarded as classification on an even higher level than instrument recognition. *Classification of musical genre; a machine learning approach*, is a project which attempted to classify six musical genres from analysis of MIDI file features [3]. Naive Bayes, Voting Feature Interval, PART rule learning, J48-trees, Nearest Neighbor rule-learning and JRip rule learning were

explored, yielding Naive Bayes as the most promising classifier. In *Musical Genre Classification Enhanced By Improved Source Separation Techniques* [4], more specialized genres were classified (four Greek musical genres) by extraction of rhythmic, timbral and pitch features from audio signals. In a comparison between NN-classifiers and Multi layer perceptrons the latter outperformed the former.

In *Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptors* [42] an algorithm for the determination of the meter (duple or triple) of musical audio signals is proposed. Kernel Density estimation, PART rule-induction, Neural Networks, 1-Nearest Neighbor (1-NN), C4.5-trees and SVMs were compared on their performance. Naive Bayes performed the best compared to the other classifiers.

As can be noticed, the classifiers being evaluated do not correspond amongst the different projects; neither do the outcomes of the most appropriate classifier. Besides that, the choice of the initial classifier-set to test with is not motivated. Therefore, for our project we have decided to evaluate several classifiers which are the most commonly used in papers that classify similarly sound content as in this project (drum timbre is more related to our project than gender identification for instance). There should also be similarity in complexity of the training data.

6.3.3. Selection of classifiers to test

As there is not any consistently chosen classifier for the classification problems of the previous section, we have been decided to evaluate most of the WEKA classifiers on the classification of playing techniques. It requires little effort to integrate these classifiers with Exprimulator, as the WEKA API can be directly invoked from Matlab, as it is written in Java. Some classifiers are excluded because they cannot handle the feature vectors or class labels. The ID3-tree classifier only interprets nominal attributes, whereas the Lazy Bayesian Rules classifier cannot handle undiscretized training data. In the case of the m5-tree on the other hand, only numeric attributes can be interpreted as class labels. The remaining classifiers that are used are depicted in Table 6.1, and are referred to throughout this course as set *C*.

WEKA category	**WEKA classifier name**
Decision trees	J48
Decision trees	NBTree
Decision trees	LMT
Decision trees	DecisionStump
Decision trees	RandomForest
Decision trees	RandomTree
Decision trees	REPTree
Lazy learning (instance based learning)	IBk
Lazy learning (instance based learning)	LWL
Lazy learning (instance based learning)	KStar
Bayes	BayesNet
Bayes	NaiveBayes
Bayes	NaiveBayesSimple
Functions	MultilayerPerceptron
Functions	SMO
Rule learning	JRip
Rule learning	NNge
Rule learning	OneR
Rule learning	PART
Rule learning	Ridor

Table 6.1: WEKA classifier used for performance comparison

6.3.4. Incontrollable interfering dimensions

As there can be interfering playing dimensions caused by the musician like discussed in section 6.1, there are also several incontrollable factors that can disrupt expression recognition. Although these factors are not incorporated in the training corpus, it is important to be aware of them, in order to keep them as constant as possible.

Recording settings

In the recording setup there are many factors that can influence the classification process. Every human performance that is recorded in a different session than the one in which the training corpus was recorded, is likely to differ in sound because of different recording settings. These settings can involve equalizer configurations of the guitar, amplifier or recorder. Besides that, there is the possibility to transform the signal by applying amplifier effects such as *chorus* or *reverb*, which can also be realized by the recorder (a computer in this project). All these factors can influence expression recognition. In the most optimistic case the degree of generality of the classifier is high enough to ignore these differences. However, for this project, we try to overcome this potential problem by keeping the recording setup as stable as possible between independent recording sessions. A better solution is to record test data in the same recording session as training data.

Instrument's characteristics

Another influencing factor is the instrument which is used to record performances. The goal of the eventual classifier is not to be able to generalize over different types of guitars, for example Spanish guitars and steel string guitars. The physical and acoustic properties of nylon strings (Spanish guitars) differ in such a degree from steel strings that it is unlikely if there can be a generalization at all. Our classifier was trained on steel string instruments and therefore the intention was to annotate steel string performances. However, within the category of steel strings, individual instruments can differ also drastically in sound characteristics. It is a challenge to generalize over as many different steel string guitars as possible, but this lies beyond the scope of this study.

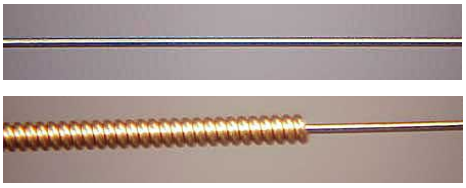


Figure 6.3: Plain steel string and wound steel string

Inconsistencies within one instrument

Within an instrument there are some factors that can influence sound characteristics. There is for example the difference in timbre between a fretted tone (a tone realized by pressing a fret on a string to create a different pitch) and an open string (no fret is pressed). Besides that, the two highest strings of a western guitar (B- and E-string) have different physical and acoustic properties than the other strings. Whereas the B- and the E-strings are made of plain steel, the other strings are made of a core wire, wound with a bronze wrap wire (see Figure 6.3). The reason of this wrapping is to give the strings more mass to realize lower tones. A side effect of this wrapping is however, difference in timbre.

Where some of the factors discussed above are beyond our scope, it is nonetheless a challenge to discover whether a classifier can be made general enough to overcome these factors. Some factors, such as different equalizer setting or instruments might be relatively easy to overcome by calibration. In case feature vectors of instrument A can be projected on the ones of instrument B by a simple linear translation, then this translation function is easily obtained by calibration. However, the chances for a classifier that detects very subtle timbre differences to generalize over different instruments are low.

A classifier should naturally be capable of generalizing the inconsistencies within one instrument for this project. If the classifier is not able to generalize between open string tones and fretted tones, there is the option to build separate classifiers for the two cases. However, this approach is impossible to maintain if too many of these inconsistencies occur when recognizing an expressive dimension. In that case, the possibility of overfitting should be investigated. For only a few inconsistencies, it loans to build separate classifiers, because the classifiers can be considered as offline systems that deserve some designing time to create optimal classifying results. This also means that there are no strict timing constraints for the training of the classifier.

7. Implementation

7.1. Implementation of Exprimulator

The interface of *Exprimulator* consists of two main components, the classifier manager and the transcriber. The features and possibilities of these sub systems are outlined in the following two subsections.

7.1.1. Classifier manager

The major tasks of the classifier manager are creating a training corpus, calculating the accompanying audio features and training classifiers that can be used as annotators by the transcriber. The corpus is created by loading raw wave files containing a continuous sequence of tones which are automatically segmented when loaded. The resulting separate wave fragments are called tones. Customizable feature vectors of every tone can be calculated which serve as training input for a selection of WEKA classifiers.

Figure 7.1 shows the GUI of the classifier. The wave data that constitutes a training corpus is presented in the first two left list boxes. These boxes display respectively the loaded wavefiles (1) as well as the segmented tones that comprise the selected wavefile (2). When tones are not properly segmented, they can be split by means of the wave plot (7).

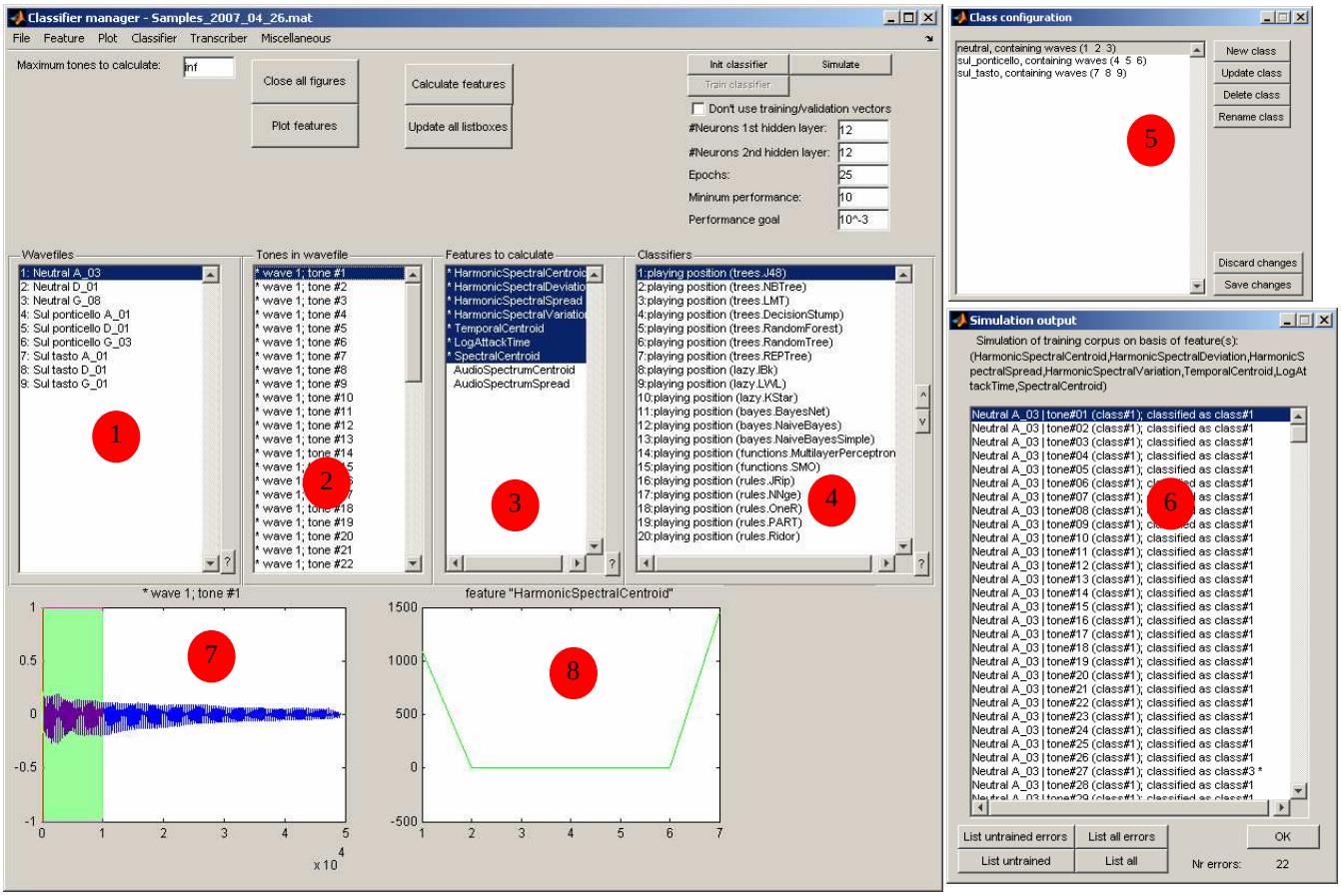


Figure 7.1: Classifier GUI together with simulation output and classifier dialog

The classifier manager also offers capabilities for deleting tones, or shifting the markers that indicate the note onsets. Besides visual inspection, all the separate tones can also be played back in order to detect tones that were not correctly played by the guitarist. Wavefiles and their tones can be grouped into

classes, which can be given names that are used by the transcriber to annotate songs, by means of dialog (5). In this example, the three classes used throughout this report are created (*neutral*, *sul ponticello* and *sul tasto*) by grouping the appropriate wave files. The customized class configuration can be assigned to a selected classifier of list box (4). The class configuration can be refined by including or excluding the tones using list box (2) (the asterisks denote inclusion). The resulting smaller training corpus leaves space for a test set that can be used for validation of the corpus. Once there is a classifier with an associated class layout, a selection of audio descriptors (3) can be used to calculate feature vectors of every tone that is present in the classifier's corpus. The feature vectors of the selected tone(s) are viewed in plot window (8). They are calculated over a specified range, visible as a green rectangle in (7). After initialization, the selected classifier can be trained with and evaluated on its training corpus, with dialog (6). In this dialog all the tones present in the training corpus are listed and as predicted with the selected classifier, hence also excluded tones are predicted. Wrong predictions are denoted with an asterisk.

All the classifiers with their training corpora configurations and feature data can be saved and loaded into Matlab-files, but the feature data can also be exported to the ARFF-file format that is compatible with WEKA.

7.1.2. Transcriber

The transcriber (Figure 7.2) is the part of Exprimulator that is responsible for transcription and annotation (Figure 7.3) of musical performances. A song corpus (2) is created by loading wave files that are segmented using the same procedure as the classifier manager, yielding the wave fragments visible in plot window (1). After tone segmentation the pitches visible in Figure 7.3 are calculated. For every song, any trained classifier from the classifier manager can now be loaded into the annotator box (3). Before an annotator can annotate a transcription, feature vectors of the song have to be computed. Naturally, these feature vectors have to be identical in composition to the ones that have been used for training the classifier. In this way they can be used as input for this classifier in order to predict a tone's class, that is presented as textual annotation in Figure 7.3. Incorrectly annotated notes are displayed in red. This prediction checking can only be done after the intended annotation is set for every note (5). With this comparison between the desired and automatic annotation, the statistics visible at (4) can also be computed. It displays the index of the selected note (shaded blue in (1)), the desired and predicted annotation and the total number of notes in the selected song. Furthermore it specifies the annotation performance (percentage of correctly annotated notes), and the absolute number of falsely and correctly annotated notes.

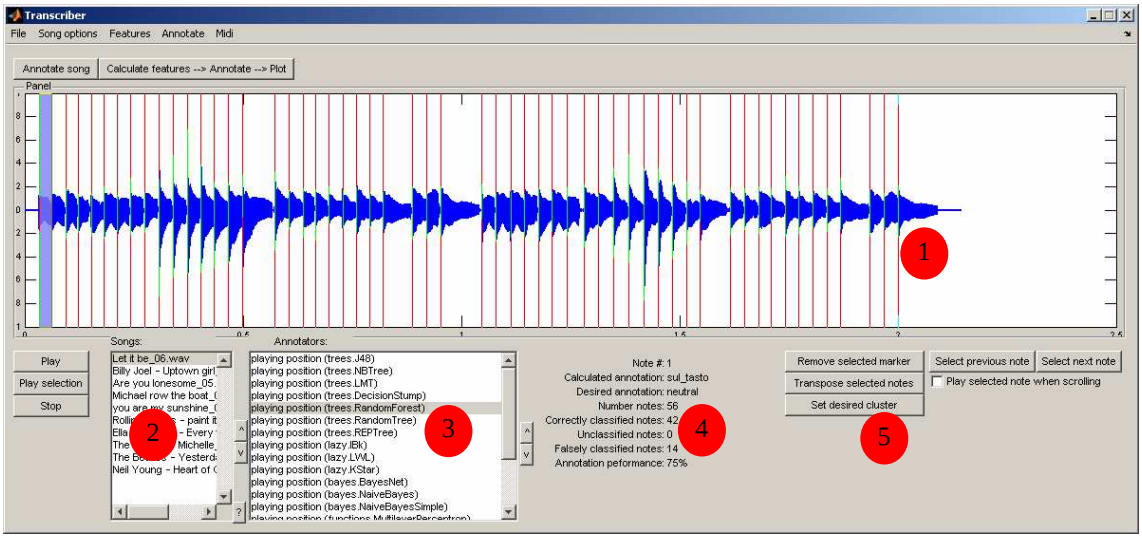


Figure 7.2: Transcriber GUI

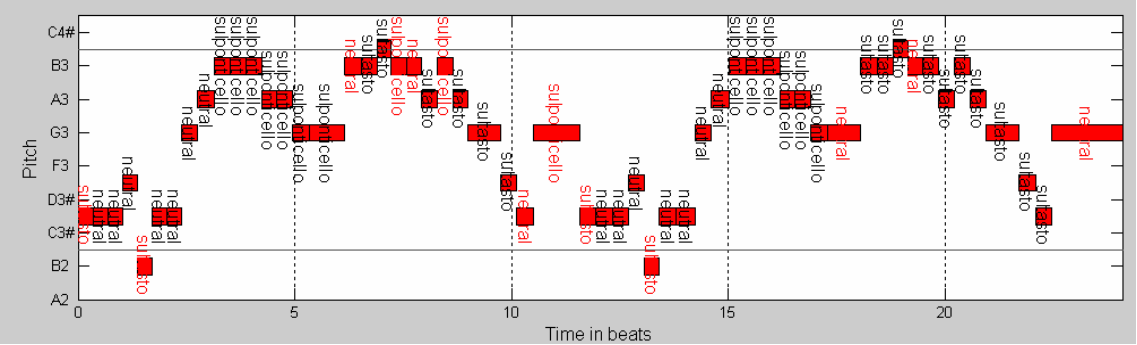


Figure 7.3: Annotated transcription generated by the transcriber

7.2. Creation of training corpora with Exprimulator

This section starts with a presentation of the layout of the corpora by which Exprimulator has been tested in chapter 8. Firstly, an initial training corpus has been created with the intention to prove the adequacy of MPEG-7 features and WEKA classifiers for musical sound classification. Hereafter, a more extensive corpus was created that served as training material for classifiers which were tested on a larger set of representative songs. This larger song corpus should enable conclusions about performance deviation among classifiers. In the subsequent sections, the practical realization of the corpora is outlined and the questions raised in section 6.1 are answered.

7.2.1. Test case: creating an initial training corpus with Exprimulator

A basic training corpus has been built to test Exprimulator's capabilities of creating corpora (see Figure 7.4). It was also used to insure an adequate course of the subsequent stages in the Exprimulator-process (the calculation of feature vectors → the training of classifiers → the annotating of songs with the trained classifiers). For the initial performance test the major part of the WEKA classifiers was used. This gained insight in the suitability of the individual classifiers for classifying the RHPP dimension. It was also expected that this test would expose structural performance differences between categories of classifiers. The initial corpus was tested with only one song *Au Claire de la Lune*. Naturally, these test results could not be used to deduce conclusions about differences in classifier performances. The test was merely carried out to provide an indication if it would be possible for MPEG-7 feature vectors to provide an adequate annotation performance (> 80%) in combination with any WEKA-classifier. In relation to this initial corpus the following questions were asked, that are answered in section 7.3.3.1:

1. Did Exprimulator adequately load and separate compound wave files into separate instances?
2. Did the chosen MPEG-7 feature configuration in combination with the selected WEKA classifiers provide a satisfying annotation performance for the song *Au Claire de la Lune*?

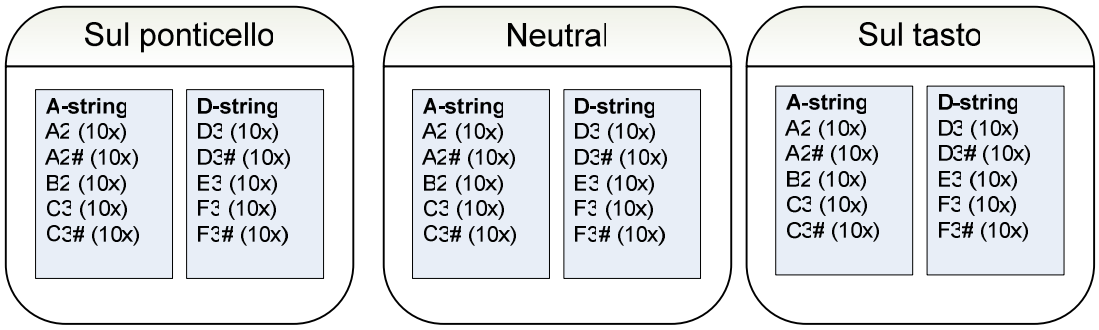


Figure 7.4: Initial training corpus

The corpus contained all the unique tones of the A-string and D-string, meaning the tones produced by the first five frets on the string as well as the open string. The inclusion of the chromatic range of tones that encompass the tones present in *Au Claire de la Lune*, is expected to ensure generalization over pitch by the classifier. The classifier is also expected to generalize over the timbral differences between fretted and unfretted tones (see section 6.3.4), and differences between strings.

7.2.2. Test case: creating a definite training corpus with Exprimulator

For the final tests presented in chapter 8, a more extensive corpus was created that spanned a wider range of tones, which enabled the annotation of songs with a larger range of tones (see Figure 7.5). Also the first six instead of five tones were incorporated per string, which enabled more flexibility in playing the songs.

The corpus was meant to be tested with 10 songs, consisting of approximately 60 tones each. This would result in 600 test instances by which the training corpus (consisting of 540 instances) could be tested, which in turn would provide adequate proof to deduce conclusions about the ratios between mutual classifiers.

In contrast to the initial training corpus, this time the tones were played without damping the other strings. When recording the initial training corpus, this artificial damping was applied to create a tone as clear and pure as possible without resonance of other strings. This effort was abandoned in order to reflect more accurate the way tones are played in the song performances.

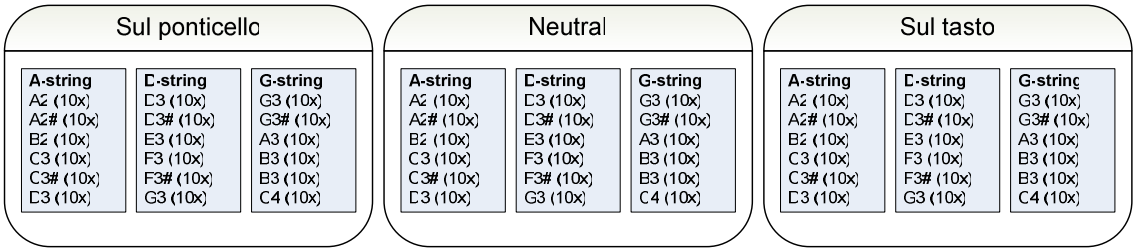


Figure 7.5: Definite training corpus

7.2.3. Design decisions

The design decisions that were made implicitly in sections 7.2.1 and 7.2.2 are accounted for in this section.

- **What expressive dimension was recognized?**

  As discussed before, the RHPP was used as a proof of concept to illustrate the possibility of annotating musical scores with a machine learning approach. The expressive dimension was divided sound classes, such that the difference would be audibly by human ear. Besides, the right hand playing position is a playing technique that is commonly used among guitarists. It is also an accepted way of expression in classical musical pieces, proved by the existence of the Italian annotative labels *sul tasto* and *sul ponticello*.

- **How were classes distributed over the dimensions?**

  The expressive dimension RHPP was divided in three classes: *sul tasto* – corresponding with playing left from the sound hole; *sul ponticello* – corresponding with playing right from the sound hole; and *neutral* – playing above the sound hole. The two extreme classes (*sul tasto* and *sul ponticello*) were the minimum classes required to span the RHPP dimension. The intermediate sound class *neutral* was chosen to reflect the standard RHPP, which is used often. Moreover, it enabled the classifiers to create a smoother interpolation function between audio features and RHPP. The creation of more intermediate classes was considered overkill, because there is no convention or annotation in musical literature that considers these playing positions. The resulting five sound classes would also be harder to distinguish from each other by human ear.

- **What interfering dimensions should be incorporated in the training corpus to reflect real performances as much as possible?**

The corpus from section 7.2.2 incorporated the interfering dimensions pitch, playing position deviation and playing the same tone with an open string or fretted. These are considered as the most common and prevalent dimensions when performing a song. An important notion is to keep the interfering dimensions as low as possible, because the number of instances increases explosively with each dimension added (see next paragraph).

- **How many instances per class?**

  The corpus from section 7.2.2 consisted of a total of 540 instances. This number was a result of the decision to incorporate interfering dimensions in the corpus. One of these dimensions was the pitch. A certain chromatic range of pitches had to be incorporated in order to insure generalization over pitch. 18 semitones, starting from A2 were considered to provide enough freedom for a guitarist to play melodies in. A natural jitter with respect to the median playing position of each class was added to resemble the inaccuracy of a guitarist when playing songs. 10 Of such slight playing position deviations were incorporated for each pitch. When multiplied, this resulted in 180 instances per class. The number of repetitions for the playing position jitter was a debatable choice. However it was expected that if a reduction of the corpus size – realized by reducing the number of repetitions of playing position jitter – with 20% (denoted by ϕ_{80}, where the index signifies the percentage of the full corpus) resulted in a significantly poorer classifier performance, an increase of the corpus size would be beneficial. To be able to draw such extrapolation conclusions, the following set of corpuses was constructed: $\Phi = \{\phi_{10} \subset \phi_{20} \subset \phi_{30} \subset \phi_{40} \subset \phi_{50} \subset \phi_{60} \subset \phi_{70} \subset \phi_{80} \subset \phi_{90} \subset \phi_{100}\}$. To obtain a performance curve, for each corpus the mean performance of all classifiers over the full song corpus $\overline{Perf\ (C,S)}$ was calculated. The set of corpuses was also used to evaluate the relation between the corpus size and the complexity of the trained rule and tree learners.

7.2.4. Practical realization of the training corpus

The corpus' tones were recorded with Cubase SX 2.0. The recording settings of this session are described in section 7.2.5. The settings were kept constant throughout the recording of the songs by which the training corpus was tested. It would be optimal if these settings would not interfere with correct classification, but the influence of these settings is beyond the scope of this research. This is why the settings were kept as constant as possible, and the songs were recorded in the same recording session as the corpus instances.

The tones of each string were recorded in separate compound wave files. This was convenient in order to keep the wave files manageable in size and recording mistakes would be more easily revisable. For another purpose, the corpus could be expanded or narrowed string-wise. The individual tones of a string were recorded by uninterruptedly recording notes of approximately 1 second each. This process was repeated for every playing mode of the RHPP and interfering dimensions that needed to be present in the corpus. The duration of 1 second was adequate for the calculation of feature vectors, which was executed over a range of the 10000 starting samples (= 0.23 sec.). Once for every string and playing technique a wave file was present, these were loaded into Exprimulator, which segmented the compound wave files into individual tones. The same segmentation process was used in a later stage to segment the tones of a performed song. In some exceptional cases two subsequent tones were not properly divided (for instance when playing *sul tasto*, which has a weaker onset). In these rare occasions, manual segmentation was used to divide these tones at locations where there appears to be an offset in the waveform. The resulted collection of separate tones was grouped into classes conforming the playing technique the tone was played with. This was done by assigning names to groups of appropriate instances manually. These names were used for the transcriber as annotation labels.

7.2.5. Recording settings

To ensure that the experiments can be revised, the technical details are given in this section. The performances and tones of the corpora were recorded with an acoustic steel-string guitar of the brand Lakewood with a B-Band pick-up that consisted of a microphone and a piëzo pick-up which were housed in the guitar's body. The guitar was connected to an AER Compact 60 amplifier that in turn was connected to the computer's sound card. The recordings had a sampling rate of *sr* = 44.100 Hz, a depth of

16 bits and were mono. These audio settings were kept constant throughout this study. The used sound card was an M-audio Audiophile 2496.

Recording settings of the guitar equalizer: Bass +8db, Mid +8db, Treble +8db, Vol +8db. Rate UST/AST = 6/6. Recording settings of the guitar amplifier: Bass 12', Middle 12', Treble 12', Gain 9', Master 9', Effect return 0', contour: off. Tones were recorded with Cubase, every tone lasted for approximately 1 second at least.

7.3. Implementation of machine learning methodology

This section describes the integration of the MPEG-7 audio toolbox with Exprimulator, as well as the selected audio descriptors with their calculation settings. For the WEKA classifiers it is outlined what parameters could be relevant with respect to their performance.

7.3.1. Calculating feature vectors

The feature vectors used for adequately describing the different sound classes of the RHPP were comprised of audio descriptors from the MPEG-7 audio toolbox. The algorithms for these descriptors were implemented for Matlab by Michael Casey from the University of London, in toolbox Matlab-XM. All the low level descriptors described in section 3.1.1 together with the description schemes, were implemented in this toolbox.

Feature vector calculation started with finding a suitable range from the source wave signal of a corpus tone. As the RHPP is concerned with the way the guitar is struck, the attack section of a tone would seem an obvious fragment for feature calculation. The beginning of a tone was determined by the segmentation algorithm, and was set at the maximum of each tone. To also incorporate the small section preceding this maximum for feature calculation, a fixed offset of -200 samples from this maximum was taken. From this point, the first 10.000 samples were used as input for the algorithms. This equals a duration of 0.23 sec., which is approximately a fourth of the length of the tones present in the corpus. As feature vector calculation of the song corpus is the same as for the training corpus, also 0.23 sec. of the tones of the songs had to be taken. For the few tones shorter than 0.23 sec. feature calculation was skipped, because no appropriate feature vectors could be calculated from these tones. The calculation range was not shortened for this purpose, because this decreased the performance of the classifiers. A long term solution could involve the creation of additional classifiers trained with smaller fragments of the corpus tones, to be able to classify the exceptionally short tones.

7.3.1.1. Settings for audio descriptor algorithms

For the calculation of the four Timbral Spectral audio *descriptors HarmonicSpectralCentroid*, *HarmonicSpectralDeviation*, *HarmonicSpectralSpread* and *HarmonicSpectralVariation*, short term Fourier Transforms (STFT) over subsequent time frames had to be calculated. These STFTs were calculated over windows of 3089 samples, shifted each time with 1545 samples (an overlap factor of 2). In short, the STFT is used to break down a time frame of a signal in constituent sinusoids of different frequencies. The formula is given below:

$$STFT(\{x[]\} \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[t]\omega[n-m]e^{-j\omega n}$$

In this formula $x[n]$ and $\omega[n]$ denote respectively the current time frame to be transformed and the used window function, a symmetric 'hill' function. In our case, the *hamming* function is used as a window function, as displayed in Figure 7.6.

In order to calculate the 4 Harmonic descriptors the harmonic peaks (multitudes of f_0, the fundamental frequency) were calculated from the STFTs, resulting in an STFT of $\frac{sr}{f_0}$ values, in which *sr* is the sample rate. The resulting reduced STFT is used for the calculation of the *centroid*, *deviation*, *spread* and *variation*. As there were 5 time frames of size 3089 in 10.000 samples with a window shift of 1545, each

descriptor produced 5 values for the entire segment of 10.000 samples. The vectors were made a scalar by calculating the mean.

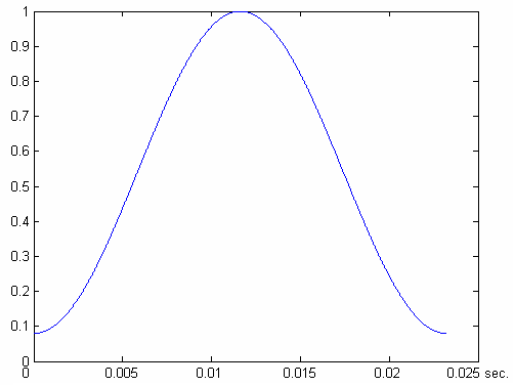


Figure 7.6: the Hamming function

In order to obtain the *SpectralCentroid* the power spectrum was calculated over the entire range of the calculation range of beginning 10.000 samples, with a resolution of 1024 Hz. From this spectrum the *centroid* was computed in the same way as for the harmonic STFT.

For the calculation of the *LogAttackTime* and *TemporalCentroid*, the energy curve (waveform contour) was calculated, by down-sampling with factor 3 and applying 20 Khz low-pass filter. This energy curve enabled the calculation of the logarithmic attack time as well as the temporal centroid.

7.3.2. Settings for classifiers

Presumably the default settings for the WEKA classifiers are not optimal for the recognition of playing techniques. Therefore, for a relevant set of best performing classifiers, the available tunable parameters that the WEKA API offers are explored. Adjustment of appropriate parameters might further increase annotation performance. Not all the parameters are considered relevant, thus only a selection is to be explored.

Concerning Boolean parameters, if there is an optimal setting that is beneficial for the classifier, WEKA has set it as default this way. Therefore these parameters do not allow for much performance gain. Examples of such parameters are pruning and subtree raising for decision trees. Another is the use of normalization for lazy learning methods such as *k*-nearest neighbours and locally weighted learning. Disabling these features is not expected to yield performance gain. Nevertheless, it is investigated what the effects of pruning are, to get an idea of the degree to which overfitting occurs.

Less obvious are the setting of numerical parameters and factors. These parameters motivate a more thorough exploration, as they can be attuned to a particular classification problem more closely. Examples of these are parameters that affect the structure of classifiers such as the number of trees for the RandomForest classifier or the number of hidden layers and neurons of a Neural Network. Other numerical parameters can be related to the mode of operation, such as the number of folds of cross validation for the REP-tree or the number of the number of used neighbors the *k*-nearest neighbours classifier uses for prediction.

Finally there are more structural choices that can be made, such as those which affect the kernel function (polynomial or radial basis) of an SVM or the base classifier of the Locally Weighted Learner. For the LWL classifier the default base classifier is the rather primitive DecisionStump. It can be explored if more complex base classifiers can contribute to increased classification accuracy.

7.3.3. Measurable results of experiments

This section presents testing procedures that are discussed in the experimental chapter 8. These procedures are carried out with the set of classifiers from section 6.3.3, trained by the definite training corpus from 7.2.2 and tested with the song corpus of 8.1.1. The procedures include metrics to:

- Cross validate the classifiers on the definite training corpus
- Determine the classifier's annotation performance
- Determine the best classifier
- Determine the classifier consistency
- Determine the optimal corpus size
- Discover causal relation between note characteristics and misclassification

- Interpret learned models

Cross validation classifiers:

Before proceeding to the testing of classifiers with actual performances, they can also be cross-validated by material of the corpus itself. Such an initial cross validation can indicate if the class configuration embedded in the training corpus is learnable by classifiers with the used feature vectors. In a later stage the cross validation performance of a classifier can be compared with the annotation performance on the song corpus, which reveals the classifier's generalization capacity.

Determination of the classifier's annotation performance:

The classifier's annotation performance denotes the percentage of correctly classified tones of the song corpus by a classifier (misclassified tones are not taken into account). The annotation performance is used to determine the set of classifiers which perform significantly better than the rest.

The annotation performance of classifier $c \in C$ on song *s* is calculated as follows:

$$Perf(c,s) = 100 \cdot \frac{\sum_{note_i \in s} Correct(c, note_i)}{|s| - \sum_{note_i \in s} Unclassified(c, note_i)}$$

where:

$$Correct(c, note) = 1, if \;\; pred_A(c, \theta(note)) = des_A(a)$$
$$0, otherwise$$

$$Unclassified(c, note) = 1, \;\; if \;\;\; pred_A(c, \theta(note)) = null$$
$$0, \;\; otherwise$$

The circumstances, under which unclassified tones can occur, are outlined in section 7.3.3.2.

The annotation performance of classifier *c* on a song corpus $S = \{s_1, ..., s_n\}$ is defined as the mean of all the annotation performances of *c* on the songs in *S*

$$\overline{Perf(c,S)} = \overline{\{Perf(c,s_i)\}_{s_i \in S}}$$

This annotation performance over a song corpus is used for comparing classifiers; under the condition that song corpus *S* is large enough and contains representative songs.

The standard deviation, $\sigma(Perf(c,S)) = \sigma(\{Perf(c,s_i)\}_{s_i \in S})$, provides an indication of the consistency by which the classifier performed on the song corpus.

Selection of best performing classifiers:

For more thorough investigation of a small set of classifiers, we have to exclude classifiers that performed worse than a certain criterion. The best performing classifiers *Best(C,S)* are defined as those whose annotation performance does not lie beneath the annotation performance of the best classifier c_{max} minus its standard deviation:

$$c \in Best(C,S) \Leftrightarrow \overline{Perf(c,S)} \geq \overline{Perf(c_{\max},S)} - \sigma(Perf(c_{\max},S)) \wedge c \in C$$
$$\vee \; c = c_{\max} \wedge c \in C$$
$$where$$
$$\tilde{c} = c_{\max} \Leftrightarrow \forall c \in C \setminus \tilde{c} : Perf(\tilde{c},S) > Perf(c,S) \wedge \tilde{c} \in C$$

The performances of the remaining classifiers that are not in *Best(C,S)* are too low to be increased by adjusting parameters, and are therefore considered to be not relevant for parameter tuning.

Corpus evaluation:

To determine an appropriate size for the training corpus, the annotation performance $\overline{Perf(c,S)}$ is calculated for every classifier trained with the different corpus fractions defined in section 7.2.3. This sequence of performances is plotted against the corpus sizes to determine the location where the performance converges to a certain optimum. This convergence point should be located at the most optimal corpus. Differences between convergence points among different classifiers reveal differences in the quantity of training data that is necessary for generalization.

Discovering relations between note characteristics and misclassification:

By investigating the relation between duration, pitch and playing technique on one hand and relative frequency of misclassification on the other, exceptional note characteristics can be ascertained. A confusion matrix reveals whether all playing techniques are equally often misclassified or not. If this distribution appears to be skewed it is interesting to look at the playing technique which is misclassified most often, and the playing technique that is predicted most often incorrectly. This relation tells which playing techniques are most likely to be mixed up with each other. This confusion might be caused by unrepresentative training corpora or inadequate feature vectors.

The relation between pitch/note duration and misclassification can be investigated to see whether exceptional cases deserve to be treated by separate classifiers. In case a certain set of pitches are misclassified more often relatively, this can be due to structural timbre differences. An example set of structurally different tones are for example open tones as opposed to fretted tones. If there is an increased chance of misclassifying short tones, this can be ascertained in like manner. This hypothesis that shorter tones are more likely to be misclassified can be supported by the notion that the timbre is more affected by a musician's preparation on the subsequent tone. The same applies to influences of the preceding tone, because of reverberation and resonance.

Interpretation of learned models:

Classifiers can be interpreted by humans by visualizing the internal models. Particularly WEKA's decision trees and rule learners can be analyzed by humans clearly. Investigating these models reveal the complexity of the trained classifier as well as the ranking of the used feature attributes the classifier was trained with. By comparing models of tree or rule classifiers with each another the predominant one can be adopted, if there is agreement at all. Strong agreement of a model amongst a major set of classifiers motivate rejection of ones that operate with different models. Such a rejection can only be justified under the assumption that the majority set has higher annotation performance. Another valuable deduction from classifier schemes could be the relationship between complexity of models (in terms of number of nodes and leaves) and annotation performance. This could give rise to limits that stipulates the maximum complexity of a tree or set of rules given a certain classification problem. In case a classifier exceeds such a limit, this might point out overfitting.

7.3.3.1. Case test: performance test with initial training corpus

For an initial impression of the performance of the WEKA classifiers we integrated several tree-classifiers, instance-based classifiers and Neural Network-classifiers with Exprimulator. The classifiers were trained with the initial training corpus from 7.2.1 and trained with the song *Au Claire de la lune*, which was recorded in the same session as the training corpus. The results are presented in this section in order to check if the combination of the MPEG-7 audio descriptors with WEKA classifiers would lead to satisfying results, and adjust the feature vector composition if necessary. If not, performance gain would be achieved by tuning classifiers.

No problems where encountered when constituting the corpus. The majority of the tones for the training corpus where segmented properly. In only a few occasions manual segmentation had to be performed. As for the transcription of *Au Claire de la Lune*, no manual segmentation had to be done, and the pitch detection was flawless.

Classifier name	Description	Classifier category	Performance (correct notes / total notes)
J48	C4.5 decision tree	Trees	70.5%
NBTree	Naive Bayes Tree	Trees	81.8%
LMT	Logistic Model Tree	Trees	**90.9%**
DecisionStump		Trees	61.4%
RandomForest		Trees	**88.6%**
RandomTree		Trees	86.4%
REPTree	Fast decision tree learner	Trees	81.8%
IBk	K-nearest neighbors classifier	Lazy	86.4%
LWL	Locally-weighted learning	Lazy	79.5%
K*	instance-based classifier	Lazy	86.4%
Bayes Network		Bayes	79.5%
Naive Bayes		Bayes	68.2%
Naive Bayes Simple		Bayes	70.5%
MultilayerPerceptron		Functions	86.4%
SMO	Support vector machine	Functions	79.5%
Neural network	Matlab Neural Network	Functions	**88.6%**
JRip	propositional rule learner (Repeated Incremental Pruning to Produce Error Reduction)	Rules	86.4%
NNge	Nearest neighbor	Rules	63.6%
OneR	One rule learner	Rules	63.6%
PART	Rule learner based on JRip and C4.5	Rules	77.3%
Ridor	RIpple-DOwn Rule learner	Rules	75%

Table 7.1: Performances of 21 classifiers trained with the initial training corpus, tested on one song

Concerning the annotation performance of the classifiers, the results from Table 7.1 look promising. More than half the classifiers perform higher than 80%, with a maximum score of 90.9% for the Logistic Model Tree. This is remarkable, as standard settings were used for as well the feature vector calculation as the WEKA classifier parameters. As there are three possible outputs for a classifier, the ones from the table perform significantly better than a random classifier, which would predict 33.3% correctly.

7.3.3.2. Exceptions in calculating feature vectors

When computing the audio descriptors that comprise the feature vectors, exceptions can occur so that no value is acquired for an attribute within the feature vector. For the training of classifiers as well as prediction of test data one should decide whether to skip these exceptional instances or not.

As a concrete example, the calculation of the *LogAttackTime* descriptor is considered, which cannot be calculated in case a tone has a 'weak' attack. In this occasion, the *LogAttackTime* procedure returns the null value. The issue is to decide whether to completely remove such an instance from the set of training instances, or to incorporate it with the specific attribute replaced by a null-value. The same decision has to be made for prediction. A test instance can be labeled as 'unclassifiable' if an exception occurs. Another approach is to substitute the missing feature attribute with for example the mean value of the remaining instances for this particular attribute.

It is trivial that when the training data would incorporate the exceptional instances, this should also be the case with the test instances (hence no 'unclassifiable' labels are assigned when predicting). In like manner, when the training data does not incorporate the exceptional instances, the transcriber would have to label these instances as 'unclassifiable'. For this project the former approach is chosen, as it is undesirable to have unannotated notes in a transcription. This decision is supported moreover by the notion that exceptional instances do not interfere the training of a classifier such that this would disrupt the prediction of 'normal' instances (3.5% of the training data are exceptional instances). Another argument for incorporation is the fact that the exceptions for the *LogAttackTime* descriptors take place more often in the *sul tasto*-class than the other classes, because instances of the *sul tasto*-class have less

strong attacks. Removal of the exceptional instances would mean that there would be less training instances for the *sul tasto*-class in relation to the other classes.

Nevertheless, in case a tone is shorter than the range over which feature calculation is performed, the entire feature vector cannot be computed. In these occasions, labeling as ‘unclassified’ is inevitable within the current implementation of Exprimulator. As stated in section 7.3.3, these unclassified notes are not taken into account for the calculation of the annotation performance.

8. Experiments and results

8.1. Testing preparations

8.1.1. Recording songs

For the experiments presented in this chapter, 10 songs have been recorded containing 67 tones each on average. The length is kept as constant as possible, so that performance rates can be compared. The resulting test corpus contains enough test material (almost 700 tones) for a corpus containing 540 training instances. From well-known lyrical songs only one strophe and/or chorus are recorded. The lyrics are used as a guideline for applying the playing techniques (different techniques are applied sentence wise, see appendix D). The songs are not difficult to play, so that the guitarist can focus on how to play the notes. The songs are selected on their familiarity, length, note reach, note variation and melodic simplicity. It is ensured that only the notes present in the corpus are used. To realize this, the song is transposed to the key in which the maximum and minimum fall between the corpus range. Every song contains the three playing techniques that are applied conforming the lyric markup in appendix D. The following songs have been chosen:

1. *The Beatles – Let it be*
2. *Billy Joel – Uptown girl*
3. *Elvis Presley – Are you lonesome tonight*
4. *Michael row the boat ashore (Afro-American spiritual)*
5. *Davis and Charles Mitchell – You are my sunshine*
6. *Rolling stones – paint it black*
7. *Ella Fitzgerald – Every time we say goodbye*
8. *The Beatles – Michelle*
9. *The Beatles – Yesterday*
10. *Neil Young – Heart of Gold*

Performances have to be sec in other retrospects: no unnecessary vibrato's, pull-ons and pull-offs, etc. It is necessary to explicate this requirement because guitarists are tended to apply these decorations naturally and unconsciously. It is pursued to keep the performance as sec as possible at a relatively low pace, because short notes are not classified because of the fixed feature calculation range. The songs were recorded by the author.

8.1.2. Transcription and annotation of guitar performances

The recorded songs were automatically transcribed using onset and pitch detection. Not all of the note onsets in the songs were recognized, in these cases (about 4%) onsets were added manually. The manual onsets were set at the highest peak visible in the attack section of the tone, similar to the way the automatic song segmentation takes place. Once the onsets were obtained, pitches were calculated of the beginning 5000 samples of the tone. The pitch calculation proved to be flawless over the 10 recorded songs.

According to the song markup of appendix D, the songs' notes were manually annotated (the songs were also played according to this markup). This manual annotation is used by the $des_{RHPP}(tone)$ function (see section 6.1), to be able to compute the annotation performance in a later stage. Feature vectors were calculated using the settings by which the corpus' features were calculated (i.e. same time range of wave signal, same audio descriptors and settings). After this data acquisition phase, the transcriber allows batch annotation of all the songs with all the classifiers trained with the classifier manager. In combination with

a procedure that extracts a performance matrix of all songs and classifiers, performance differences can be discovered quickly when classifier settings are changed.

8.1.3. Creating classifiers

The WEKA classifiers were trained with the corpus presented in section 7.2.2 using the standard WEKA settings. This is because the standard settings are mostly beneficial for the classifier, and enable the capabilities of the classifiers (settings such as subtree raising and pruning are enabled standard). For settings that would have major impact on the structure of the classifier, different configurations were compared to determine the most optimal. This parameter fine-tuning was only performed on the best performing classifiers.

8.2. Results

In this section the results of the metrics introduced in section 7.3.3 are presented. The training corpus is evaluated on its size by measuring the performance of different fractions of the corpus. After that the training corpus is evaluated on consistency by cross-validating all classifiers. The cross-validation process also enables us to eliminate the worst performing classifiers, and can also be compared with the classfier's annotation performance on the song corpus. This gives insight in the generality of the classifiers. Once the best performing classifiers have been determined, the influence of some decisive classifiers parameters on the performance is outlined.

8.2.1. Corpus size

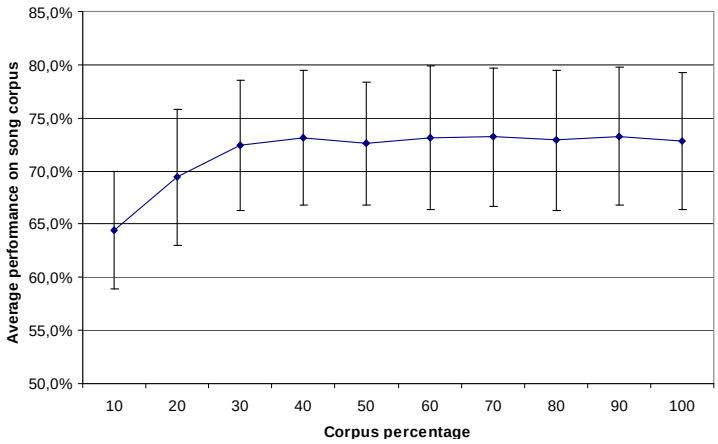


Figure 8.1: Average performance of classifiers on full song corpus trained with different corpus sizes

The average performances of all classifiers over all 10 songs are plotted in Figure 8.1. The different corpus sizes are obtained by eliminating one or more of the 10 repetitions of one tone. Remember that these 10 repetitions were the result of incorporating the interfering dimension *natural playing position deviation* in the corpus, as defined in section 7.2.3. A glance at Figure 8.1 shows that for the average classifier a corpus with only 3 repetitions of one tone for every class suffices, resulting in a corpus size of 162 tones (3 strings x 3 playing techniques x 6 tones per string x 3 repetitions per tone) instead of 540 tones.

However, as can be deduced from Figure 8.2, a 30% corpus is not sufficient for every classifier. In this graph the average performances on the song corpus on five classifiers are displayed. Clearly visible are the different slopes and convergence locations. Only the J48-tree, the Support Vector Machine and the RandomForest classifier reach approximately their maximum performance after a corpus size of 30%. For the K* and k-Nearest Neighbour algorithm this point lies at 70%. Because of the fact that the Bayesian network reaches its peak at 100%, future experiments were executed with the full corpus. The slope of the performance curve of the Bayesian network justifies an evaluation of a bigger corpus in future. The dependency of Bayesian networks on the corpus size is explainable by the notion that classes are predicted on basis of chances which have been deduced from the training instances. The higher the number of instances, the more robust the calculated chance distribution is.

In Figure 8.2 there is a remarkable peak at 90% of the corpus size for the RandomForest classifier. For some reason, this corpus size results in a valley for the J48-tree.

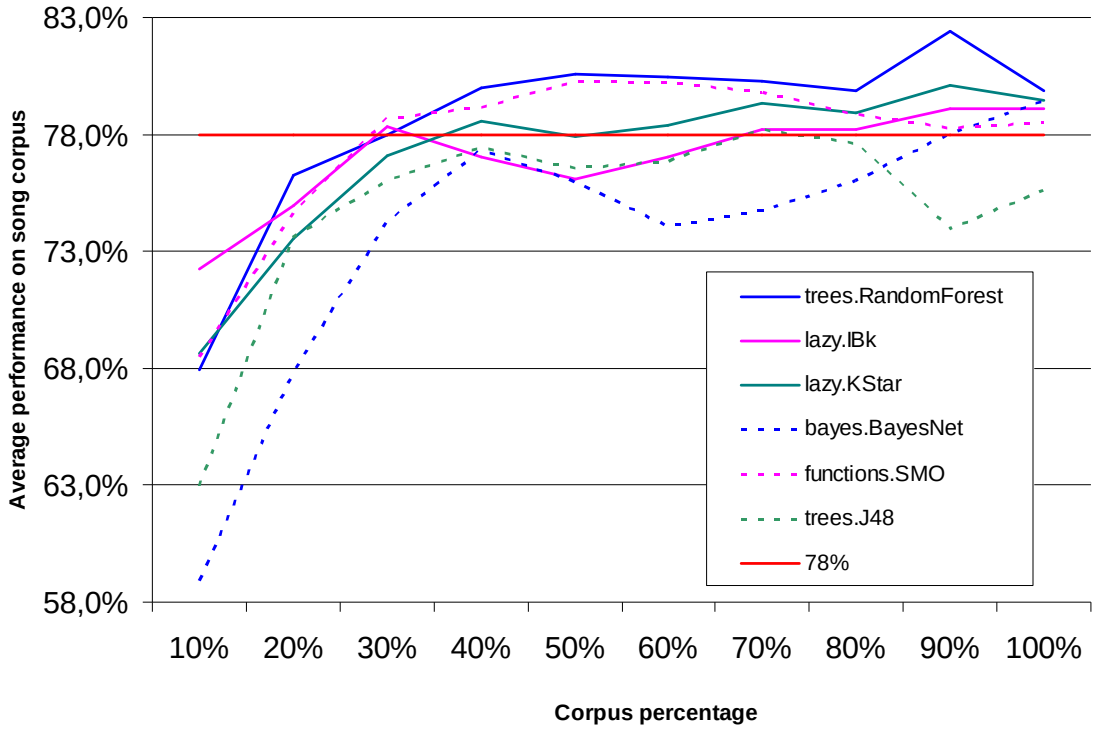


Figure 8.2: Relation between corpus sizes and classifier performance

To evaluate the effect of the corpus size on the complexity of the classifier's model, the number of leaves of the J48-tree was plotted as well as the size of the tree, in Figure 8.3. The curves demonstrate an increasing complexity as the corpus size increases. This motivates the use of small corpora for decision trees, because Figure 8.2 tells that the performance of the J48-tree does not increase after a corpus size of 40%.

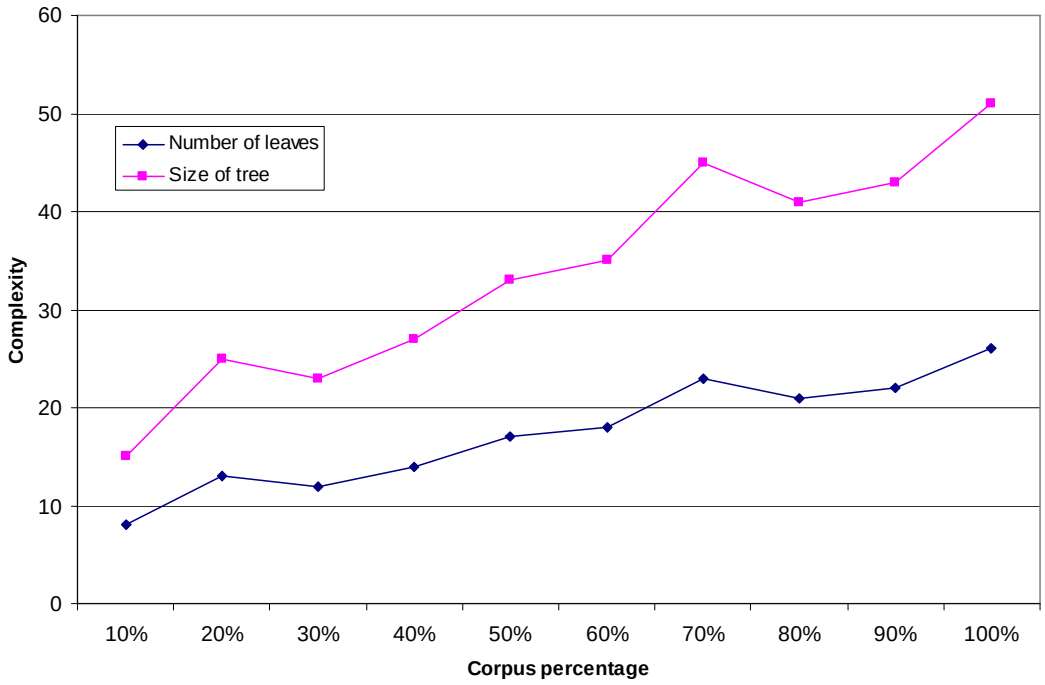


Figure 8.3: Relation between corpus size and tree complexity

In an attempt to optimize the training corpus, instances that deviated significantly with respect to some feature attribute value (*HarmonicSpectralSpread* and *HarmonicSpectralVariation*) were removed. These instances appeared isolated in the scatter plots of Figure 8.10. Besides that it was evaluated by ear if the accompanying tones indeed did not belong in its current class. In these cases it appeared that the tones sounded differently because of the string that was not struck properly or the fret that was not pressed properly. These technical imperfections resulted in a noisy sound. After removal of five of these instances (less than 1% of the corpus), the majority of the classifiers had higher performance on the song corpus. It appeared that the tree and rule learners were most sensitive to these small corpus changes. The REP-tree performance for example increased 9.5% for this small corpus change. This high increase might indicate that the REP-tree was overfitted. Whether the trees and rules are apparently sensitive to noise in the training data, the instance-based learners (K*, Locally Weighted Learning and the 1-Nearest Neighbour algorithm) were not affected by the corpus change. This is because these classifiers use the proximity of an instance to other instances as a measure for prediction. Hence it is likely that the five excluded instances were never taken into account for prediction of the song corpus.

8.2.2. Cross validating classifiers

In Figure 8.4 the performances of all the classifiers were evaluated by performing 10-fold cross validation on the training corpus, with a split of 90% test and 10% training. The worst performing classifiers are the simple classifiers that only consider one feature attribute (DecisionStump), or one rule (OneRule). A closer look at the confusion matrix of DecisionStump learns that not a single tone is classified as *sul ponticello*, hence a performance worse than 66.67%. The DecisionStump should be a multi-class classifier (capable of classifying more than 2 classes), but apparently because of its simplicity (only one feature attribute is considered in the underlying tree), the performance is not adequate. Therefore, the DecisionStump classifier is not used in future experiments. Because Locally Weighted Learning standard has DecisionStump as a base classifier, it performs equally low (apparently LWL cannot boost the DesicionStump's performance). This is why for this cross-validation experiment and the following experiments the standard base classifier of LWL is replaced by the Support Vector Machine. The OneRule and DecisionStump classifiers are excluded from further experiments, as it appears they cannot even classify corpus material adequately. The rest of the classifiers exhibit slight deviations relating to each other that are not decisive to exclude more classifiers.

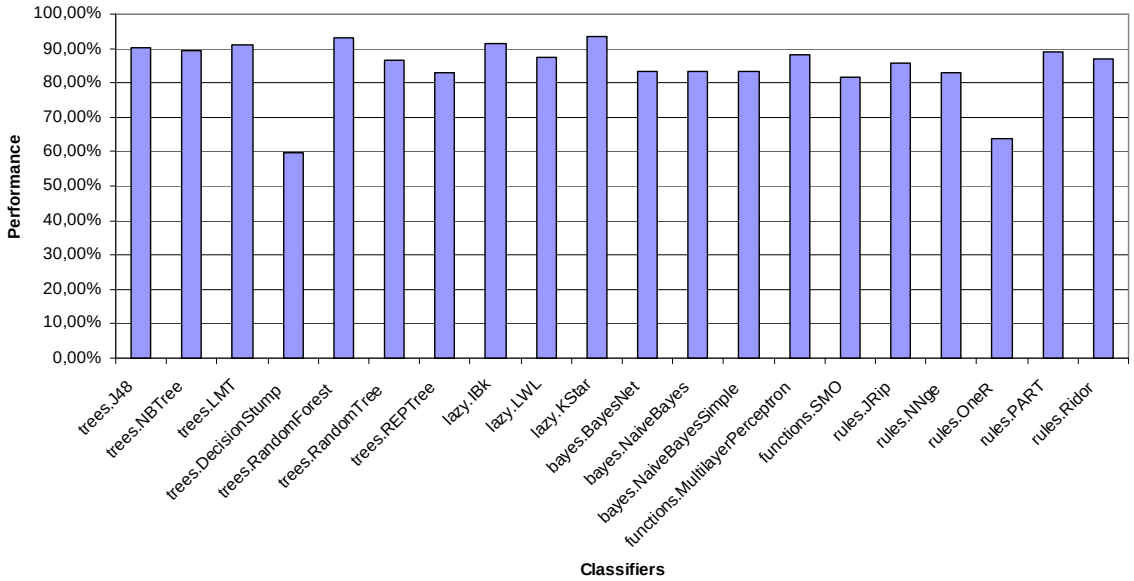


Figure 8.4: Average performance of 10-fold cross validated classifiers

8.2.3. Annotation performance

In this section the results of performances of the classifiers on the song corpus are evaluated. Automatic annotation of all the songs by the selected classifiers from the previous section resulted in Table 8.1, in which the annotation performance of every classifier for each song is displayed.

The average performance over the entire song corpus for every classifier is listed in the last column, in which classifiers performing better than 78% are formatted bold. In like manner the average song annotation performance is displayed in the last row. These scores deviate more than the average performances of the classifiers. The songs *Yesterday* and *Heart of Gold* have the lowest annotation score. It is hard to ascertain what properties of these songs contribute to these low scores, but it is likely due to higher use of error prone playing techniques such as *neutral* or *sul tasto*.

	The Beatles – Let it be	Billy Joel – Uptown girl	Elvis Presley – Are you lonesome tonight	Michael row the boat ashore – (Afro-American spiritual)	Davis and Charles Mitchell – You are my sunshine	Rolling stones – paint it black	Ella Fitzgerald – Every time we say goodbye	The Beatles – Michelle	The Beatles – Yesterday	Neil Young – Heart of Gold	Mean song performance
J48	76,8%	76,9%	79,5%	80,4%	71,2%	81,2%	**91,8%**	71,4%	60,7%	66,1%	75,6%
NBTree	69,6%	76,9%	85,2%	76,1%	80,8%	82,2%	83,7%	71,4%	55,4%	71,2%	75,3%
LMT	82,1%	82,1%	78,7%	80,4%	71,2%	79,2%	87,8%	72,9%	75,0%	69,5%	77,9%
RandomForest	75,0%	**93,6%**	84,1%	73,9%	75,3%	85,1%	87,8%	74,3%	75,0%	74,6%	**79,9%**
RandomTree	55,4%	62,8%	71,6%	80,4%	67,1%	80,2%	71,4%	67,1%	69,6%	66,1%	69,2%
REPTree	82,1%	85,9%	71,6%	60,9%	68,5%	79,2%	83,7%	71,4%	64,3%	62,7%	73,0%
IBk	71,4%	80,8%	80,7%	84,8%	86,3%	81,2%	77,6%	78,6%	75,0%	74,6%	**79,1%**
LWL	78,6%	84,6%	80,7%	73,9%	86,3%	86,1%	83,7%	81,4%	85,7%	69,5%	**81,1%**
KStar	73,2%	85,9%	84,1%	82,6%	**93,2%**	81,2%	71,4%	84,3%	64,3%	74,6%	**79,5%**
BayesNet	75,0%	87,2%	78,4%	**93,5%**	86,3%	81,2%	87,8%	72,9%	73,2%	59,3%	**79,5%**
NaiveBayes	69,6%	67,9%	72,7%	76,1%	65,8%	83,2%	71,4%	62,9%	60,7%	52,5%	68,3%
NaiveBayesSimple	73,2%	69,2%	73,9%	76,1%	64,4%	83,2%	71,4%	63,8%	60,7%	52,5%	68,8%
MultilayerPerceptron	80,4%	80,8%	81,8%	78,3%	71,2%	82,2%	71,4%	67,1%	71,4%	69,5%	75,4%
SMO	76,8%	80,8%	78,7%	76,1%	82,2%	84,2%	79,6%	80,0%	83,9%	62,7%	**78,5%**
JRip	80,4%	82,1%	76,1%	69,6%	71,2%	82,2%	71,4%	80,0%	67,9%	61,0%	74,2%
NNge	66,1%	75,6%	74,2%	71,7%	67,1%	82,2%	79,6%	65,7%	50,0%	52,5%	68,5%
PART	80,4%	80,8%	83,0%	78,3%	72,6%	80,2%	81,6%	74,3%	64,3%	71,2%	76,7%
Ridor	83,9%	82,1%	69,3%	76,1%	69,9%	80,2%	81,6%	72,9%	67,9%	71,2%	75,5%
Mean classifier performance	75,0%	**79,8%**	**78,0%**	**77,2%**	75,0%	**81,9%**	**79,7%**	72,9%	68,1%	65,6%	75,3%

Table 8.1: Annotation performances of classifiers on songs

In Figure 8.5, the standard deviation from Figure 8.4 is illustrated in relation to the annotation performance of the classifiers on the song corpus, together with the standard deviation. It is worthwhile noticing that the ratios between the classifiers' annotation performances are quite similar to the ratios between the cross validation performances. This indicates that the cross validation metric is suitable for comparing classifiers with each other for a certain classification task, and is representative for the distribution of the annotation performances on test material.

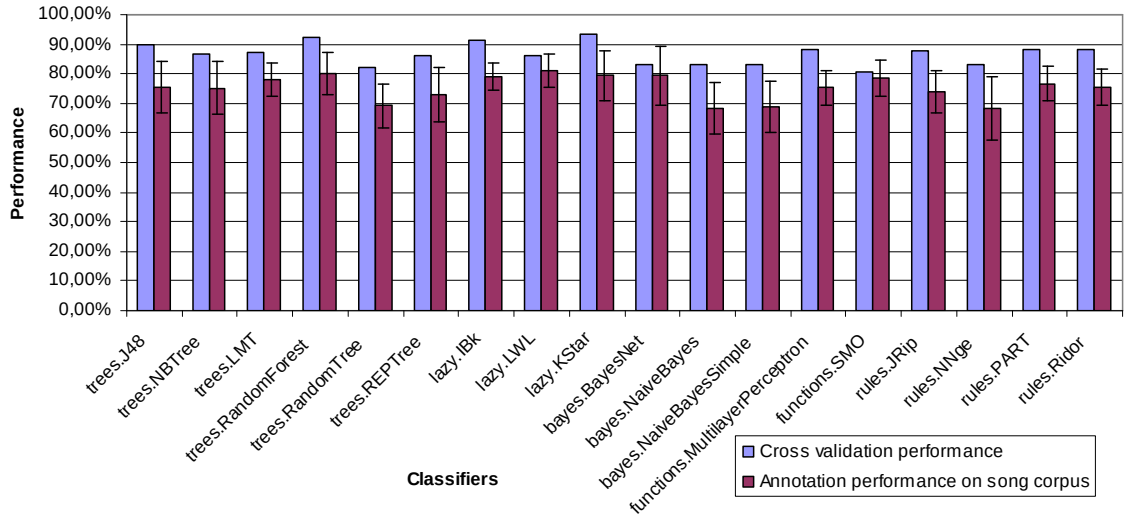


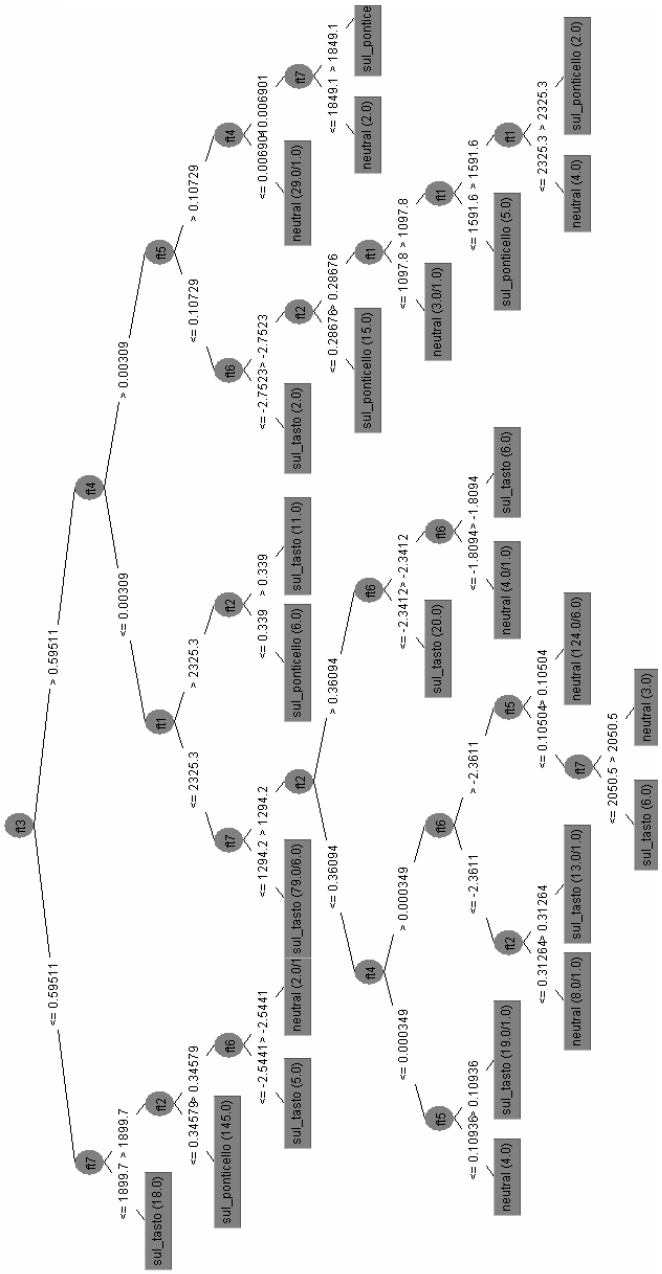
Figure 8.5: Cross validation performance of Figure 8.4 together with annotation performance on song corpus with standard deviation

From Figure 8.5 the consistency and generality of classifiers can be derived. The 1-Nearest Neighbour classifier (IBk) is regarded as the most consistent performing classifier, because of its lowest standard deviation of 0.046%. The cross-validation performance can be compared with the annotation performance to obtain a measure for the generality of the classifier. Therefore the cross-validation / annotation performance ratio is computed. This gives a measure for the decrease of performance on test data. Significantly better at generalizing were the Support Vector Machine (performance decrease of 4.11%) and the Bayesian Network (4.85%). The worst generalizing classifier is the RandomTree. This is supported by the observation that the RandomTree was also sensitive for instance removal as was illustrated in section 8.2.1.

Striking is that the Simple Naive Bayes and Naive Bayes classifier significantly decrease in performance in comparison with the Bayesian Network. The cross validation performances are roughly the same.

8.2.4. Interpretation of learned models

Figure 8.6: J48 tree



In this section, a selection of trained rule and tree classifiers are visualized, compared and analyzed. By examining in which order the feature attributes occur in the nodes of the trees and how many instances are divided under the node, a feature ranking can be obtained. In this section, the graphs of 4 trees, the J48-tree, NBTree (Naive Bayes tree), LMT-tree (Logistic Model Tree) and the REP-tree are visualized. In these trees, the feature attributes are located in the nodes, depicted by the abbreviations ft1 till ft7:

ft1	*HarmonicSpectralCentroid*
ft2	*HarmonicSpectralDeviation*
ft3	*HarmonicSpectralSpread*
ft4	*HarmonicSpectralVariation*
ft5	*TemporalCentroid*
ft6	*LogAttackTime*
ft7	*SpectralCentroid*

The branches of the tree represent the Boolean decisions that are made on basis of the value of the node's feature attribute. In Table 8.2 the attribute ranking is given that was inferred from the trees.

The **J48-tree** (see Figure 8.6) appeared to be the most complex tree. Its performance was not worse than the other trees however. In the **NBTree** (Figure 8.7) and the **REP-tree** (Figure 8.9) some feature attributes are absent. Remarkable is that the *SpectralCentroid* misses in the NBTree while this attribute appears in the fourth position in the J48-tree. What's more, the *HarmonicSpectralCentroid* (missing in the REP-tree) is the third important attribute according to the J48 and LMT-tree. The two best performing trees, the J48-tree and the **LMT-tree** (Figure 8.8) seem to have the most similar feature ranking (5 corresponding attribute rankings), whereas amongst the other trees there seems to be little agreement. This can implicate that the descriptive strength of the attributes are approximately equally powerful. All trees have *HarmonicSpectralSpread* as the most descriptive attribute, which is also subscribed by the ranking methods of Table 8.3. These methods incorporate the classifier JRipper, from which the order of the rules and its comprising attributes, as well as the number of affected instances was determinative for the feature attribute ranking. The feature ranking for the Bayesian Network was obtained by calculating the annotation performance 7 times, each time omitting a different attribute. The descending order of annotation performances for each iteration determined the attribute ranking this time. The third method was performing attribute selection with the WEKA toolkit. This time the importance of a feature attribute was estimated by measuring the information gain with respect to the class. The last method presented in

Table 8.3 presents the feature attribute-class correlation. To calculate this correlation the nominal classes were converted into the numerical equivalents -1, 0, and 1 for respectively *sul ponticello*, *neutral* and *sul tasto*. This was done under the assumption that there would be a linear correlation between the RHPP and the feature attributes. Just like with the trees, the miscellaneous attribute ranking methods did not agree on a consistent ranking method.

ranking	J48-tree	NBTree	LMT-tree
1	*HarmonicSpectralSpread*	*HarmonicSpectralSpread*	*HarmonicSpectralSpread*
2	*HarmonicSpectralVariation*	*HarmonicSpectralCentroid*	*HarmonicSpectralVariation*
3	*HarmonicSpectralCentroid*	*TemporalCentroid*	*HarmonicSpectralCentroid*
4	*SpectralCentroid*	*HarmonicSpectralVariation*	*SpectralCentroid*
5	*HarmonicSpectralDeviation*	*LogAttackTime*	*HarmonicSpectralDeviation*
6	*TemporalCentroid*	*HarmonicSpectralDeviation*	*LogAttackTime*
7	*LogAttackTime*		*TemporalCentroid*

ranking	REP-tree
1	*HarmonicSpectralSpread*
2	*HarmonicSpectralVariation*
3	*SpectralCentroid*
4	*HarmonicSpectralDeviation*
5	*TemporalCentroid*
6	
7	

Table 8.2: Ranking of feature attributes by tree classifiers

ranking	JRipper	Bayesian Network	Attribute selection by info gain ranking
1	*HarmonicSpectralSpread*	*HarmonicSpectralSpread*	*HarmonicSpectralSpread*
2	*HarmonicSpectralDeviation*	*HarmonicSpectralDeviation*	*LogAttackTime*
3	*SpectralCentroid*	*LogAttackTime*	*SpectralCentroid*
4	*HarmonicSpectralVariation*	*HarmonicSpectralVariation*	*TemporalCentroid*
5	*LogAttackTime*	*TemporalCentroid*	*HarmonicSpectralVariation*
6	*HarmonicSpectralCentroid*	*SpectralCentroid*	*HarmonicSpectralCentroid*
7		*HarmonicSpectralCentroid*	*HarmonicSpectralDeviation*

ranking	Ranking on attribute – class correlation
1	*HarmonicSpectralSpread*
2	*HarmonicSpectralDeviation*
3	*SpectralCentroid*
4	*TemporalCentroid*
5	*HarmonicSpectralCentroid*
6	*HarmonicSpectralVariation*
7	*LogAttackTime*

Table 8.3: Feature attribute ranking with miscellaneous methods

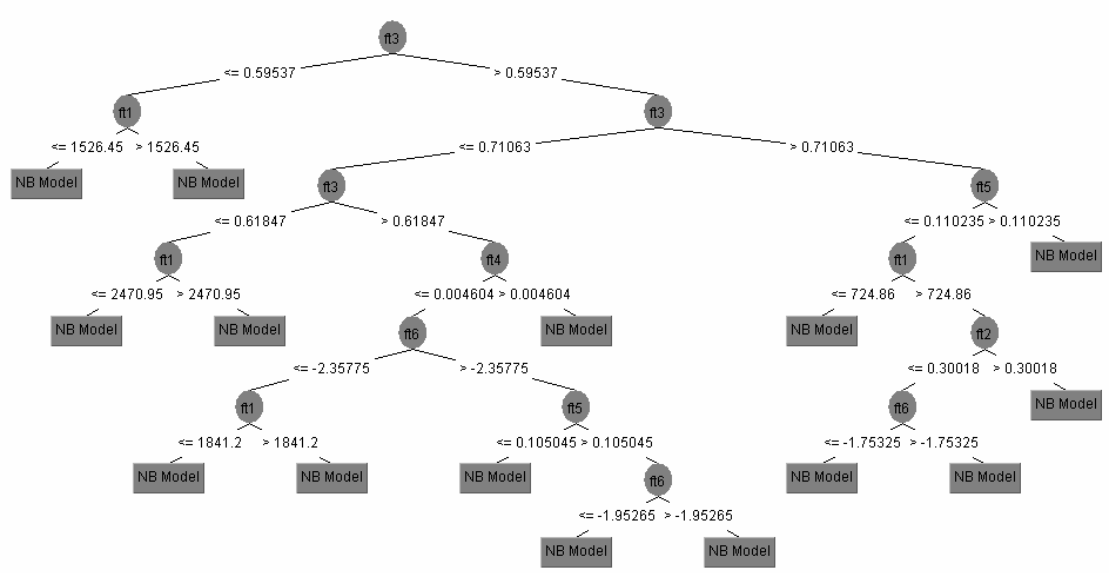


Figure 8.7: Naive bayes tree

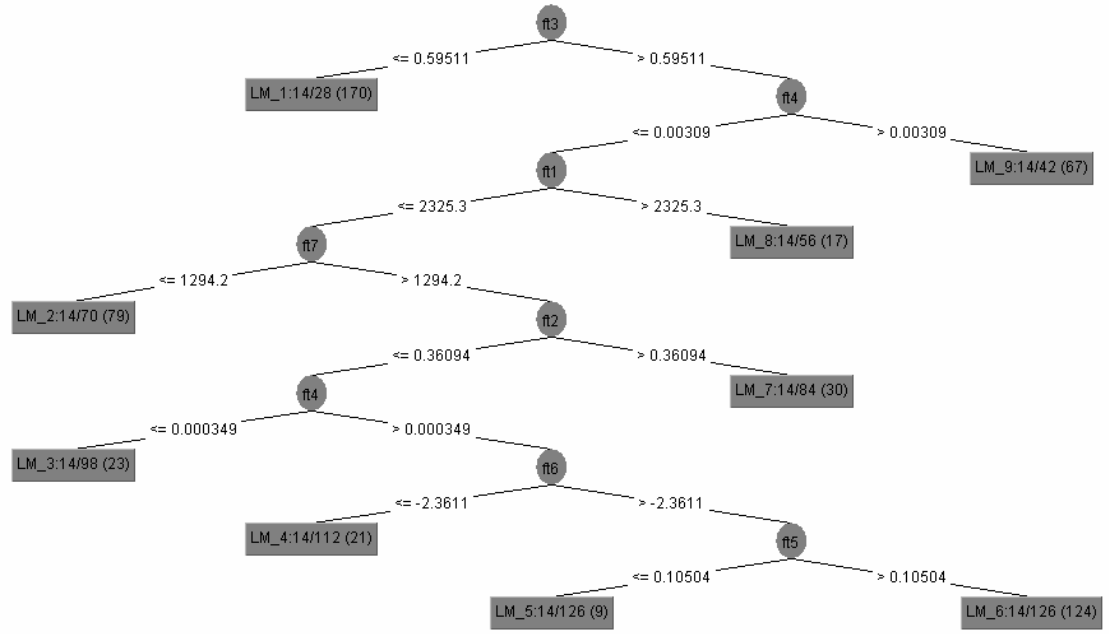


Figure 8.8: Logistic model tree

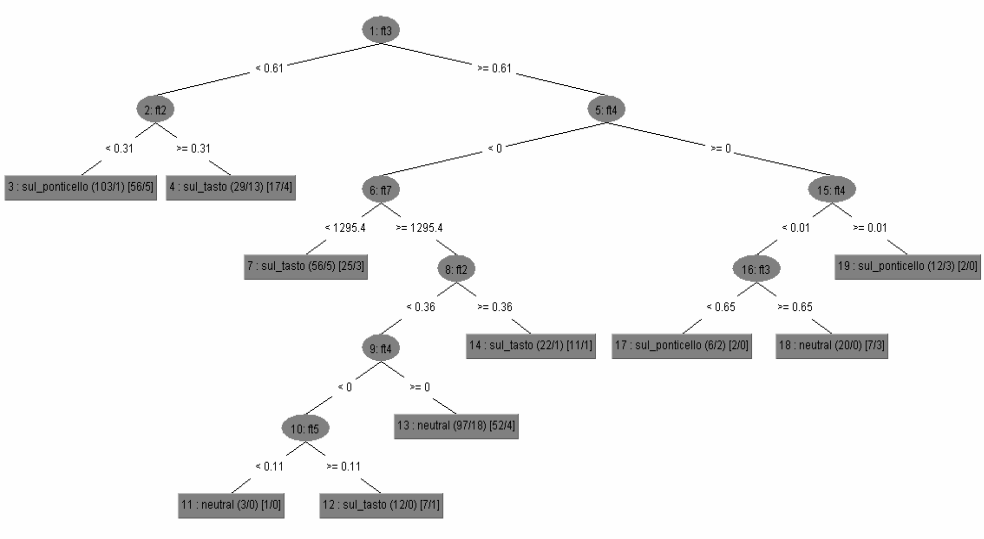
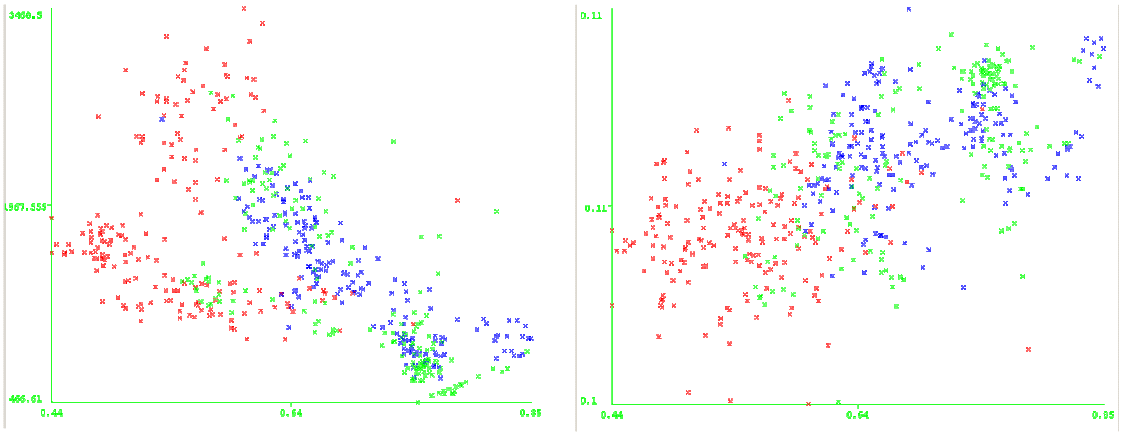


Figure 8.9: REP-tree

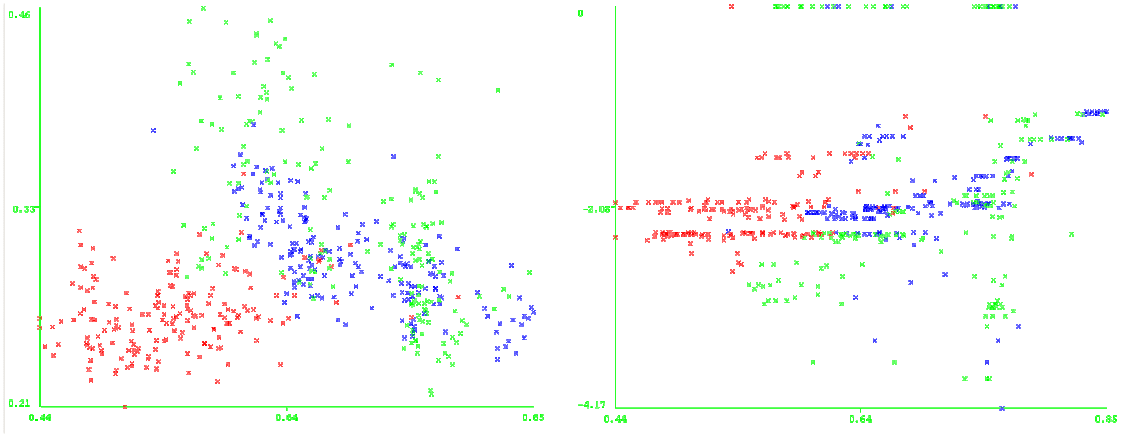
In Figure 8.10 six scatter plots are drawn that relate the most discriminating feature (HarmonicSpectralSpread according to the analyzed decision trees) against the remaining six feature attributes. Clearly visible is the separation of the sul ponticello instances, which are separated most distinctively from the instances of the other classes. This corresponds with results of the confusion matrix, in which sull ponticello is misclassified least often. The Sul tasto and neutral swarms are overlapping in all the scatter plots, but it can be judged by eye that *HamonicSpectralSpread* gives best distinction with *HarmonicSpectralDeviation* and *HarmonicSpectralVariation*. This is in accordance with 5 of the analyzed classifiers.



■ Sul ponticello
■ Sul tasto
■ Neutral

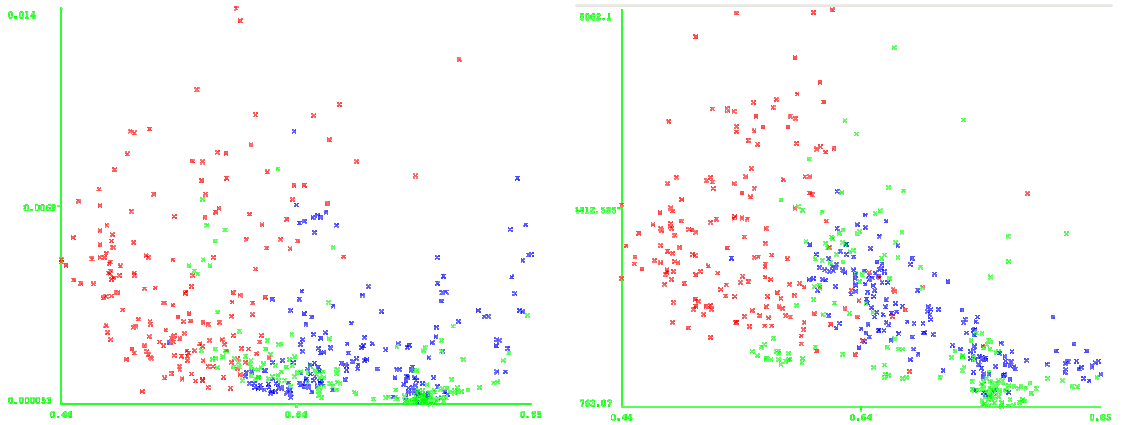
(a) x: HarmonicSpectralSpread →y: HarmonicSpectralCentroid

(b) x: HarmonicSpectralSpread →y: TemporalCentroid



(c) x: HarmonicSpectralSpread →y: HarmonicSpectralDeviation

(d) x: HarmonicSpectralSpread →y: LogAttackTime



(e) x: HarmonicSpectralSpread →y: HarmonicSpectralVariation

(f) x: HarmonicSpectralSpread →y: SpectralCentroid

Figure 8.10: Feature attributes plotted against most discriminating feature (HarmonicSpectralSpread)

8.2.5. Relation between note characteristics and misclassification

A confusion matrix is calculated to ascertain which playing technique is misclassified most often. The confusion matrix from Table 8.4 is the mean confusion matrix over the 20 classifiers. The confusion matrix was calculated by performing 10-fold cross-validation over the training set. The bold numbers above the heading labels denote how often a tone was incorrectly misclassified as the column's class, whereas the bold numbers most right tell how often the row's class was misclassified. The matrix tells us that *sul ponticello* is the least misclassified playing technique (12 misclassifications on average). *Sul ponticello* is also the least often incorrectly used as label. The playing techniques *neutral* and *sul tasto* are approximately equally often misclassified, whereas neutral is most often used as an incorrect label. This is easily explained by the notion that neutral is an intermediate class. The fact that *sul tasto* and neutral are more often mixed up, means that the distance of the feature vectors of the mutual classes is not linearly correlated with the physical distance on the guitar string between the classes.

	36	**11**	**29**		
Classified as→	neutral	sul ponticello	sul tasto		
	149	5	26	neutral	**31**
	9	168	3	sul ponticello	**12**
	27	6	147	sul tasto	**33**
				actual class ↑	

Table 8.4: Average confusion matrix of 20 classifiers

With a different approach, the relation between pitch and misclassification can be investigated. For every pitch is counted how often it occurs in the ten songs of the song corpus (Figure 8.11b), and how often a tone is calculated wrong more often than 10 times amongst the 18 classifiers (see Figure 8.11a). Division of the number of misclassifications by the total number of notes per pitch yields Figure 8.11c. This figure shows that midi note 53 has the highest chance of misclassification (F3), followed by midi note 45 (A2). Midi note 47 (B), 49 (C#3) and 56 (G#3) have higher chance of correct classification, but also occur less often in the songs absolutely.

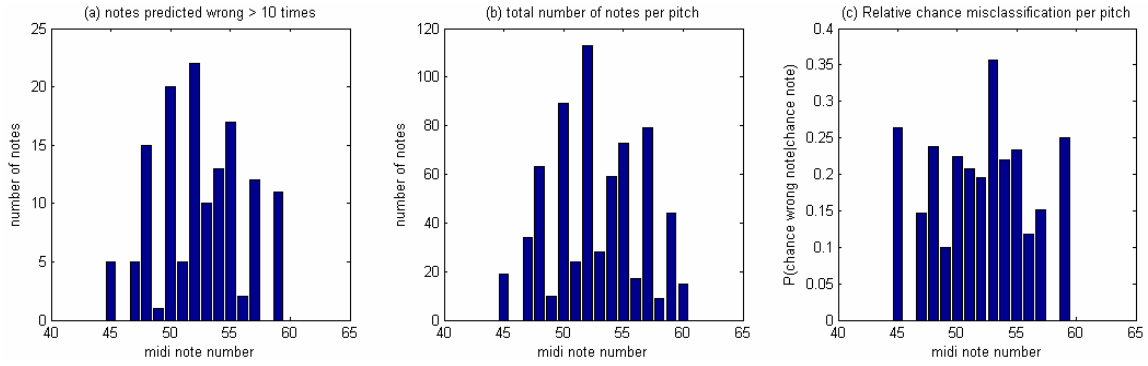


Figure 8.11: Number of misclassifications of pitches

In order to find relations between note duration and misclassification, the notes are grouped in classes of 0.3s wide, resulting in the distributions of Figure 8.12. It appears that a note duration d of 2.4s$\leq d <$2.7s is most likely misclassified. In general, longer notes are more likely to be misclassified than short notes, which is surprising as it was suspected that shorter notes would have higher chance on misclassification. This may be due because of the fact that the training corpus contained tones that are consistently 1s long. From the histogram can be derived that $d<$1s. has a relative low chance on misclassification.

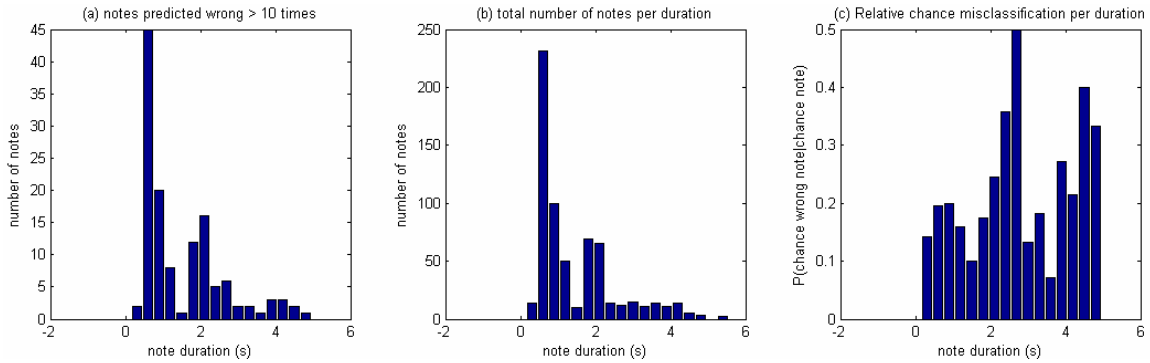


Figure 8.12: Number of misclassifications per note duration

8.2.6. Fine-tuning classifiers

By default, the Locally Weighted Learner is trained with a DecisionStump tree as base classifier, resulting in an unsatisfying performance as was stated in section 8.2.2. Therefore LWL was trained with different base classifiers that had on its own a performance above average (Support Vector Machine, *k*-Nearest Neighbour decision tree, RandomForest and the Bayesian network). Unfortunately, a number of classifiers could not be incorporated as base classifier in the LWL, because they were not weighted instance handlers, such as the K* rule learner and the Logistic Model Tree. The performance results of these different classifier configurations that were allowed are denoted in Table 8.5. As expected, the performance increased significantly and also outperformed the used base classifiers on its own. The SVM appeared to be the most suitable match for the LWL. The LWL - RandomForest tree combination second best, but was very slow at prediction. This is explained by the fact that it is a 3 level meta-classifier as the RandomForest on its turn is comprised of multiple DecisionStumps. Through the course of this chapter, the SVM was used as a base classifier for the LWL classifier.

Classifier (default parameter → altered parameter)	**Initial performance (default settings)**	**Eventual performance (altered settings)**	**Performance gain**
IBk (k = 1 → k = 2)	78,5%	77,3%	-1,6%
LMT (numBoostingIterations = 1 → 10)	77,9%	77,5%	-0,6%
RandomForest (numTrees = 10 → 17)	79,9%	81,7%	2,3%
REPTree (numFolds=3 → 5)	73,0%	73,6%	0,8%
REPTree (numFolds=3 → 7)	73,0%	73.7%	0,9%
REPTree (numFolds=3 → 9)	73,0%	75,6%	3,6%
REPTree (pruning=False → True)	73,0%	71,8%	-1,7%
J48 (pruning=False → True)	75,6%	76,3%	1,0%
PART (pruning=False → True)	76,7%	76,3%	-0,4%
JRip (pruning=False → True)	74,2%	78,1%	5,3%
LMT (LogitBoost=-1 → 5)	77,9%	77,4%	-0,6%
Functions.SMO (useRBF=False → True)	78,5%	64,9%	-17,3%
Functions.SMO (buildLogisticModels=False → True)	78,5%	79,2%	0,9%
LWL (baseClassifier=DecisionStump → SMO)	61,1%	81,1%	32,7%
LWL (baseClassifier=DecisionStump → IBk)	61,1%	78,1%	27,9%
LWL (baseClassifier=DecisionStump → RandomForest)	61,1%	80,4%	31,6%
LWL (baseClassifier=DecisionStump → BayesNet)	61,1%	79,1%	29,6%

Table 8.5: Performance gain of different classifier settings related to default settings

As one may suspect, pruning should be beneficial for the classifier performance on test sets, as branches which are the result of overfitting are removed. However, in the case of the J48-tree and the JRIPPER rule learner, disabling pruning resulted in an increased performance of respectively 1% (which is not considered a significant improvement), and 5.3% on average. Pruning reduced the number of rules for the

JRipper rule learner with 50%, and for the J48-tree with 13%. Not all the songs were consistently annotated better with the two unpruned classifiers, and the overall improvement is not significant. This is why pruning was enabled by default for the rest of the experiments in this thesis. Moreover, if trees and rules are to be compared, the settings should be as alike as possible.

For the RandomForest classifier the optimal number of trees was determined by increasing this parameter until expanding did not result in better performance. As Figure 8.13 shows, RandomForest performs optimal when built with a decision forest consisting of 17 DecisionStump trees, with a performance of 81,7%. This is a performance gain of 2.3% with respect to the default 10 trees a RandomForest consists of. The previously attempted incorporation of the default RandomForest (with 10 subtrees) within the LWL classifier, yielded a performance gain of 0,6%. Unfortunately, incorporation of a RandomForest with 17 subtrees within the LWL-classifier, resulted in performance loss (from 81,7% to 81,1%), whereas an increase would be expected.

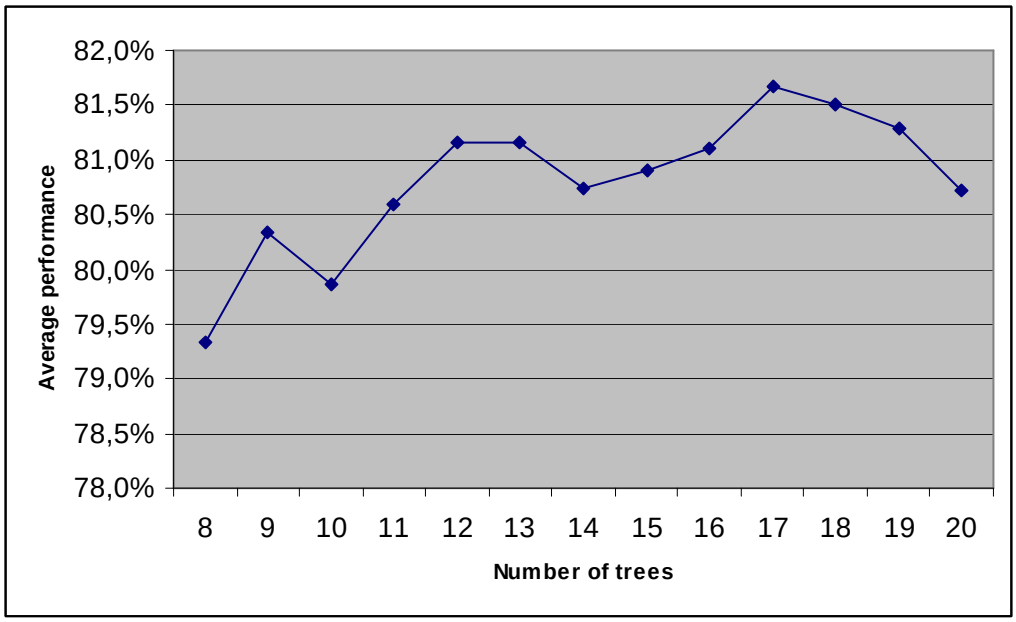


Figure 8.13: Performance of RandomForest classifier with different amount of subtrees

9. Conclusions

The main goal of this thesis was to extract transcriptions with expressive annotation from human musical performances. As a proof of concept, a concrete expressive dimension – Right Hand Playing Position (RHPP) – was chosen to be recognized by several machine learning techniques. The developed classifiers were capable of classifying this expressive dimension in three classes: *sul tasto*, *neutral* and *sul ponticello*. This chapter summarizes the performance of these classifiers, and discusses whether this performance suffices for the proposed applications in the introductory chapter. To ensure a higher potential of success for future research, two questions are asked:

- Do the performance results motivate future recognition of other expression modes?
- How suitable is the Exprimulator environment for general sound classification and musical transcription?

9.1. Performance evaluation

The performance related experiments of the previous chapter were carried out to make choices about selection and parameterization of corpora, classifiers and feature vectors. These choices would raise the chance of success and reduce the execution time of future expressive sound classification experiments. In this section it is investigated if these choices can be made on basis of the experimental results from chapter 8.

To start with *classifier selection*, performance evaluation on a test corpus of 10 songs did not result in a classifier that consistently performed better. Locally Weighted Learning with a Support Vector Machine as a base classifier performed the best on average (81.1%) on the song corpus, but considering the standard deviation it does not outperform other classifiers convincingly. However, to obtain a manageable set of classifiers for future experiments, an exclusion rule like proposed in 8.2.3 can be used. The rule in question identifies a set of classifiers that performs significantly worse than the best performing classifier. An excluded classifier has the property that its annotation performance is worse than that of the best classifier minus the standard deviation of the best classifier's annotation performance. The following classifiers can be excluded by applying this rule: RandomTree, REP tree, Naive Bayes, Naive Bayes Simple and Nearest Neighbour. The remaining set of 13 classifiers is not a small set of best performing classifiers as one would desire. However, the reality is that the annotation performances of these classifiers lie too close to one another to apply more drastic exclusion.

Fine-tuning parameters of the classifiers did result in improved performance, with a maximum of 5.3% for the JRip rule learner. However, this improvement did not lead to a position amongst the top classifiers (the five best performing classifiers). For RandomForest, a classifier in the top segment, increasing the number of random trees did result in an increase of 2.3%. The fact that none of the classifiers performed much better than 80%, and that there is not much performance deviation amongst the best performing classifiers, leads to the conclusion that the current feature vector configuration does not allow a better performance. Cross validation on the training corpus appeared to provide a good indication for the adequacy of classifiers on testing material, as they followed roughly the same trends as the performance distribution on the song corpus. The positive correlation between the cross validation performance and the test performance also means that the training material of the corpus appears to be representative for the tones that occur in melodic guitar performances.

From the fractions Φ of the total corpus ϕ_{100} defined in section 7.2.3, it appeared that ϕ_{40} sufficed for the most of the classifiers. Only the Bayesian network could have benefited from a bigger corpus than ϕ_{100}, as the curve still showed a rising trend at ϕ_{100}. The fact that most classifiers reached their performance maximum at ϕ_{40}, proved that deliberate addition of positional jitter did contribute only to a certain extent to the performance. This contrasts to the incorporation of the full chromatic scale of pitches in the corpus, because the removal of only one pitch already results in performance loss.

A corpus that is as small as possible is desirable for the current design of Exprimulator, because the creation of a corpus for every expressive dimension is time consuming and expensive in terms of manual creation time, computation time and disk space. In the case of tree and rule learners smaller corpora appear to result in simpler models in terms of number of leaves and tree size. Simpler classifiers that annotate with a comparable performance to a more complex model are preferred because they are more easily interpreted, perform faster and are less likely suffer from overfitting.

Because of the large degree of freedom of adjustable parameters for the *calculation of feature vectors* and the creation of classifiers, a fixed feature calculation algorithm was chosen. Therefore, standardized audio descriptors were used that already have been proven useful in literature on other audio classification tasks. The 7 timbre related MPEG-7 audio descriptors sufficed for RHPP recognition. For more subtle recognition tasks, additional audio descriptors can be added. It is worth noticing that the classifiers did not agree on the order of importance of the audio features. The classifiers only agree on the usage of *HarmonicSpectralSpread* as the most important feature. These disagreements could implicate that the remaining features did contribute approximately equally to the classification process. Together with the notion that the maximum classifier performance lies around 80% this motivates further exploration of suitable audio descriptors.

Despite the fact that Exprimulator is capable of classifying sound classes of any type, it is wise to boost the recognition accuracy of the RHPP first before starting with recognizing other expressive dimensions. This is because it is believed that a higher annotation performance than 80% must be possible for recognizing the playing position, as the differences between the classes under consideration is clearly hearable for a human listener. Therefore, the annotation of a relatively clear expressive dimension must be improved first, before paying attention to more subtle differences such as the finger configuration (*appoyando – tirando*). Because fine-tuning the classifiers did not result in significantly higher performances, it must be explored in what degree the expansion of available calculable audio descriptors contributes to a better annotation performance.

9.2. Evaluation of Exprimulator

Exprimulator met the requirements presented in chapter 5, as can be concluded from section 7.1. Besides meeting primary requirements, it is important to determine how well Exprimulator aided the process of achieving the goals presented in section 2.1 and carrying out experiments. Exprimulator needs to facilitate an *efficient, flexible, adjustable, transparent* and *general* course of experiments. The efficiency is not concerned with optimizing the computation speed of processes, but with the capability to automate repetitive task that otherwise would have to be done manually. The flexibility is concerned about how easily the experimental setting can be varied by means of the GUI. For the adjustability of Exprimulator, it is evaluated how easily new features were implemented in code. Exprimulator must enable insight in calculated data such as feature vectors, transcriptions, etc. The generality is concerned with the usability of Exprimulator for other sound classification problems and other annotation tasks. This latter assessment is important as this thesis focused solely on the specific case of recognizing the RHPP.

Efficiency:

The following features of Exprimulator saved a lot of time during the execution of experiments:

- Dividing performances and compound corpus files into separate tones. In the exceptional case a tone onset was not found, manual segmentation could be realized fast by clicking on the visual representation of the waveform.
- Calculate feature vectors of a large amount of tones. Feature vectors of a specified range of all the tones in the training corpus can be calculated.
- Training and initializing multiple classifiers. Classifiers of different types can be initialized and trained on the same training corpus in one run.
- Transcription and annotation of multiple songs with multiple classifiers can also be realized in one run.

Flexibility:

- Training corpora can be easily modified by excluding tones that are not played well. Another purpose, for which exclusion was used, was the creation of different corpus sizes. The effects of the different corpora were evaluated quickly because of the ability to initialize and train all the classifiers at once, and annotate the entire song corpus with the classifiers in one run.
- The classifiers presented in this thesis were also fully configurable according to the available features offered by WEKA via the Java API.

Adjustability:

- Because the standardized MPEG-7 descriptors were considered more descriptive and elegant than earlier custom implemented descriptors based on formula from literature, the latter were replaced by the MPEG-7 descriptors. These programmatic code adjustments were easily made, because of the structured and modular design of Exprimulator.

  The incorporation of WEKA classifiers was also easily implemented within the current framework of Exprimulator. Matlab fulfilled a catalyzing role in achieving this. Firstly, the Java API of WEKA could be invoked in its entirety in Matlab and Java commands could be inserted within Matlab scripts. Secondly, fast iterative development was made possible by Matlab's scripting engine, which enabled runtime additions and revisions of code sections.

Transparancy:

- Exprimulator provided visual insight in the entire process of calculating feature vectors, constructing classifiers and annotating musical performances. The feature vectors could be visualized by plotting them per class grouped by color, as well as making scatter plots of all the instances in which two feature attributes are plotted against each other (Figure 8.10). The underlying models of all the WEKA classifiers could be examined by drawing them (in case of some of the trees) or by outputting textual information about the structure of the classifier. The transcriptions made by note onset detection, pitch detection and expressive annotation were visualized in a score similar to a MIDI sequence (Figure 7.3).

Generality:

- The classifier manager module of Exprimulator does not place any restrictions on the type of content an audio signal carries, so in principle any sound class configuration imaginable can be constructed with the classifier manager. The transcriber module is more dedicated to the transcription of musical performances, as it outputs a note sequence, but speech signals share the property that they are dividable into fragments of any order (phonemes, syllables or words) according to any of these sound divisions. In this way there can be thought of a variety of applications for which Exprimulator could be used, such as the classification of environmental,

  It is also up to the user which and how expressive dimension are defined. Therefore performances of any instrument can be annotated with any form of distinguishable expression.
- Exprimulator enforces a sequential way of operating in order to obtain the annotated transcriptions. This standardization ensures that future experiments are carried out consistently.

To make Exprimulator useful for future research and other users it needs some improvements for the interface. Within this thesis the system was mostly used as an experimental tool to prove the possibility of creating annotated transcriptions. For that reason, not all of the program's features are presented in an insightful and intuitive way. Particularly tasks that need to be rounded off sequentially (creating corpus → calculating features → initializing classifiers → training classifiers) need to be indicated more explicitly in Exprimulator's GUI. In like manner the routine for annotating songs needs to be explicated: creating song corpus → transcribe songs → select classifier → calculate features → annotate song.

Exprimulator proved to be a useful tool for realizing the goals of this thesis. It could also provide its service for future research, as the system is not restricted to sound content of any specific type. To ensure that any sound corpus can be classified according to some class configuration, the set of available calculable audio descriptors and their accompanying parameters can be extended. If besides these

additions Exprimulator's GUI is improved, it would be easier operable by other users and motivate future research.

10. Future work

As a result of preceding conclusions, several extensions and optimizations that would be valuable for Exprimulator and the experimental set-up are proposed in this chapter. The optimizations and extensions incorporate 'invisible' additions and modifications to algorithms as well as added features to Exprimulator's GUI. The last section of this chapter evaluates the adequacy of Exprimulator when applied in a bigger context, for more realistic applications.

10.1. Annotating more realistic performances

The songs that have been recorded for testing the classifiers (8.1.1) have been kept basic and simple to be able to focus on the core of the problem of recognizing an expressive playing mode. As a result, the performances have not much in common with the way a guitarist plays naturally. Below, the most important musical enrichments are listed which make a performance more credible, as well as possible solutions to achieve annotation of these more complex performances:

- Monophonic → Polyphonic performances

  One of the most obvious enrichments of the performances of appendix D would be the use of polyphony (playing more than one tone simultaneously). This could bring along some serious difficulties for the recognition of timbre by machine learners, however. The current approach of incorporating every pitch in the training corpus cannot be used for recognizing polyphonic performances. The inclusion of polyphonic chords in the training corpus causes it to grow beyond acceptable limits. It is also questionable if it would result in adequate recognition, as chords can be played with any degree of arpeggio (not playing the notes of the chord simultaneously), which makes matters even more complex. A polyphonic transcription of the performance could help to guide the annotation task. The composite polyphonic spectrum of certain time frame can be dissected into individual tones according to the information of what tones are played at a certain time. These decomposed tones can be processed by a classifier that has been trained on a monophonic corpus like the one used for this thesis.

- Recognizing one single expressive dimension at a time → Recognizing multiple expressive dimensions

  In the current version of Exprimulator, only one expressive dimension is recognized at a time. In reality, a guitarist can effect variation in multiple dimensions simultaneously (for example left handed playing modes can be applied independently of right handed playing modes). In case more two or more playing modes are present in an audio signal, the expressive dimensions in which they are arranged must be independently recognized. However, it is likely that there exists some correlation between two expressive dimensions for some audio descriptors within a feature vector. This correlation could disturb recognition of one of the expressive dimensions.

  An easy solution is to create a classifier that has been trained with a training corpus that contains classes conform the multiplication of the two dimensions (the multiplicated expressive dimensions RHPP and finger configuration yields for example: {*sul tasto, neutral, sul ponticello*} x {*appoyando, tirando*} = {*sul tasto/ appoyando, neutral/ appoyando, sul ponticello/ appoyando, sul tasto/ tirando, neutral/ tirando, sul ponticello/ tirando*}). An obvious drawback is that this results in an undesired amount of classes when more than two expressive dimensions are correlated. The increase of classes leads to less accuracy for a classifier. Moreover, the resulting feature vectors of the multiplied training corpus are not guaranteed to be linearly separable. Another disadvantage is that the classifier output is compound (it specifies the playing modes in two expressive dimensions), while annotation concerning only one expressive dimension might be needed.

  Besides a solution by means of the corpus layout, audio descriptors could be calculated unique time frames or frequency bands. Signal analysis could help in finding time sections (for example the attack or sustain section), or frequency bands (1-100Hz, .. , 900-1000Hz), in which varying a certain expressive dimension affects the most radical changes in relation to other dimensions. With this

approach, the calculation of a certain audio descriptor can be optimized for each expressive dimension in a unique way, to minimize mutual disturbance. Another obvious solution is to find a unique set of audio descriptors for each expressive dimension that results in the best classification.

- Playing songs within a limited range of tones → Playing songs within the entire tonal range of the guitar

  Within the current experiments described in this thesis, only three strings of the guitar are used for the performances of appendix D (16 pitches chromatically from A2 till C4). A guitarist understandably wants to use the full scale of notes that can be realized on a guitar. In order to ensure that the corpus contains all the pitches, it has to grow with factor 2.25 (assuming the range of E2-E4 is used). The incorporation of a higher number of pitches results in a lower annotation performance. This is because the classifier has to generalize over a larger amount of training instances which are more scattered in feature space. This trend has been proven by investigating the difference of using classifiers trained and tested with tones of two strings against those using three strings. A possible solution for this problem is to divide the entire chromatic scale into several subscales for which separate classifiers are trained.

  A related issue is the overlap of possible ways to realize one pitch (an E3 for example can be played on 4 strings, because on a guitar a tone can be realized on a lower string 6 frets further). There obviously exist timbral differences between these realizations, so it is not optimal to incorporate the different realizations of one pitch in one corpus. Therefore if one decides to create separate classifiers for subscales such as proposed in the previous paragraph, it is obvious to create classifiers per string. To determine which pitch belongs to which string, a hierarchy of classifiers can be constructed in which the highest level classifier determines the string that was struck (as each string has unique ratios between sub(harmonics)). The lowest level contains classifiers that perform the actual recognition of a playing mode in an expression dimension per string.

- Dividing an expressive dimension in discrete classes → Regarding an expressive dimension as a continuous scale

  In musical scores it is common to denote expressive markup in a discrete way, as the example of RHPP illustrates (*sul tasto – sul ponticello*). In reality, a guitarist has the freedom to strike the string anywhere between the bridge and the location were a fret is pressed.

  For the proposed application of deducing expressive rules (section 1.2.3) from a corpus of annotated transcriptions, it can be desirable to annotate the transcriptions with markup that can lie in the entire range between two extreme playing modes within an expressive dimension. A scalar $\varepsilon \in [0,1]$ can specify the exact location within this range. This annotation refinement can already be realized with the current approach of offering discrete training classes to the classifiers. Classifiers that are not limited to outputting nominal classes (such as SVMs and MLPs) are able to interpolate between the classes contained in the training corpus so that in theory intermediate playing modes (such as *50% sul ponticello* or *50% sul tasto*, see 4.6) can be recognized if the classifier interpolates correctly.

10.2. Extensions to Exprimulator

In order to realize the annotation of the more complicated performances from section 10.1, the set of available calculable audio descriptors should be expanded, as also was concluded in section 9.1. The 7 timbre related MPEG-7 features did not enable a performance annotation much higher than 80% for the RHPP, which justifies additional audio descriptors. Moreover, recognizing expressive dimension simultaneously might need unique descriptors per expressive dimension to avoid mutual disturbance. There are several timbral audio descriptors which can be considered for addition, that have not been evaluated in this thesis, such as the Median frequency or the Spectral flux.

Besides the addition of new audio descriptors, existing ones can be made more optimal for separating a class configuration of a certain expressive dimension, by fine-tuning parameters that affect the calculation process of the descriptor. One should think of parameters concerning the window size over which a spectrum is calculated, the offset between two windows or the sample range within a tone that is being processed. Calculation of one descriptor can yield a vector or even a matrix of values, because of iteration

over multiple time frames and/or frequency bands. This brings along another configurable dimension: the choice of statistical measures (like *mean*, *kurtosis*, *skewness*, *median*, etc) to reduce these dimensions to a scalar. However, there is no necessity for the reduction to a scalar, if it is presumed that the entire vector contains descriptive power. This decisive freedom must be incorporated in Exprimulator's GUI to enable better control over matching audio descriptors to expressive dimensions.

10.3. Generalizing issues

In section 6.3.4 several factors are outlined that can hamper the annotation performance. The most important factors are the use of different guitars, musicians and recording sessions. Even when only one guitar is used, the classifier performance can drop drastically, as was demonstrated by the replacement of the strings (section 10.4). In like manner, one musician can cause disturbance when he/she is in a different mood. Before trying to achieve to generalize these variables, there must be defined clear bounds to what extent a classifier has to generalize. For the recognition of RHPP for example, it is questionable if a classifier can generalize over electric and acoustic guitars, as these families of instruments differ too radical in timbre.

The pursuit of making classifiers robust with respect to variable factors is a future goal to which much time can be devoted. Within the pursuit it is important to find a balance in constructing a classifier that is general as well as accurate. This means the use of a classifier that is least sensitive to overfitting (like SVM) as well as audio descriptors that are the least correlated with the variable factors.

10.4. Annotation adequacy for future applications

In the light of the applications that are proposed in section 1.2, the performances presented in chapter 8 can be judged.

One of the proposed applications was the *supervision of guitar students to learn playing techniques*. For this purpose, the recognition of Exprimulator in its present form is too sensitive to timbre changes due to different instrument properties or different recording sessions. This is illustrated by the fact that a song recorded in a different recording session than the corpus was recorded in, was annotated with scores not much higher than random annotation (33.33%). The only difference between these recording sessions was new strings on the guitar, which apparently disturbed the recognition process drastically. The strings were of the same brand (D'Addario), but were one level thicker, and produced a clearer sound because of their novelty. However the replacement of strings can effect radical timbral differences, more subtle recording session differences could also result in such drastic drops of performance. This example, in which only one guitar was used, illustrates that there lies still a challenge in recognizing playing techniques for multiple guitars and recording circumstances. An obvious solution to overcome this problem is to calibrate each different guitar by playing a few notes with the different playing techniques that a classifier has to recognize. This calibration could be used to obtain parameters for a transformation function that scales the feature vectors of the deviating instrument to coincide better with the ones of the training corpus. Besides this deficiency, also real time feature calculation should be made available so that a pupil can get feedback while playing. This is a challenging task, as note onset calculation, pitch detection, feature calculation and classifier prediction needs to be executed sequentially. The speed at which the current version of Exprimulator performs these tasks is not suitable for this purpose.

For the proposed application of *extracting rules from an annotated set of transcription that can be applied for more natural synthesis of music*, a higher annotation performance is desired. 20% of misclassified notes could likely result in unfounded or incorrect rules. A higher annotation performance can be obtained by supervising a beginning fraction of the annotation process. This means that some songs of the corpus should be revised by the user in order to adjust and refine the classifiers in case errors were made. This involves a manual calibration process such as used in the previous paragraph; with the difference that this time no new training instances are used. The classifier then should be automatically adjusted on basis of corrections made by the user.

11. Glossary of musical terms

- *Accelerando:* Gradual increase of tempo
- *Annotator:* A classifier within the context of the transcriber (i.e. that annotates the score with labels) is called an annotator.
- *Appoyando (rest stroke):* Classical guitar technique of plucking a string by which the finger leans on the adjacent string after a string is struck
- *Class:* A class corresponds in the context of this project to a specific playing technique. A group of classes can form training corpus by which a classifier can be trained.
- *Classifier:* Any machine learning algorithm that can be trained to with audio features to recognize playing techniques within this context.
- *Crescendo:* The gradual increase of sound volume
- *Descrescendo:* The opposite of *crescendo*, i.e. the gradual decrease of sound volume
- *Expressive dimension:* The scale between two complementary playing techniques in which a musician can vary
- *Flanger:* A sound technical effect by which a short delayed copy of the source signal is added to the source. The duration of this delay is modulated cyclical. [37]
- *Fret:* The mechanism on the guitar fret board that enables a guitarist to determine the length of the vibrating guitar string and thereby determining the pitch that the struck string brings forth. Each fret represents a semitone in the standard western system.
- *Glissando:* Italian musical term that signifies the glide of one tone to another
- *Label:* Textual annotation outputted by the annotator that corresponds with a certain playing technique
- *MIDI file:* a file format that contains a sequence of notes, along with several annotations such as timing, pitch, and effects. MIDI files can be played on any computer with a sound card as well as keyboards.
- *Note:* A graphical representation in a notation system of a fixed pitch with a certain duration
- *Performed signal:* The signal that is obtained by recording a guitar performance on the computer
- *Playing dimension:* All the dimensions in which a musician can vary that comprise a musical performance. This covers dimensions such as pitch, loudness, note duration but also all the expressive dimensions.
- *Playing mode: = playing technique*
- *Playing technique:* A concrete way of playing within an expressive dimension by which the musician can convey expression.
- *Right hand playing position:* An expressive dimension that is spanned by the playing modes *sul tasto* and *sul ponticello*
- *Rubato:* Slight deviations in tempo to achieve expressiveness
- *Sample:* The values that comprise a wave signal. They represent the voltage of the wave signal at a certain time frame
- *Sampler:* A software or hardware program that is capable of playing samples at different frequencies. With a programmed sequence specified in for example a MIDI file and the appropriate samples, a complete song can be synthesized.
- *Staccato:* A musical annotation which indicates that notes should be played apart from each other
- *Synthesized signal:* The signal that is obtained by using by sampling guitar tones using pitches specified in a MIDI score that contains the same song as in the performed signal.

- *Tirando (free stroke):* Classical guitar technique which is the opposite of *appoyando*. When playing with *tirando*, a string is struck from below without leaning on the subsequent string.
- *Tone:* The realization of a note with any musical instrument
- *Vibrato:* Musical playing style that is caused by slightly variation of tone pitch

References

1. *The national association for music education.* [cited; Available from: http://www.menc.org/publication/books/performance_standards/glossary.html.
2. Widmer, G. and W. Goebl, *Computational Models of Expressive Music Performance: The State of the Art.* Journal of New Music Research, 2004. **33**(3): p. 203-216.
3. Basili, R., *Classification of musical genre: a machine learning approach.* 2004.
4. Lampropoulos, A.S., P.S. Lampropoulou, and G.A. Tsihrintzis. *Musical Genre Classification Enhanced By Improved Source Separation Techniques*. in *ISMIR*. 2005. University of London.
5. Pardo, B. and W. Birmingham. *Query by Humming: How good can it get?* in *Workshop on Music Information Retrieval*. 2003. Toronto, Canada.
6. Haus, G. and E. Pollstri. *An audio front end for query-by-humming systems*. in *Proceedings of International Conference in Music Information Retrieval (ISMIR)*. 2001. Bloomington, Indiana.
7. Notess, M. and M.B. Swann. *Timeliner: Building a Learning Tool into a Digital Music Library*. in *Proceedings of ED-MEDIA*. 2004. Lugano, Switzerland.
8. Kaji, K. and K. Nagao. *MiXA: A Musical Annotation System*. in *Proceedings of the 3rd International Semantic Web Conference*. 2004. Hiroshima, Japan.
9. Tzanetakis, G. and P.R. Cook. *Experiments in computer-assisted annotation of audio*. in *Proceedings of the ICAD*. 2000. Atlanta, GE.
10. Amatriain, X., et al. *The CLAM Annotator: A crossplatform audio descriptors editing tool*. in *Proceedings of the 6th International Conference on Music Information Retrieval*. 2005. London, UK.
11. Herrera, P., et al. *Mucosa: a music content semantic annotator*. in *Proceedings of 6th International Conference on Music Information Retrieval*. 2005. London, UK
12. Gouyon, F. and P. Herrera. *Exploration of techniques for automatic labeling of audio drum tracks' instruments*. in *MOSART*. 2001.
13. Widmer, G., *Using AI and Machine Learning to Study Expressive Music Performance: Project Survey and First Report.* AI Communications, 2001. **14**.
14. Goebl, W. and G. Widmer. *Exploring expressive performance trajectories: six famous pianists play six chopin pieces*. in *Proceedings of the 8th Interna-tional Conference on Music Perception and Cognition (ICMPC'04)*. 2004. Evanston, Illinois.
15. Dixon, M., *Automatic Extraction of Tempo and Beat from Expressive Performances.* Journal of New Music Research, 2001. **30**(1): p. 39-58.
16. Arcos, J.L. and R. López de Mántaras, *An interactive case-based reasoning approach for generating expressive music.* Applied Intelligence, 2001. **14**(1): p. 115-129.
17. Dixon, S. *Towards Automatic Analysis of Expressive Performance*. in *5th Triennial Conference of the European Society for the Cognitive Sciences of Music (ESCOM5)*. 2003. Hanover, Germany.

18. Canazza, S., et al., *Sonological Analysis of Clarinet Expressivity*, in *Music, Gestalt, and Computing - Studies in Cognitive and Systematic Musicology*. 1997, Springer-Verlag. p. 431-440.
19. Eronen, A., *Automatic Musical Instrument Recognition*, in *Department of Information Technology*. 2001, Tampere University of Technology: Tampere. p. 69.
20. Tindale, A.R., et al. *Retrieval of percussion gestures using timbre classification techniques.* in *ISMIR*. 2004.
21. Herrera, P., A. Yeterian, and F. Gouyon. *Automatic Classification of Drum Sounds: A Comparison of Feature Selection Methods and Classification Techniques*. in *ICMAI '02: Proceedings of the Second International Conference on Music and Artificial Intelligence*. 2002. London, UK: Springer-Verlag.
22. Benetos, E., M. Kotti, and C. Kotropoulos. *Musical Instrument Classification using Non-Negative Matrix Factorization Algorithms and Subset Feature Selection*. 2006.
23. Simmermacher, C., D. Deng, and S. Cranefield. *Feature Analysis and Classification of Classical musical Instruments: An Empirical Study*. in *ICDM 2006*. 2006: Springer-Verlag Berlin Heidelberg.
24. Kostek, B., P. Szczuko, and P. Zwan. *Processing of Musical Data Employing Rough Sets And Artificial Neural Networks*. in *RSCTC*. 2004: Springer-Verlag Berlin Heidelberg.
25. Kostek, B., *Musical instrument classification and duet analysis employing music information retrieval techniques.* Proceedings of the IEEE, 2004. **92**(4): p. 712-729.
26. Xin, Z. and Z.W. Ras. *Differentiated harmonic feature analysis on music information retrieval for instrument recognition*. 2006.
27. Burred, J.J. and A. Lerch. *A hierarchical approach to automatic musical genre classification*. in *Proc. of the 6th Int. Conference on Digital Audio Effects*. 2003. London, UK.
28. Jhing-Fa, W., et al. *Home environmental sound recognition based on MPEG-7 features*. 2003.
29. Bonnevier, F., *Audio based Context Awareness on a Pocket PC*, in *KTH Signals Sensors and Systems*. 2006, KTH: Stockholm.
30. Dalibor Mitrovic, M.Z., and Horst Eidenberger, *Analysis of the Data Quality of Audio Descriptions of Environmental Sounds*. 2006, Institute of Software Technology and Interactive Systems: Vienna.
31. Huang, Y.-C. and S.-K. Jeng. *An Audio Recommendation System Based on Audio Signature Description Scheme in MPEG-7 Audio*. in *IEEE International Conference on Multimedia and Expo (ICME*. 2004. Taipei, Taiwan.
32. Hoque, M.E., M. Yeasin, and M.M. Louwerse. *Robust Recognition of Emotion from Speech*. in *IVA*. 2006: Springer-Verlag Berlin Heidelberg.
33. Dubnov, S., *Polyspectral analyis of musical timbre*. 1996, Hebrew University. p. 4.
34. Schouten, J.F. *The perception of timbre*. in *6th International Congress on Acoustics*. 1968. Tokyo.

35. LaMonica, F. *The Total Classical Guitar Method*. 2002 [cited; Available from: http://www.classic-guitar.com/lesson7.html.
36. *WEKA API*. [cited; Available from: http://weka.sourceforge.net/doc/.
37. *Wikipedia*. [cited; Available from: http://en.wikipedia.org.
38. Tzanetakis, G. *Audio-based gender identification using bootstrapping*. 2005.
39. Lewis, R., X. Zhang, and Z.W. Ras. *Blind Signal Separation of Similar Pitches and Instruments in a Noisy Polyphonic Domain*. in *Proceedings of Foundations of Intelligent Systems ISMIS('06)*. 2006. Bari, Italy: Springer.
40. Zhang, X. and Z.W. Ras. *Analysis of Sound Features for Music Timbre Recognition*. in *International Conference on Multimedia and Ubiquitous Engineering, 2007. MUE '07*. 2007. Seoul, Korea.
41. Chen, L., S. Gunduz, and M.T. Ozsu, *Mixed type audio classification with support vector machine*, in *ICME* 2006.
42. Gouyon, F. and P. Herrera. *Determination of the meter of musical audio signals: Seeking recurrences in beat segment descriptors*. in *AES*. 2003. Amsterdam.
43. Cuadra, P.d.l., A. Master, and C. Sapp. *Efficient Pitch Detection Techniques for Interactive Music*. in *ICMC*. 2001. Havana.
44. Sun, X. *Pitch determination and voice quality analysis using subharmonic-to-harmonic ratio*. in *International Conference on Acoustics, Speech, and Signal processing*. 2002. Florida.

Appendix A: Pitch detection

A.1. Weighted Autocorrelation:

The autocorrelation function picks peaks in the time domain to estimate the frequency, with the following formula:

$$\phi(\tau) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(n+\tau)$$

This measures the extent to which a signal correlates with a shifted version of itself. Because a periodic signal correlates strongly when the offset equals the fundamental period, we expect to find a peak at this offset value.

A.2. Harmonic Product Spectrum (HPS) [43]:

The simplest method to implement, and does well on a wide range of conditions. The HPS algorithm measures the maximum coincidence for harmonics according to equation (1) for each spectral frame,

$$Y(\omega) = \prod_{r=1}^{R} \left| X(\omega r) \right| \quad (1)$$

$$\hat{Y} = \max_{\omega_i} \{ Y(\omega_i) \} \quad (2)$$

where R is the number of harmonics to be considered, and frequency ω_i in the range of possible fundamental frequencies. The resulting periodic correlation array, $Y(\omega)$, is searched for a maximum value, $\hat{Y}$, as is shown in equation (2).

A.3. Maximum Likelihood [43]:

Searches though a set of possible ideal spectra and chooses the one which best matches the shape of the input spectrum. The algorithm is based on a preconfigured pitch resolution. Such a discrete pitch estimation makes it less suitable for estimation of the pitch of a guitar, which can produce a continuous frequency range. Besides that, it is less tolerant to noise and weak signals than HPS.

A.4. Subharmonic-to-harmonic ratio:

Sun describes in his article [44] an algorithm that employs a logarithmic frequency scale and a spectrum shifting technique to obtain the amplitude summation of the harmonics and sub-harmonic, respectively. Through comparing the amplitude ratio of sub harmonics and harmonics with the pitch perception results, the pitch of normal speech as well as speech with alternate pulse cycles (APC) can be determined. The algorithm is optimized for speech signals, but should perform for musical signals as well. The algorithm is compared to several other pitch determination algorithms, and outperformed 7 of them (HPS included). Only the enhanced super resolution pitch determinator proved to be better at pitch determination.

Appendix B: Software tools

B.1. Matlab

Matlab is a computational environment that allows easy numerical manipulation of matrices, function and data plotting and algorithmic programming. Other strengths of Matlab are that it enables design of graphical interfaces and invocation of other programming languages like Java from within the scripting language. Matlab incorporates extensive toolboxes (relating to finance, statistics, aerodynamics, mechanics, etc.) that make it popular in various scientific fields.

B.2. WEKA toolkit

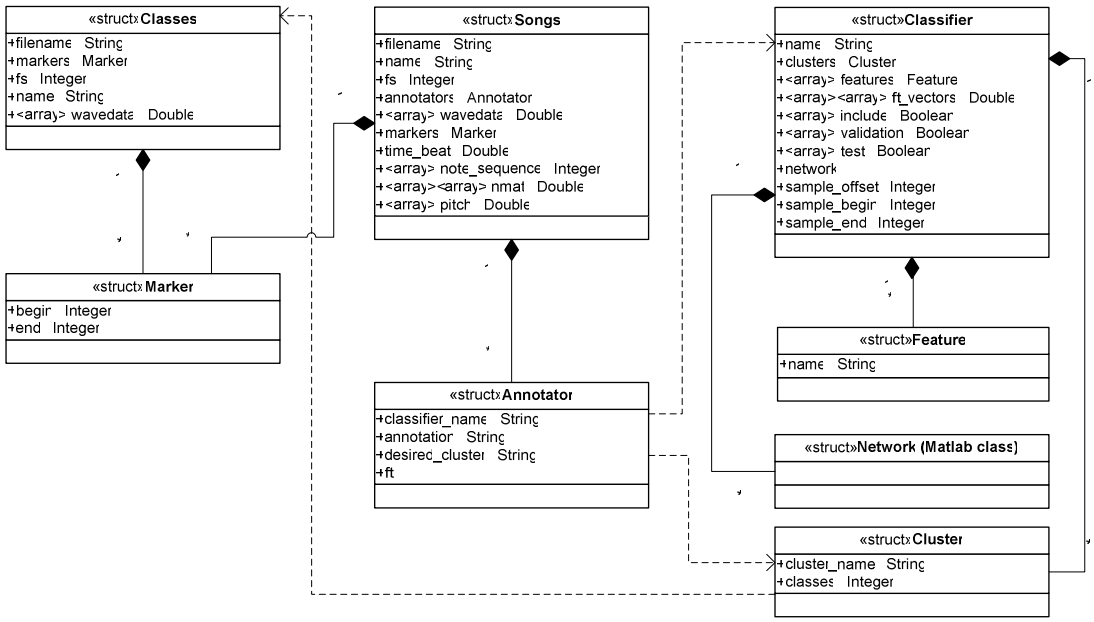
The WEKA toolkit is a collection of machine learning algorithms to solve data mining problems. It is freely available open source software, and is therefore easy adjustable and extendable with custom machine learning algorithms. It allows for analyzing, clustering, classifying and attribute selection of large datasets. The toolkit is platform independent because of its implementation in Java. A short review of sound classification research reveals that it has been used in many projects.

B.3. Matlab-XM

Matlab-XM is an unofficial toolbox written in Matlab's scripting language that implemented all MPEG-7 low level audio descriptors and high level description schemes. It also offers functionality to output audio meta-data to XML descriptions. The toolbox is maintained by Michael Casey from the University of London.

Appendix C: Class diagram Exprimulator

The there are no actual objects present in Matlab like in object-oriented languages, we tried to display the structure of the data storage via this method.



Appendix D: Songs with intentional playing techniques

1. The Beatles – Let it be, starts with D3:

find myself in times of trouble
Mother Mary comes to me
Speaking words of wisdom
Let it be-e-e-e
find myself in times of trouble
Mother Mary comes to me
Speaking words of wisdom
Let it be-e-e-e

Sul tasto
Neutral
Sul ponticello

2. Billy Joel – Uptown girl, starts with A3:

Uptown girl
Shes been living in her uptown world
I bet she never had a back street guy
I bet her mama never told her why
I'm gonna try for an uptown girl
Shes been living in her white bread world
As long as anyone with hot blood can
And now shes looking for a downtown man
Thats what I am

3. Elvis Presley – Are you lonesome tonight, starts with B2:

Are you lonesome tonight
do you miss me tonight
Are you sorry we drifted apart
Does your memory stray to a bright sunny day
When I kissed you and called you sweetheart
Do the chairs in your parlor seem empty and bare
Do you gaze at your doorstep and picture me there
Is your heart filled with pain, shall I come back again
Tell me dear, are you lonesome tonight

4. Michael row the boat ashore (Afro-American spiritual), starts with D3:

Michael row the boat ashore, hallelujah
Michael row the boat ashore, hallelujah
Michael row the boat ashore, hallelujah
Michael row the boat ashore, hallelujah

5. Davis and Charles Mitchell – You are my sunshine, starts with A2:

The other nite, dear,
As I lay sleeping
I dreamed I held you in my arms.
When I awoke, dear,
I was mistaken
And I hung my head and cried

You Are My Sunshine
My only sunshine.
You make me happy
When skies are grey.
You'll never know, dear,
How much I love you.

So please don't take my sunshine away

6. Rolling stones – paint it black, starts with C3

I see a red door and I want it painted black
No colors anymore I want them to turn black
I see the girls walk by dressed in their summer clothes
I have to turn my head until my darkness goes
I see a line of cars and theyre all painted black
With flowers and my love both never to come back
I see people turn their heads and quickly look away
Like a new born baby it just happens every day

7. Ella Fitzgerald – Every time we say goodbye, starts with E3

Everytime we say goodbye, I die a little,
Everytime we say goodbye, I wonder why a little,
Why the gods above me, who must be in the know.
Think so little of me, they allow you to go.

8. The Beatles – Michelle, starts with E3:

Michelle, ma belle.
These are words that go together well,
My Michelle.

Michelle, ma belle.
Sont les mots qui vont très bien ensemble,
Très bien ensemble.

I love you, I love you, I love you.
That's all I want to say.
Until I find a way
I will say the only words I know that
You'll understand.

9. The Beatles – Yesterday, starts with D3:

Yesterday,
All my troubles seemed so far away,
Now it looks as though they're here to stay,
Oh, I believe in yesterday.

Why she
Had to go I don't know, she wouldn't say.
I said,
Something wrong, now I long for yesterday.

10. Neil Young – Heart of Gold (starts with C3):

I want to live,
I want to give
Ive been a miner for a heart of gold.
Its these expressions I never give
That keep me searching for a heart of gold
And Im getting old.
Keeps me searching for a heart of gold
And Im getting old.