2009/2010
University of Coimbra
Faculty of Sciences and Technology
Department of Informatics Engineering

Master in Informatics Engineering - MSc. Thesis

Automatic Playlist Generation via Music Mood Analysis

Author: João André Ferro Fernandes [jaff@student.dei.uc.pt]

Supervisor: Professor Rui Pedro Paiva [ruipedro@dei.uc.pt]

[Coimbra, Portugal, 2010]

Contents

List of figures

List of tables

Acronyms

APG – Automatic Playlist Generation

BH – Beat Histogram

BPM – Beats per Minute

DBMS – Database Management System

DSC – Discrete Cosine Transform

DWCH – Daubechies Wavelet Coefficient Histogram

FFS – Forward Feature Selection

FFT – Fast Fourier Transform

GMM – Gaussian Mixture Model

GUI – Graphical User Interface

ISMIR – International Conference on Music Information Retrieval

k-NN – k-Nearest Neighbor

LS – Local Search

MARSYAS – Music Analysis, Retrieval and Synthesis for Audio Signals

MFCC – Mel Frequency Cepstral Coefficient

MER – Music Emotion Retrieval

MIR – Music Information Retrieval

MIREX – Music Information Retrieval Evaluation eXchange

MLR – Multiple Linear Regression

PATS – Personalized Automatic Task Selection

PCM – Pulse Code Modulation

PDF – Probability Distribution Function

RMS – Root Mean Square

RMSE – Root Mean Square Error

SA – Simulated Annealing

SSE – Sum-Square Error

STFT – Short-Time Fourier Transform

SVM – Support Vector Machines

SVR – Support Vector Regression

SST – Total Sum of Squares

UI – User Interface

ZCR – Zero-Crossing Rate

Acknowledgments

In the first place, I would like to thank my thesis supervisor, Professor Rui Pedro Paiva, for all the support, encouragement and guidance throughout the development of this thesis, making somewhat seem more familiar to me an area to which I was (almost) completely ignorant.

In the second place, my thesis colleague, Renato Panda, whom I must thank for all the laughs, help and encouragement, even when things seemed to go wrong. Some of his thesis important results were crucial for the success of my own thesis.

I must also mention Luís Cardoso, the BSc student who helped both me and Renato, by implementing the client side of our application with commitment and devotion.

Other persons to which I must thank include my family (my brother, my sister and my parents) who gave me support even when everything looked not so good.

Last but not least, I must thank Anita, for being the best friend in the world and for all the support and loyalty given throughout the last years - especially in the worst of times - all of my close friends (spread around the country and the world – you know who you are ;]). Also all the people that don't even know that were helping me keeping going at some stage and that cannot imagine that this is for them (sometimes these are the most important ones). I cannot forget mention Drª Ana Melo, for all the empathy and help given in this tricky road in the last year.

Finally, I must think the main reason for choosing this thesis – my addiction for music. Without this passion I would never choose this theme and wouldn't be writing this MSc thesis. So, I feel committed to thank all the artists that have pleased my ears (and my eyes too!) in the last decade. Thank you!

1. Introduction

"Music heals everything."

Hermeto Pascoal

1.1. Motivation and scope

Nowadays, music is everywhere. From small television commercials, music videos and shopping centers to the more traditional music sell (albums, both on physical or digital format, and tickets), radio play or live events (concerts, gigs, festivals, etc), we cannot help consider now it as an industry. This can be explained through economical reasons but also through emotional ones: why some concerts get sold out in just a few minutes[1], even when the tickets price is high?[1] Why are there true fan communities online which create means to bring some of their most beloved artists to their countries or cities (petitions, organizing gigs/concerts, etc)[2]? And why big part of humanitarian and charity events are based in (or incorporate) music (e.g., Live 8)?[3] Certainly not only because of temporary fashions or trends. Even some of the most popular social networks on the web are music-oriented: MySpace[4] (with more than 110 million active users[5]) and Last.fm[6] (with more than 30 million users[7]). People need to relate to the way they feel (or want to feel), and music is one of the most utilized means to achieve it. This is done not only by relating to the instrumental part of the songs, but in the case of the existence of voice, the analysis of the lyrical content.

These are only some points that relate music to mood and emotions. Later it will be defined and discussed in what the concepts of emotion and mood resemble, differ and relate to each other.

In this context, it is easy to understand that mood analysis has gained increased notoriety in the last few years, with increasing popularity coming from research in the Music Information Retrieval (MIR) field in the last decade (where the Music Emotion Retrieval (MER) area emerged). One of the most important achievements in this area

[1] http://entertainment.timesonline.co.uk/tol/arts_and_entertainment/music/article2848485.ece
[2] http://www.lugarcomum.pt/
[3] http://www.live8live.com/
[4] http://www.myspace.com/
[5] http://www.web-strategist.com/blog/2008/01/09/social-network-stats-facebook-myspace-reunion-jan-2008/
[6] http://www.last.fm/
[7] http://blog.last.fm/2009/03/24/lastfm-radio-announcement

was the inclusion, since 2007, of a Music Mood Classification evaluation contest in the 3rd Music Information Retrieval Evaluation eXchange (MIREX)[8], a part of the 8th International Conference on Music Information Retrieval (ISMIR) (ISMIR 2007)[9]. ISMIR is the most important conference dedicated to MIR in the world.

Possible MIR applications include automatic music recognition, automatic cataloging of musical pieces and automatic generation of music playlists based in a similarity criterion (mood, style, etc). The term "automatic" is used explicitly to underline the fact that these playlists and cataloging are produced based entirely on the extraction and analysis of songs features, without the use of any kind of descriptive text tags (typical ones are artist, album, genre, etc.).

In the more specific context of this thesis (mood analysis), the applications can be extended to the generation of music playlists based in certain moods, without bothering the user with the task of browsing his personal musical collection (which can easily reach (dozens of) thousands of songs) to manually choose the songs. This will simultaneously be both more comfortable and time sparing to the user. For instance, if the user wants music to play while he is jogging or doing sports (imagining the application is being used in a portable music player) he may want to hear some joyful, "happy" songs, beat-driven and/or with rhythm. On the other hand, if someone is driving he may want to be presented with some quiet, relaxing music. The same would apply to a stressful situation, where the user just wants to sit and calm down, or a psychological treatment. Even a shop owner or a DJ could benefit from this type of application, by automatically selecting cheerful music to serve as his store's background music or selecting music based on the mood of the venue he is playing in (excitement for a club where rock tunes are played traditionally, for instance), respectively. Finally, another possible daily situation where this type of software could be applied in a useful way is the selection of music for a party, where the user could choose the input mood based on the theme (or even dress code) of it.

1.2. Objectives and approaches

The main objective of this thesis is, as the title suggests, the design and implementation of an application that provides the user with an automatically generated playlist of

8 http://www.music-ir.org/mirex2007/index.php/Audio_Music_Mood_Classification

9 http://ismir2007.ismir.net/

songs based on the mood of a song specified by the user and given as input. In another perspective, it can be used to generate a playlist to complement the mood of the user – e.g., the user introduces that his mood is anxious, it will be expected that the playlist will include some relaxing music. The song given as input will be an existing song in the application's database (through a query). The selection would be done based on a database consisting of several music files/musical pieces. The pieces that would fit the mood specified by the user (or the pieces that would complement the specified mood) would be added to the final playlist presented to it. The user would then be presented with several options to customize these operations. Details about how this will be done are referred to and discussed later in this document.

The approach carried out on this thesis involves two aspects: it is both an engineering and a research problem. It is a typical engineering problem in what concerns both to the paradigm (a client-server application, with a backoffice for database/application management) and to the software development process used – in this case, the waterfall process, with the typical phases associated to it: requirements specification, design, implementation, model validation and testing/debugging. The investigation started with the bibliographic search that was made and that will be detailed later on this report. The research also focused on the choice of the framework to be used to extract features from songs. From the three frameworks analyzed (jAudio[10], Music Analysis, Retrieval and Synthesis for Audio Signals (MARSYAS)[11] and MIRtoolbox[12], a toolbox for MATLAB), the one that proved to be faster was MARSYAS (performance results in the MIREX 2008 context prove it)[13], since it is implemented in highly optimized C++ code. It is also important to underline that although both MARSYAS and jAudio are open source, the latter one, as the name suggests, is implemented in Java, which must certainly be the main factor for not being as fast as MARSYAS. On the other hand, MIRtoolbox uses MATLAB, which is also partially implemented in Java, making it heavy in what concerns to computational requirements and also much slower than MARSYAS. Also, from the three frameworks compared, only MATLAB is not open source. So, MARSYAS was naturally chosen as the framework to be used within this project. Since it is open source, it is possible to add other functionalities to the application in the future (for instance, new features to extract, new classifiers, etc). The application also has a Graphical User Interface (GUI), which

[10] http://jaudio.sourceforge.net/

[11] http://marsyas.info/

[12] https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

[13] http://www.music-ir.org/mirex/2008/index.php/Audio_Music_Mood_Classification_Results#MIREX_2008_Audio_Mood_Classification_Run_Times

was implemented in Qt[14] (a fast and open source application for creating interfaces). This decision was basically based on the fact that Qt is a cross-platform application and UI framework. Also, MARSYAS is natively prepared for seamless integration with Qt.

In this work, we follow a classification-based song similarity analysis. A classifier is first trained and then song similarity is calculated based on the distance between songs in Thayer's arousal-valence plane. Here, Support Vector Machines (SVM) are employed. Moreover, an extensive research of the features available in the three frameworks analyzed was conducted. This research is presented later in the document. As mentioned, the emotional model to be used is the Thayer model, which will also be discussed in detail later.

In what concerns to the GUI, the application presents the user with a graphical representation of the Thayer's model, where songs are represented as dots in the arousal / valence plane, and each of the four quadrants has an appropriated color (e.g. blue for depression and green for contentment, with a color gradient inside each quadrant). Also, for the creation of the playlist, a playlist path in the Thayer's plane is drawn by the user, from which songs are obtained.

The application that is the subject of this thesis was co-developed with a colleague, Renato Panda. The core of the application is similar, although Renato's thesis is focused on automatic mood tracking in audio music (mood change analysis during one song), opposed to this thesis theme (automatic playlist generation via music mood analysis). The main core of the application was developed in team work, while the functionalities related to playlist generation were the subject of this thesis and functionalities related to mood tracking were implemented by Renato.

1.3. Planning

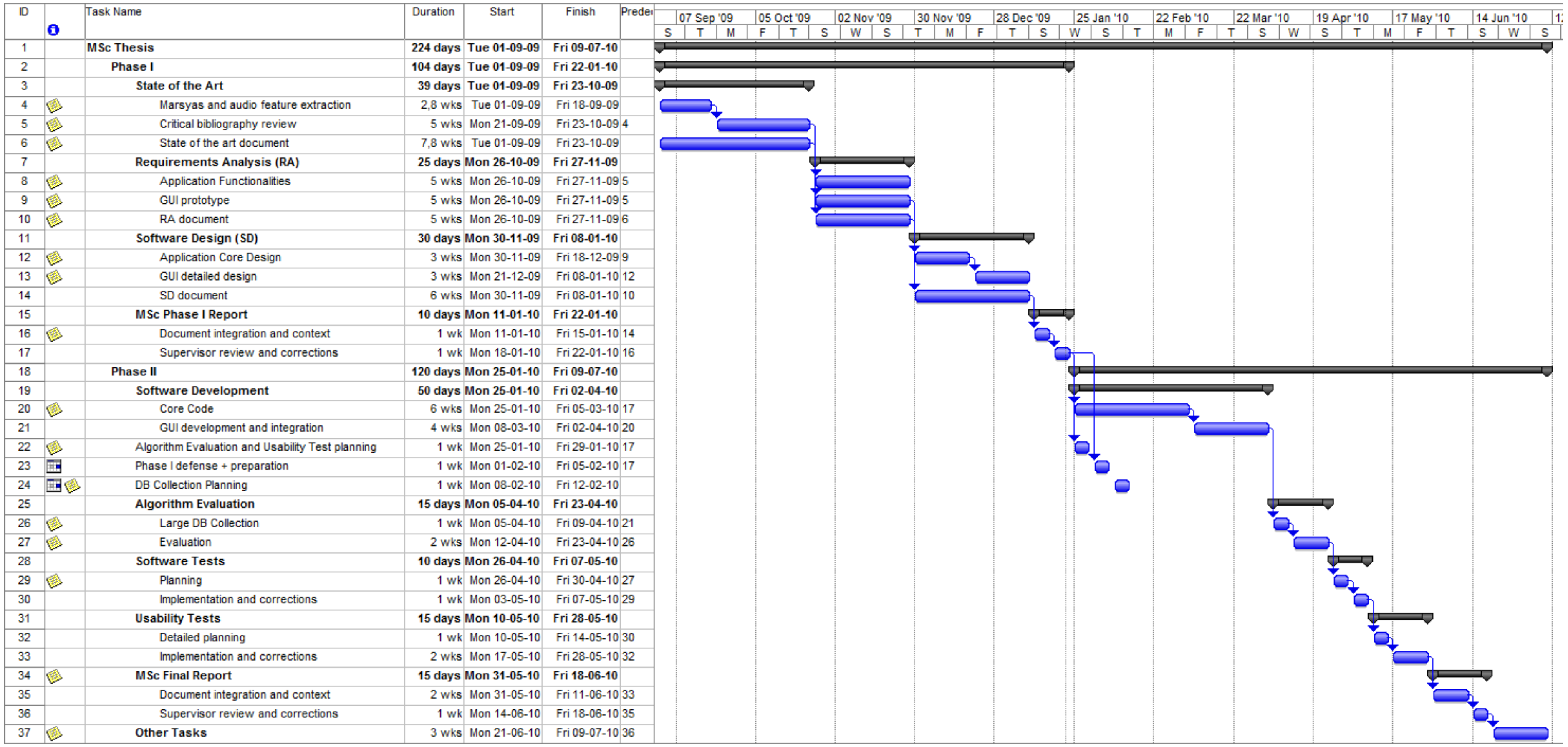
The planning and schedule of this thesis is now presented. It suffered some changes, basically due to the inclusion of new elements on the project and to the update on the application requirements (the project grew in dimension over time). It is important to notice that the time that was initially planned for the state of the art document revealed short, since the bibliographic research and reading proved to be more time consuming. Also, some delay was introduced due to software experimentation (more specifically

[14] http://qt.nokia.com/

MARSYAS) – although its open source nature, documentation is almost inexistent (some manuals exist, but they are still incomplete). So, on the final planning Gantt diagram some of the phases of the project became overlapped in some degree.

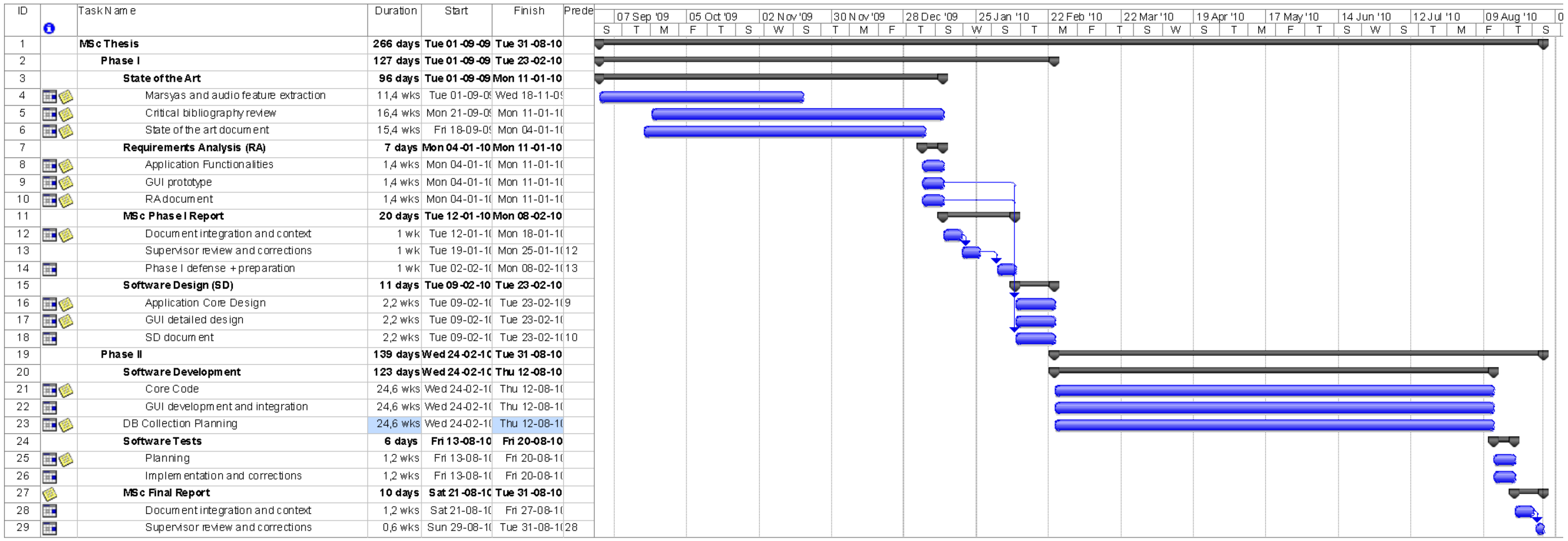
The second part (semester) consisted mainly in the application implementation, evaluation and testing. Also, some evaluation tests were performed, in what concerns to classification accuracy and playlist generation.

The initial planning and the current one are now presented below (Figure 1 and Figure 2, respectively) as Gantt diagrams:



ID	ℹ	Task Name	Duration	Start	Finish	Predec
1		**MSc Thesis**	**224 days**	**Tue 01-09-09**	**Fri 09-07-10**	
2		**Phase I**	**104 days**	**Tue 01-09-09**	**Fri 22-01-10**	
3		**State of the Art**	**39 days**	**Tue 01-09-09**	**Fri 23-10-09**	
4	📝	Marsyas and audio feature extraction	2,8 wks	Tue 01-09-09	Fri 18-09-09	
5	📝	Critical bibliography review	5 wks	Mon 21-09-09	Fri 23-10-09	4
6	📝	State of the art document	7,8 wks	Tue 01-09-09	Fri 23-10-09	
7		**Requirements Analysis (RA)**	**25 days**	**Mon 26-10-09**	**Fri 27-11-09**	
8	📝	Application Functionalities	5 wks	Mon 26-10-09	Fri 27-11-09	5
9	📝	GUI prototype	5 wks	Mon 26-10-09	Fri 27-11-09	5
10	📝	RA document	5 wks	Mon 26-10-09	Fri 27-11-09	6
11		**Software Design (SD)**	**30 days**	**Mon 30-11-09**	**Fri 08-01-10**	
12	📝	Application Core Design	3 wks	Mon 30-11-09	Fri 18-12-09	9
13	📝	GUI detailed design	3 wks	Mon 21-12-09	Fri 08-01-10	12
14		SD document	6 wks	Mon 30-11-09	Fri 08-01-10	10
15		**MSc Phase I Report**	**10 days**	**Mon 11-01-10**	**Fri 22-01-10**	
16	📝	Document integration and context	1 wk	Mon 11-01-10	Fri 15-01-10	14
17		Supervisor review and corrections	1 wk	Mon 18-01-10	Fri 22-01-10	16
18		**Phase II**	**120 days**	**Mon 25-01-10**	**Fri 09-07-10**	
19		**Software Development**	**50 days**	**Mon 25-01-10**	**Fri 02-04-10**	
20	📝	Core Code	6 wks	Mon 25-01-10	Fri 05-03-10	17
21		GUI development and integration	4 wks	Mon 08-03-10	Fri 02-04-10	20
22	📝	Algorithm Evaluation and Usability Test planning	1 wk	Mon 25-01-10	Fri 29-01-10	17
23	📅	Phase I defense + preparation	1 wk	Mon 01-02-10	Fri 05-02-10	17
24	📅 📝	DB Collection Planning	1 wk	Mon 08-02-10	Fri 12-02-10	
25		**Algorithm Evaluation**	**15 days**	**Mon 05-04-10**	**Fri 23-04-10**	
26	📝	Large DB Collection	1 wk	Mon 05-04-10	Fri 09-04-10	21
27	📝	Evaluation	2 wks	Mon 12-04-10	Fri 23-04-10	26
28		**Software Tests**	**10 days**	**Mon 26-04-10**	**Fri 07-05-10**	
29	📝	Planning	1 wk	Mon 26-04-10	Fri 30-04-10	27
30		Implementation and corrections	1 wk	Mon 03-05-10	Fri 07-05-10	29
31		**Usability Tests**	**15 days**	**Mon 10-05-10**	**Fri 28-05-10**	
32		Detailed planning	1 wk	Mon 10-05-10	Fri 14-05-10	30
33		Implementation and corrections	2 wks	Mon 17-05-10	Fri 28-05-10	32
34	📝	**MSc Final Report**	**15 days**	**Mon 31-05-10**	**Fri 18-06-10**	
35		Document integration and context	2 wks	Mon 31-05-10	Fri 11-06-10	33
36		Supervisor review and corrections	1 wk	Mon 14-06-10	Fri 18-06-10	35
37	📝	**Other Tasks**	3 wks	Mon 21-06-10	Fri 09-07-10	36

Figure 1: Initial planning Gantt diagram



ID	Task Name	Duration	Start	Finish	Prede
1	**MSc Thesis**	**266 days**	**Tue 01-09-09**	**Tue 31-08-10**	
2	**Phase I**	**127 days**	**Tue 01-09-09**	**Tue 23-02-10**	
3	**State of the Art**	**96 days**	**Tue 01-09-09**	**Mon 11-01-10**	
4	Marsyas and audio feature extraction	11,4 wks	Tue 01-09-0	Wed 18-11-0	
5	Critical bibliography review	16,4 wks	Mon 21-09-0	Mon 11-01-1	
6	State of the art document	15,4 wks	Fri 18-09-0	Mon 04-01-1	
7	**Requirements Analysis (RA)**	**7 days**	**Mon 04-01-10**	**Mon 11-01-10**	
8	Application Functionalities	1,4 wks	Mon 04-01-1	Mon 11-01-1	
9	GUI prototype	1,4 wks	Mon 04-01-1	Mon 11-01-1	
10	RA document	1,4 wks	Mon 04-01-1	Mon 11-01-1	
11	**MSc Phase I Report**	**20 days**	**Tue 12-01-10**	**Mon 08-02-10**	
12	Document integration and context	1 wk	Tue 12-01-1	Mon 18-01-1	
13	Supervisor review and corrections	1 wk	Tue 19-01-1	Mon 25-01-1	12
14	Phase I defense + preparation	1 wk	Tue 02-02-1	Mon 08-02-1	13
15	**Software Design (SD)**	**11 days**	**Tue 09-02-10**	**Tue 23-02-10**	
16	Application Core Design	2,2 wks	Tue 09-02-1	Tue 23-02-1	9
17	GUI detailed design	2,2 wks	Tue 09-02-1	Tue 23-02-1	
18	SD document	2,2 wks	Tue 09-02-1	Tue 23-02-1	10
19	**Phase II**	**139 days**	**Wed 24-02-1**	**Tue 31-08-10**	
20	**Software Development**	**123 days**	**Wed 24-02-1**	**Thu 12-08-10**	
21	Core Code	24,6 wks	Wed 24-02-1	Thu 12-08-1	
22	GUI development and integration	24,6 wks	Wed 24-02-1	Thu 12-08-1	
23	DB Collection Planning	24,6 wks	Wed 24-02-1	Thu 12-08-1	
24	**Software Tests**	**6 days**	**Fri 13-08-10**	**Fri 20-08-10**	
25	Planning	1,2 wks	Fri 13-08-1	Fri 20-08-1	
26	Implementation and corrections	1,2 wks	Fri 13-08-1	Fri 20-08-1	
27	**MSc Final Report**	**10 days**	**Sat 21-08-1**	**Tue 31-08-10**	
28	Document integration and context	1,2 wks	Sat 21-08-1	Fri 27-08-1	
29	Supervisor review and corrections	0,6 wks	Sun 29-08-1	Tue 31-08-1	28

Figure 2: Final Gantt diagram

2. Mood-based playlist generation in audio music: context and overview

2.1. Mood and emotion

Although at first glance mood and emotion can be easily confused, the two concepts are considered to be different. As a subjective theme, everyone knows what an emotion is, but rarely one can define it. Since then, psychologists have made several attempts to achieve definitions of emotion and mood and a reliable model of human emotion in the mind.

In the MSc thesis "A Mood-Based Music Classification and Exploration System" by Meyers (Meyers, 2007), emotion is defined as:

> "Emotion is a complex set of interactions among subjective and objective factors, mediated by neural/hormonal systems, which can (a) give rise to affective experiences such as feelings of arousal, pleasure/displeasure; (b) generate cognitive processes such as perceptually relevant effects, appraisals, labeling processes; (c) activate widespread physiological adjustments to the arousing conditions; and (d) lead to behavior that is often, but not always, expressive, goal-oriented, and adaptive."

In the same thesis, mood is interpreted as something more specific, shorter emotional state and that can be (indirectly) influenced by the surroundings, the "environment" around the individual, so to speak. It can be attributed to a particular stimulus and usually has a prolonged effect.

2.1.1. Mood taxonomies

Among the several papers addressed in the last paragraphs, many distinct mood categories and taxonomies where proposed. One typical and common problem in this area is the existence of dozens (even hundreds or thousands) of different words or terms to describe moods, most them describing somewhat similar and redundant ones. Usually these words are adjectives, but there is the need of normalizing the terms used, since there is not a standard mood taxonomy. Mood taxonomies can be grouped in two main approaches: categorical and dimensional.

The first one (the categorical approach), consists of several different classes of emotional states (e. g., Hevner model). On the other hand, dimensional models maps emotional states along several axes (e.g. Thayer and Tellegen-Watson-Clark models). Categorical models can be classified further into discrete or continuous, since emotions inside a quadrant (in the case of a bi-dimensional model, with 2 axes) can still be ambiguous (happiness, excitement or pleasure, for instance, are different in nature, but all mapped to the high arousal and high valence quadrant). More precisely, discrete models are category-like in nature (as described before), while continuous models view the emotion plane as a continuous space and recognize each point as an emotional state. This way, the aforementioned ambiguity is solved. However, arousal and valence are not always independent.

In some cases the same model can be interpreted as discrete or continuous, depending on the way it was implemented (e. g., Thayer's model).

Some of the most common models are now presented: Hevner, Thayer and Tellegen-Watson-Clark.

Hevner model

One of the first approaches to present an affective model was made by Kate Hevner, circa 1936. Hevner mapped a group of 67 adjectives divided into eight different emotional categories (each one containing from 6 to 11 adjectives), which were mapped into a circular model. The main emotions in each emotional category are dignified, sad, dreamy, serene, graceful, happy, exciting and vigorous (from group 1 to 8, respectively) (Meyers, 2007). The categorical model is represented in Figure 3:

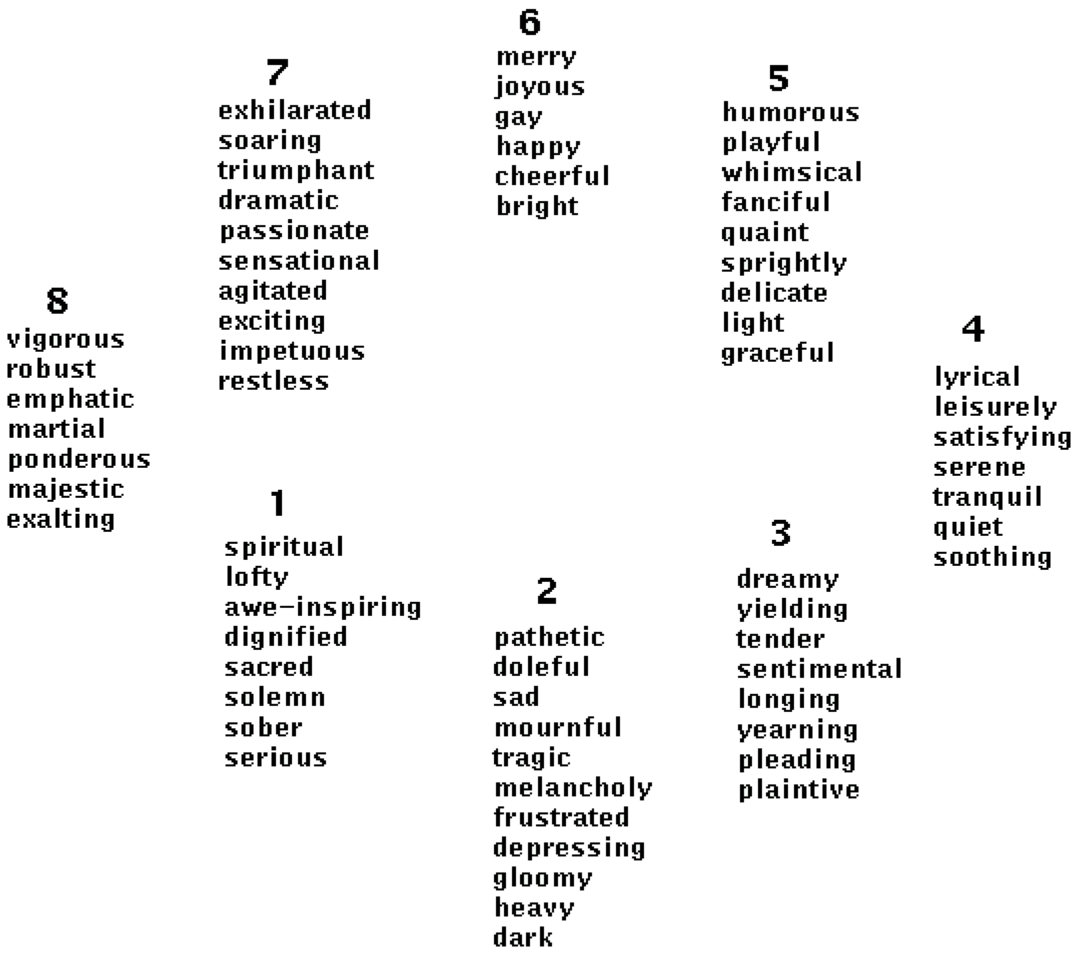


Figure 3: Hevner's circular model (Meyers, 2007)

Hevner also made pioneering studies that tried to relate emotions and moods (the ones represented in her circular model) to music – six musical features (mode, tempo, pitch, rhythm, harmony and melody) were studied, with main conclusion that music and emotions were indeed connected and that music clearly carries an emotional meaning (Laar, 2005).

Thayer's model

A popular and more recent approach is based on the simple Robert Thayer's Model ((Meyers, 2007), (Lu, Liu, & Zhang, 2006), (Laar, 2005)), a dimensional model where a musical piece can be classified in one of four categories mapped into a four-quadrant, bi-dimensional space, as seen in Figure 4, below:

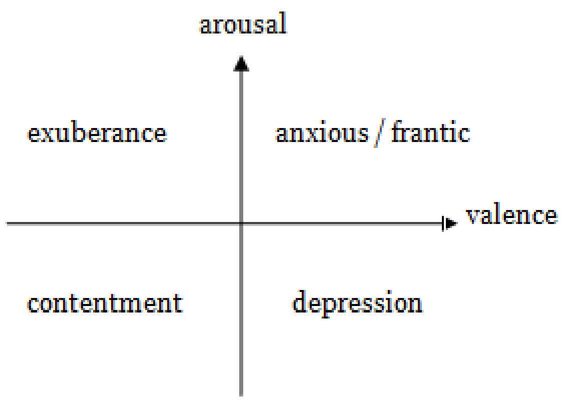


Figure 4: Thayer's dimensional model (Lu, Liu, & Zhang, 2006)

This approach has its advantages, since although simple it relates two different dimensions effectively, and so each category could be the result of some combination of the valence and arousal components. In the horizontal axis (valence) the quantity of stress is measured, while the vertical axis (arousal) denotes the quantity of energy. Although the model consists of four basic moods, the relation between the two amounts (coordinates in each axis) can determine an exact position in the model, which then can be represented by a point (x, y).

Tellegen-Watson-Clark model

In 1999, the Tellegen-Watson-Clark model was proposed. This is a more complex model, as it can be seen in Figure 5. It contains a lot more moods and emotions than Thayer's circular model. The idea behind this dimensional model is to relate positive/negative affective rate as one dimension and pleasantness/unpleasantness versus engagement/disengagement (which are 45º rotated) as another. This is more understandable looking at Figure 5, where the model is represented:

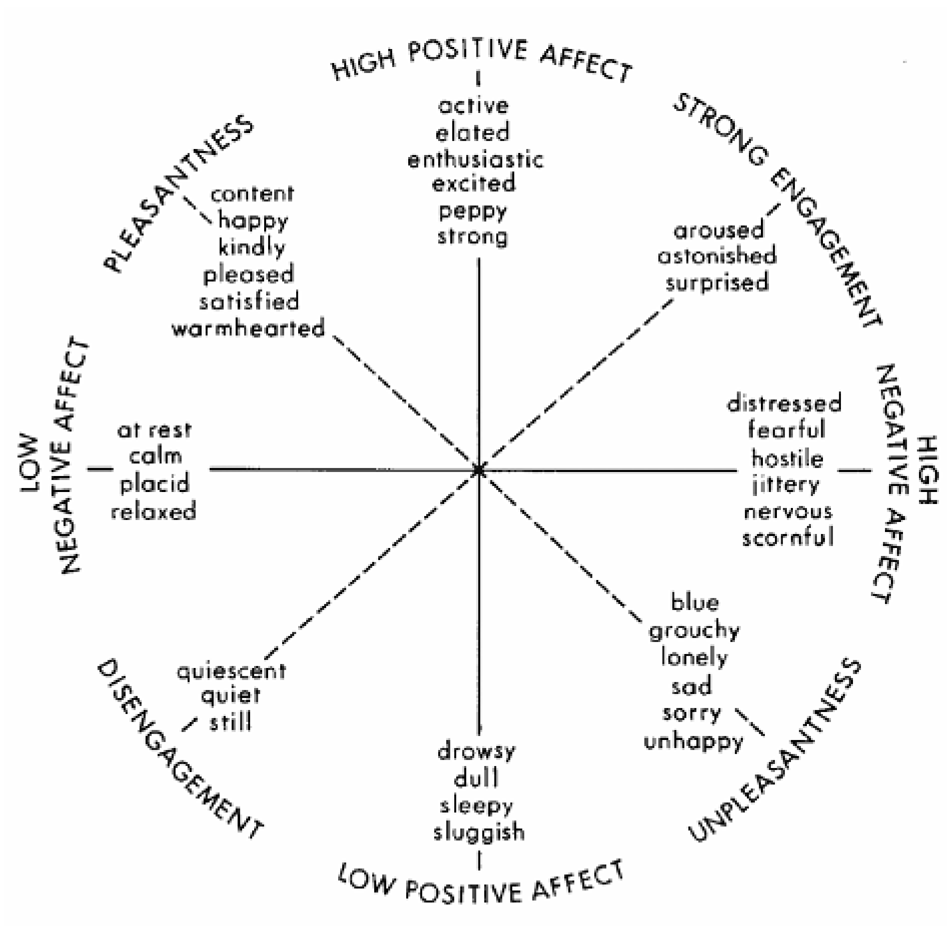


Figure 5: Tellegen-Watson-Clark mood model (Laar, 2005)

These were the most representative and popular models that were found in the bibliography. Other works include Russell, Rigg and Watson (Meyers, 2007).

2.2. Mood analysis in MIR research

The present work has as its main goal, as the title suggests, the ability to automatically generate a music playlist according to the mood specified by the user (or, in another perspective, it can be used to generate a playlist to complement the mood of the user – e.g., the user introduces that his mood is anxious, it will be expected that the playlist will include some relaxing music). This selection would be done based on a database consisting of several music files/musical pieces. The pieces that would fit the mood specified by the user (or the pieces that would complement the specified mood) would be added to the final playlist presented to the user.

In particular, this section will analyze the state of art in what concerns to MIR platforms, software tools, feature identification, similarity analysis, playlist creation and algorithm evaluation methods, comparing results and presenting features.

2.2.1. Research in the area

Although this project operates in a relatively recent field, where the investigation record is still small, there are already some interesting papers and results regarding this subject. The following paragraphs will analyze current investigation in what concerns to mood analysis, similarity and playlist generation, three key concepts in this project.

Mood

As written before, this area, although recent, has already produced new, fresh information. Much is to be found, but also much has already been achieved by now. Some of the papers that inspire this thesis are mentioned in the next paragraphs in what concerns to the mood categories, features and classification used. The results presented in these papers are also discussed.

One of them is "Automatic Mood Detection and Tracking of Music Audio Signals" (Lu, Liu, & Zhang, 2006). It proposes music mood classification with four main clusters: anxious/frantic, depression, contentment and exuberance (based on Thayer's model of mood). In what concerns to features extraction, it proposes the use of three different feature sets: intensity, timbre and rhythm. The paper experiments with two different frameworks for mood detection: hierarchical and non-hierarchical. In both cases, it is used a GMM along with training data. The results presented showed better results for the hierarchical framework.

Another paper (Laar, 2005) discusses and collects information from different papers, regarding different aspects from mood detection in musical pieces. Regarding to mood categories, they range from the four proposed in the paper above to more complex systems (positive/negative affect as one dimension and the pleasantness/unpleasantness versus engagement/disengagement as the other). Another algorithm categorized the songs in two classes (Hostility, Sadness, Guilt and

Love, Excitement, Pride). The paper addresses timbral texture features (centroid, roll off, spectral flux, zero crossings and average silence ratio), tonality coefficient, spectral crest factor, Mel Frequency Cesptral Coefficients (MFCC), Daubechies Wavelet Coefficient Histogram (DWCH), beat and tempo detection, genre information, lyrics, pitch content features, Beats per Minute (BPM) detection and Sum of Absolute Values of the Normalized Fast Fourier Transform (FFT). Classification methods include the use of a neural network with three layers, a GMM (as explained above). The results achieved here showed that there is not "an absolute winner", ranging from medium to high precision, depending on the relation between granularity (number of categories) and diversity of the music database.

The paper "A Regression Approach to Music Emotion Recognition ", by Yang et al. (Yang, Lin, Su, & Chen, 2006) addresses mood categories with the innovation of introducing coordinates in the Thayer's model (arousal and valence axis), making it from a continuous perspective. The features extracted were divided into 2 feature sets, one with 114 and another one with a selection of 15 of them. In what concerns to classification, it is addressed in a representative way, and so the system trains 3 different regression algorithms to directly predict arousal and valence values (a, v): multiple linear regression (MLR), support vector regression (SVR) and AdaBoost.RT (BoostR). The results presented in this paper are based in the R^2 statistics, where the best combination of data and feature space reaches 58.3% for arousal and 28.1% for valence.

Finally, in the MSc thesis "A Mood-Based Music Classification and Exploration System" by Meyers (Meyers, 2007), the objective was to analyze not only mood on audio signal of musical pieces but also add mood analysis of the song's lyrics. The output would be a playlist featuring music with similar mood. The author opted to implement categories based on Russell's circumplex model of emotion. Features were extracted in conjunction with Hevner's original mapping of musical features to an emotional space, with slight differences (4 of the 6 features were used – mode, tempo, rhythm and harmony were used, melody and pitch were discarded), plus loudness. Music classification is performed in two steps: a preliminary classification of the song database is made through a decision tree and then it is used the k-NN classifier. The results achieved were reasonable, although the author did not revealed classification statistics: instead, some information about the classification of the music database, lyrics classification, classification vs. music classification experts, classification vs. social tagging services and user evaluation was provided.

Similarity

In what concerns to similarity (basically, the computation of audio and web-based music similarity), two main documents were approached. They are summarized in the next paragraphs.

The tutorial “Music Similarity”, by Elias Pampalk (Pampalk, 2005), refers that music similarity is not only subjective but also context-dependent, with important dimensions such as instrumentation, timbre, melody, harmony, rhythm, tempo, mood, lyrics and social background. The basic schema for this evaluation between two songs consists in the input given – the representation of the song (e. g. Pulse Code Modulation (PCM)), where feature extraction is made. Then the distances between the two songs are computed and compared, using a specified metric (typically Euclidean distance). The tutorial refers experimentation with different features: Zero Crossing Rate (ZCR), MFCC, spectral similarity, fluctuation patterns, chroma complexity and harmony (higher level). The author also points out limitations – or how 100% accuracy is impossible to achieve, because even human experts do not agree always, and some of the aspects pointed out in the article are almost or totally impossible to extract, like sociocultural background, lyrics, mood. It is also pointed out that maybe a perfect similarity measure for applications, at least, is not desirable.

Another paper is “Music Similarity Measures: What’s The Use?”, by Autocourier et al. (Aucouturier & Pachet, 2002), which proposes a timbral similarity measure (applied to a whole song), based on a Gaussian model of cepstral coefficients (basically, using MFCC modeled with GMM). The application folds in two parts: a timbre extractor (an algorithm that creates a representation of the timbre of a song) and a descriptor, which outputs the proposed timbral similarity measure. The aforementioned timbral measure between two songs can be obtained in two ways: likelihood (matching the probability that the MFCCs of the first song can be generated by the model of the second one, using GMM) or sampling (in the case of not being possible to access the MFCCs of a song while computing the distance, sampling of both GMMs are compared, applying the first method – likelihood – to the mentioned models; Consequently, this method is much more efficient memory-wise). Conducted benchmarking experimentations included the comparison between duplicated songs, songs from the same artist or genre. Then, for each song in the database its timbral distances to all the other songs were computed, comparing these results to textual data (the genre of the titles). These results revealed very poor, and that led to a subjective test, where users were presented with a target song S and 2 songs, A (more closer to S) and B (more distant), and had to order the songs in what concerns to distance to S. The test was performed resorting to ten users and the results showed that about 80% of the songs were well ordered by the application.

Playlist generation

Having in mind that the application to be developed is going to not only classify a chosen song in what concerns to its mood, but also generate a playlist of songs with similar mood, it was crucial to do some investigation on this subject.

In "PATS: Realization and User Evaluation of an Automatic Playlist Generator" (Pauws & Eggen, 2002), it was proposed a system called Personalized Automatic Task Selection (PATS) to generate music playlists satisfying a specified context of use (the real-world environment in which the music is heard). The application creates playlists through a dynamic clustering method, grouping songs based on their attribute similarity. The attribute values are weighted according to their importance to the specified context-of-use. These weights are achieved with an inductive learning algorithm, based on the input given by the user, as feedback. The quality of the generated playlists was evaluated in a controlled user experiment were two different contexts of use were proposed. The results were compared to random generated playlists. The aforementioned playlist quality was measured not only by precision (songs that suit the given context-of-use) but also by coverage (songs that suit the context-of-use but that were not already in previous playlists) and by a rating system. In all these three measures, results revealed that PATS was able to generate not only playlists with more music that suits the given context-of-use (precision), but also more diversified (coverage) and higher rated than random assembled playlists. Although the attributes used were basically tags (which are not to be used at all in this application) and requires some user feedback (which is not foreseen to be used in the application subject of this thesis, although that option is not totally closed), the approach to give different weights to the attributes of a song (in the context of this thesis work, the extracted musical features) looks interesting, and can be of great value to this work.

The MSc thesis "Local Search for Automatic Playlist Generation" (Vossen, 2005), also addresses playlist generation based on similarity comparison between songs, in which song attributes (tags) are also used. However, as the name of the project thesis suggests, the playlist generation is automatic, with little feedback from the users, which makes it of higher relevance for this project (in the case of PATS (Pauws & Eggen, 2002), the user could indicate which songs did not fit the context of use – in this case the user can only specify constraints to generate the playlist and adjust them till the output fits the users desires). Also, the introduction of restrictions by the user in the generation of the desired playlist is the bigger improvement in this work, making it apparently much more complex than the one presented in (Pauws & Eggen, 2002). These restrictions are translated into the Automatic Playlist Generation (APG) algorithm as constraints, which can be of three types: unary, binary and global. Also, this work introduces a *penalty function*, which evaluates to what extent the restrictions specified by the user are violated in the playlist generated. More than this, the penalty function is used in conjunction with a *constraint penalty function*, which evaluates, for each constraint specified, how much the playlist violates it. Finally, a *constraint weight function* expresses the importance of each constraint in the development of the playlist and a *constraint transformation function*, which returns how important a constraint violation is to the generated playlist. To improve on the APG algorithm, a local search (LS) algorithm was used - simulated annealing (SA), since it has the ability to skip local optima while searching the solution space. The results achieved by this algorithm are very interesting, not only both in execution time and scalability but also in mean penalties in the playlists generated.

In conclusion, it is understandable that both solutions have useful concepts to the present work, although (Vossen, 2005) is more similar to the system that is intended to be achieved.

2.3. Audio features

In this section all the features addressed in the research conducted throughout three different feature extraction software tools (jAudio, MARSYAS and MIR toolbox) and some papers, which include the ones referenced in the previous part of this section, will be presented and explained with some detail. As mentioned before, this is an extensive list since it is unknown at this point which musical features are going to be used in this project or which ones are dispensable.

Features were divided into four categories: intensity, pitch, rhythm and timbre.

2.3.1. Intensity

Root Mean Square

Root Mean Square (RMS) returns the power of a signal over a window (McKay, 2005). It can be computed simply by taking the root average of the square of the amplitude, (RMS) (Lartillot, 2008):

$$= \quad -$$

Root Mean Square derivative

This feature measures the window-to-window change in RMS, serving as an indication of change in signal power (McKay, 2005).

Root Mean Square variability

This feature outputs the standard deviation of the RMS of the last 100 windows (McKay, 2005).

Less-than-average energy

An assessment of the temporal distribution of energy can be obtained through the energy curve, in order to see if it remains constant throughout the signal, or if some frames are more contrastive than others. One way to estimate this consists in computing the low energy rate, *i.e.* the percentage of frames showing less-than-average energy (Lartillot, 2008). Figure 6 shows the visualization of this feature (Lartillot, 2008):

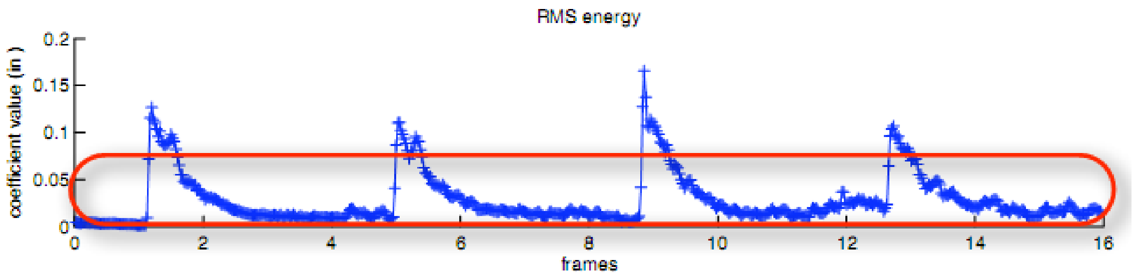


Figure 6: Selected part of the energy curve sows a high value for less-than-average energy feature value (Lartillot, 2008)

Fraction of low energy frames

The fraction of the last 100 windows where the RMS value is less than the mean RMS of the last 100 windows. This can indicate how much of a signal section is quiet relative to the rest of the signal section (McKay, 2005).

2.3.2. Pitch

Pitch (F0)

Pitch (as an audio feature) typically refers to the fundamental frequency of a monophonic sound signal and can be calculated using various different techniques. It is a subjective property of sound that can be used to order sounds from low to high and is typically related to the fundamental frequency (Tzanetakis, 2002). One of the methods employed in MARSYAS to estimate pitch uses the YIN algorithm, which is based on the autocorrelation method with a number of modifications that combine to prevent errors (de Cheveigné & Kawahara, 2002).

Strongest frequency via FFT Maximum

An estimate of the strongest frequency component of a signal, in Hz, achieved via finding the FFT bin with the highest power (McKay, 2005).

2.3.3. Rhythm

Beat sum

This feature consists in the sum of all bins in the beat histogram. This is a good measure of the importance of regular beats in a signal (McKay, 2005).

Rhythmic fluctuation

One way to estimate the rhythmic content of a signal is based on spectrogram computation transformed by auditory modeling and then spectrum estimation in each band (Lartillot, 2008). The result of the three phases is illustrated in Figure 7:

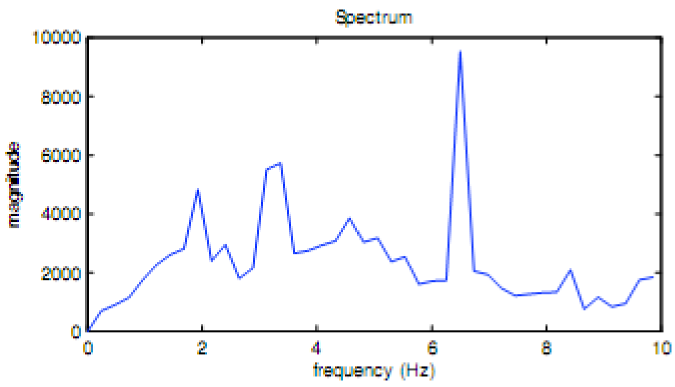


Figure 7: Spectrum summary showing the global repartition of rhythmic periodicities (Lartillot, 2008)

Strongest of strongest beat

This feature measures how strong the strongest beat (the strongest beat in a signal is, in BPM, achieved by finding the highest bin in the beat histogram) is compared to other potential beats (McKay, 2005).

Tempo

This feature consists in the speed (or pace) of a given musical piece - in modern music is indicated in BPM. Its value is estimated by detecting periodicities from the onset detection curve, as exemplified in Figure 8 (Lartillot, 2008):

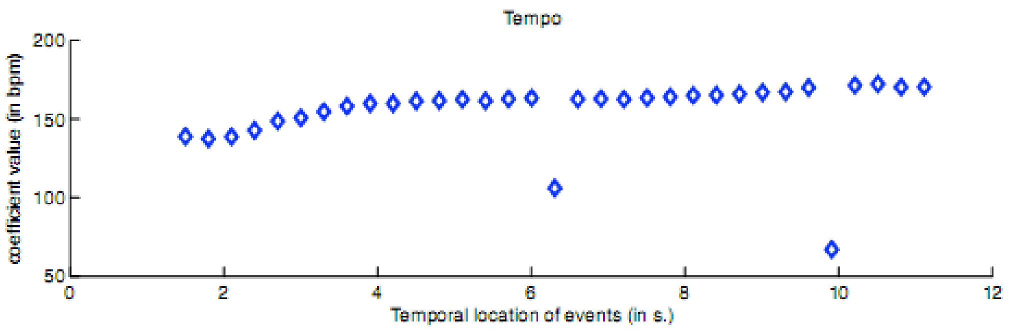


Figure 8: Tempo curve *t* (Lartillot, 2008)

2.3.4. Timbre

Attack time

Attack time is the estimation of temporal duration for a signal to rise to its peak (e.g., in amplitude). One simple way of describing and compute this feature consists in estimating the attack phase temporal duration - see Figure 9 below (Lartillot, 2008):

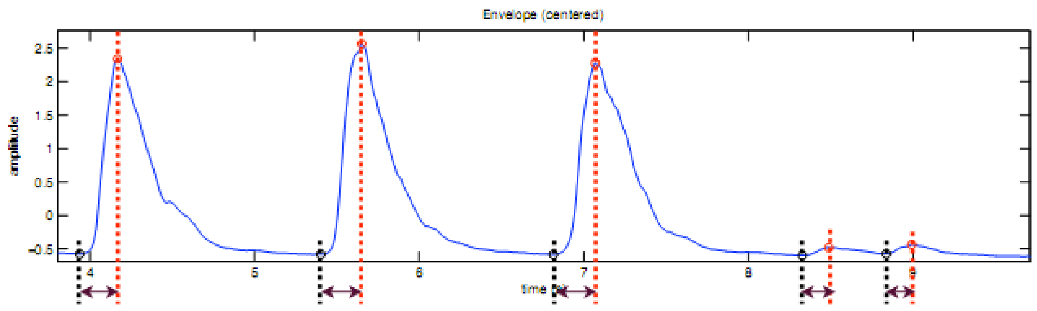


Figure 9: Attack time example through temporal duration (Lartillot, 2008)

Attack slope

This feature can be calculated as a ratio between the magnitude difference at the beginning and the ending of the attack period, and the corresponding time difference (Figure 10) (Lartillot, 2008):

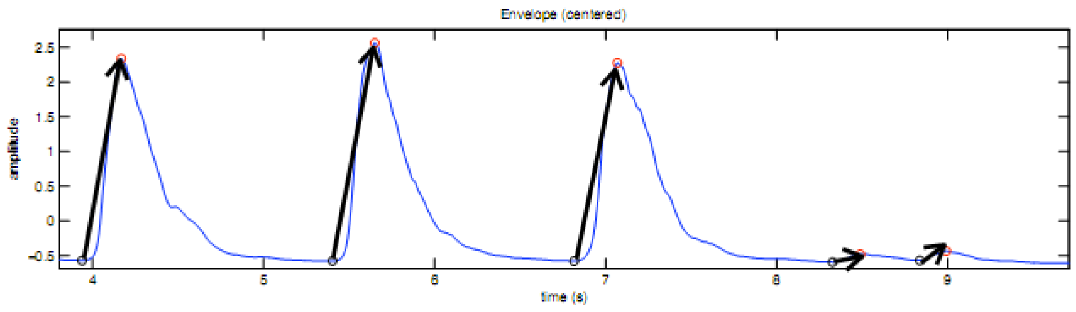


Figure 10: Example of attack slope, given by arrows (Lartillot, 2008)

Spectral roll off

One way to estimate the amount of high frequency in the signal consists in finding the frequency such that a certain fraction of the total energy is contained below that frequency. This ratio is typically fixed by default to 0.85. An example of this is given in Figure 11 (Lartillot, 2008):

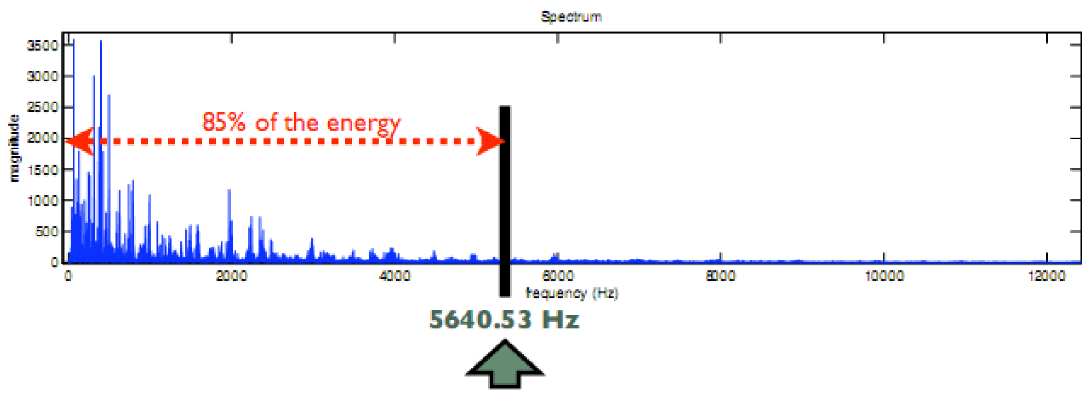


Figure 11: Example of spectral roll off point (Lartillot, 2008)

Its formal mathematical formula is defined as (Tzanetakis, 2002):

High frequency energy (brightness)

This feature is achieved by fixing this time the cut-off frequency, and measuring the amount of energy above that frequency. An example (1500 Hz) is shown in Figure 12:

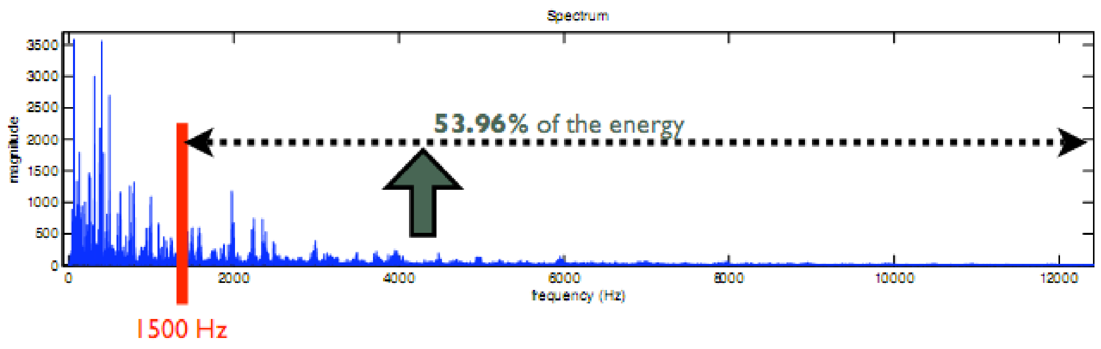


Figure 12: Example of high frequency energy (Lartillot, 2008)

Mel Frequency Cepstral Coefficients

MFCC returns a description of the spectral shape of the sound. The computation of the cepstral follows the scheme present in Figure 13 (Lartillot, 2008):

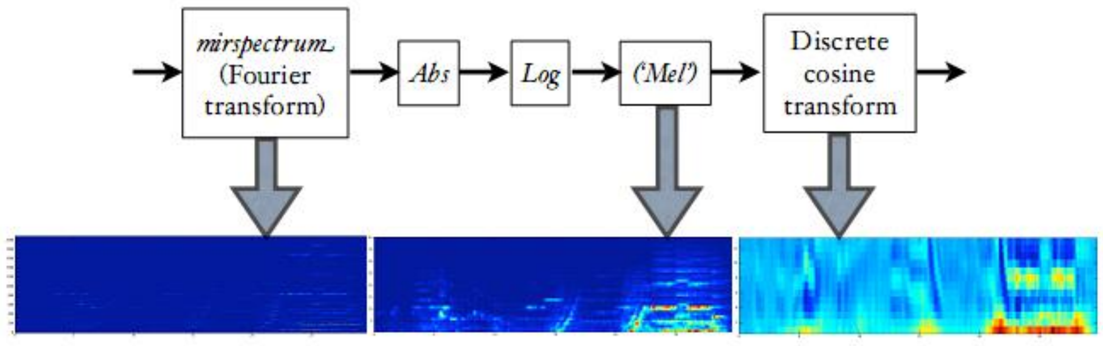


Figure 13: Phases of the MFCC computation (Lartillot, 2008)

MFCC are perceptually motivated features that are also based on the Short-time Fourier Transform (STFT). After taking the log-amplitude of the magnitude spectrum, the FFT bins are grouped and smoothed according to the perceptually motivated Mel-frequency scaling. Finally, in order to decorrelate the resulting feature vectors, a Discrete Cosine Transform (DCT) is performed. Although typically 13 coefficients are used for speech representation, it was found that the first five coefficients are adequate for music representation (Tzanetakis, 2002).

Linear Prediction Reflection Coefficients

Linear Prediction Reflection coefficients are used in speech research as an estimate of the speech vocal tract filter (Tzanetakis, 2002). They are also usually used in musical signals.

Sensory dissonance

Also known as roughness, this feature is related to the beating phenomenon whenever a pair of sinusoids is close in frequency. It can be estimated on the frequency ratio of each pair of sinusoids represented in Figure 14 (Lartillot, 2008):

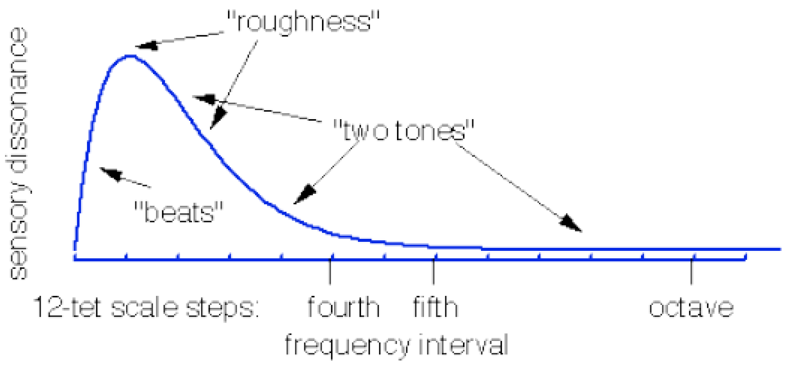


Figure 14: A representation of roughness (Lartillot, 2008)

Spectral centroid

The spectral centroid is defined as the center of gravity of the magnitude spectrum of the STFT:

$$= \frac{\quad}{\quad}$$

$M_t[n]$ is the magnitude of the Fourier transform at frame *t* and frequency bin *n*. The centroid is a measure of spectral shape and higher centroid values are related to "brighter" textures with more high frequencies. The spectral centroid has been shown by user experiments to be an important perceptual attribute in the characterization of musical instrument timbre (Tzanetakis, 2002).

Inharmonicity

This feature measures the amount of partials that are not multiples of the fundamental frequency *fo* (Figure 15).

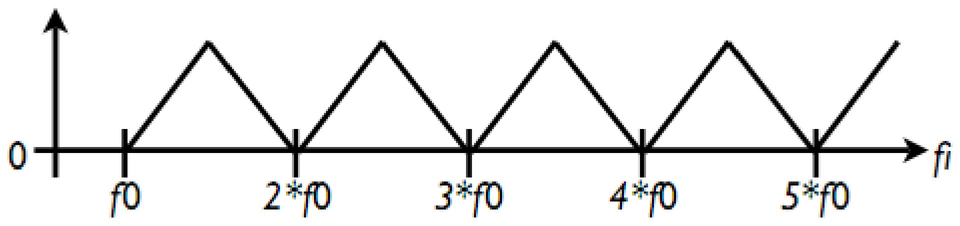


Figure 15: Graphical view for a given fundamental frequency *f0* and its multiples (Lartillot, 2008)

Spectral flux

It is defined as the squared difference between the normalized magnitudes of successive spectral distributions:

N$_t$*[n]*, *N*$_t$-1 *[n]* stand for the normalized magnitude of the Fourier transform at the current frame *t*, and the previous frame *t*-1, respectively. This feature is a measure of the amount of local spectral change. It has also been shown by user experiments to be an important perceptual attribute in the characterization of musical instrument timbre (Tzanetakis, 2002).

Spectral peaks variability (irregularity)

The irregularity measures the amount of local spectral change. It corresponds to the standard deviation of time-averaged harmonic amplitudes from a spectral envelope, and it can be described by the formula:

, where stands for and stands for (Paiva, 2006).

Strongest frequency via spectral centroid

An estimate of the strongest frequency component of a signal, found via the spectral centroid (McKay, 2005).

Zero-crossing rate

Indicates the number of times the waveform changed sign in a window (the number of times the signal crosses the X-axis), it can be used as an indication of frequency as well as noisiness (McKay, 2005). Figure 16 exemplifies this feature (Lartillot, 2008):

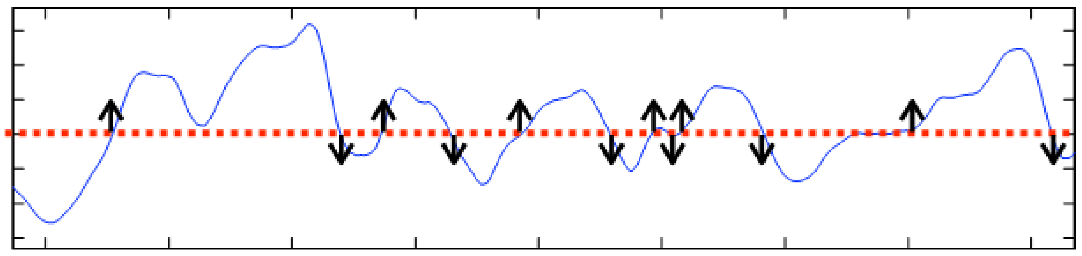


Figure 16: Zero crossing rate example (Lartillot, 2008)

Zero-crossing derivative

This feature can be defined as the absolute value of the window to window change in zero crossings. It can also be considered an indication of change of frequency as well as noisiness (McKay, 2005).

2.3.5. Tonality

Tonal centroid

The Tonal Centroid is a six-dimensional feature vector based on the Harmonic Network or *Tonnetz*, which is a planar representation of pitch relations where pitch classes having close harmonic relations such as fifths, major/minor thirds have smaller Euclidean distances on the plane, represented in Figure 17 (Lee & Slaney, 2007).

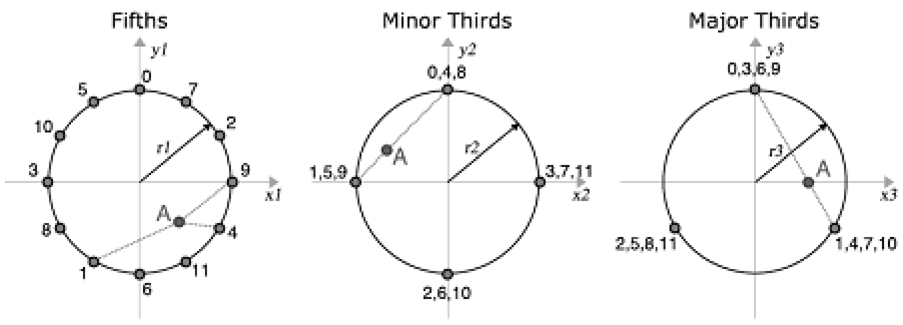


Figure 17: Visualization of the 6-D Tonal Space as three circles: fifths, minor thirds, and major thirds (from left to right) (Lee & Slaney, 2007)

It can be seen in Figure 17 that numbers on the circles correspond to pitch classes and represent nearest neighbors in each circle. Tonal Centroid for A major triad (pitch class 9, 1, and 4) is shown at point A (Lee & Slaney, 2007).

Harmonic change detection function

The Harmonic Change Detection function is the flux of the tonal centroid (Lartillot, 2008).

Key (tonal center positions)

The key feature gives a broad estimation of tonal center positions and their respective clarity, as seen in Figure 18 (Lartillot, 2008):

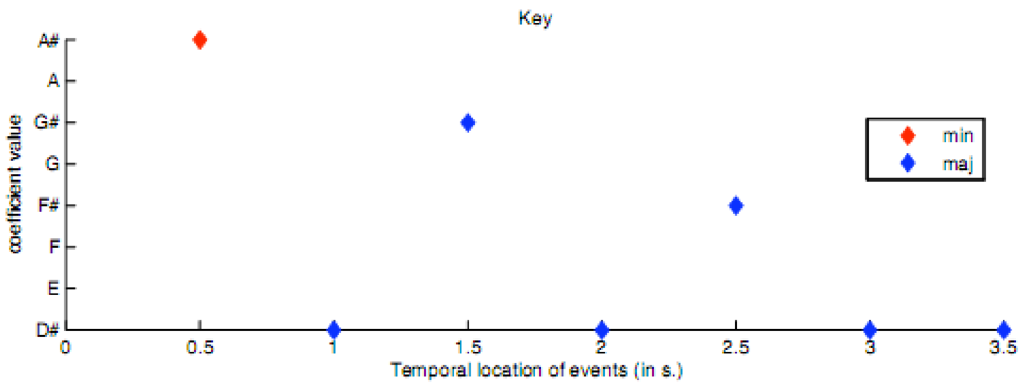


Figure 18: Key graphic example (Lartillot, 2008)

Modality

This feature returns the mode of a key (major or minor, for instance).

2.3.6. Musical content features

Musical content features are a set of both rhythmic and pitch content features introduced by George Tzanetakis (Tzanetakis, 2002). This set (and each of the 2 subsets) is based in features extracted previously.

Rhythmic content features

This subset is based on the BH (Beat Histogram) of a song:

- **A0, A1:** relative amplitude (divided by the sum of amplitudes) of the first, and second histogram peak;
- **RA:** ratio of the amplitude of the second peak divided by the amplitude of the first peak;
- **P1, P2:** Period of the first, second peak in BPM;
- **SUM:** overall sum of the histogram (indication of beat strength);

Pitch content features

The following features are computed from the Unfolded Histogram (UPH) and Folded Histogram (FPH) in order to represent pitch content:

- **FA0:** Amplitude of maximum peak of the folded histogram. This corresponds to the most dominant pitch class of the song. For tonal music this peak will typically correspond to the tonic or dominant chord. This peak will be higher for songs that do not have many harmonic changes;
- **UP0:** Period of the maximum peak of the unfolded histogram. This corresponds to the octave range of the dominant musical pitch of the song;
- **FP0:** Period of the maximum peak of the folded histogram. This corresponds to the main pitch class of the song;
- **IPO1:** Pitch interval between the two most prominent peaks of the folded histogram. This corresponds to the main tonal interval relation. For pieces with simple harmonic structure this feature will have value 1 or -1 corresponding to fifth or fourth interval (tonic-dominant);
- **SUM:** The overall sum of the histogram. This is feature is a measure of the strength of the pitch detection;

2.3.7. Statistical features

Finally, it is important to underline that is possible to extract statistical information (typically first and second order statistics: e. g. mean and standard deviation) from almost every single feature, as well as higher-order statistics (like skewness, kurtosis, etc.).

This is the main reason for the higher number of features presented by jAudio compared to other frameworks like MARSYAS (as it will be discussed later on this document, sections 2.6.1. jAudio and 2.6.4. Frameworks comparison), since many of them are statistical features obtained from basic features (again, mean and standard deviation, for instance).

2.4. Support Vector Machines

This classifier uses a support vector algorithm that simply looks for the largest margin (distance) from the separating hyperplane to avoid overfitting as maximum as possible. The support vectors play a key role as the critical elements of the training set. If other training points are changed (or removed) and the training repeated, the same separating hyperplane would be found. All these concepts are visible in Figure 19: SVM example with optimal separating hyperplane Figure 19:

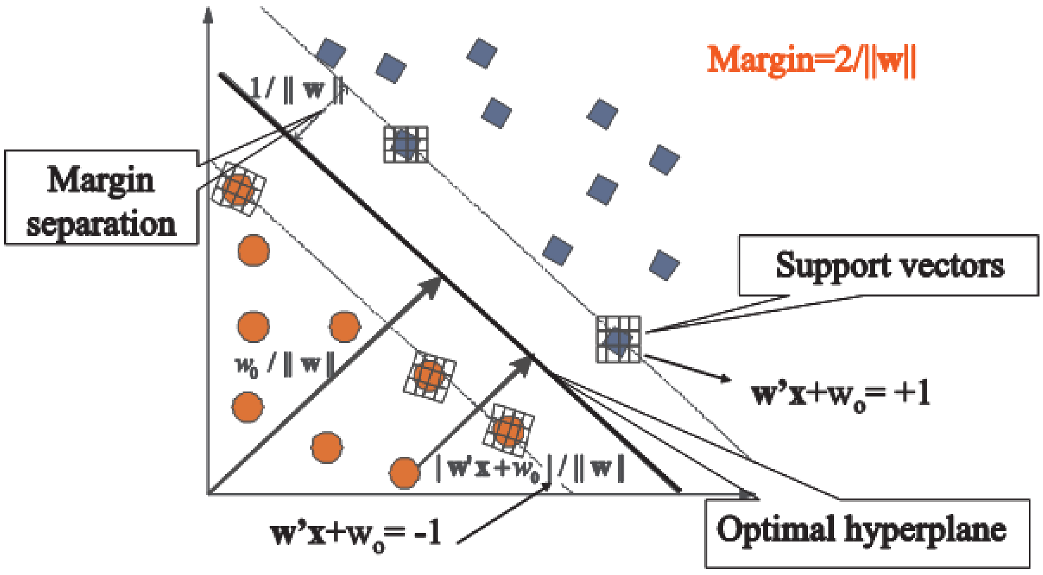


Figure 19: SVM example with optimal separating hyperplane (Ribeiro, 2009)

The separating hyperplane can be linear or non-linear, according to the problem to be solved. The SVM classifier is considered a good solution in what concerns to classification performance achieved / execution time rate, also due to its sophisticated kernel functions and possibility of multiclass classification (Ribeiro, 2009).

2.5. Euclidean distance in Thayer's model

This distance metric was proposed in (Yang, Lin, Su, & Chen, 2006) and basically considers that the similarity between two songs can be reached by the Euclidean distance of the songs (points represented as pair of coordinates, x and y) in the Thayer's model, arousal-valence plane.

The Euclidean distance (also known as the Pythagorean metric) is the most common distance metric used, since it is the "ordinary" distance between two points that one would measure with a ruler.

We'll now remember the formula that gives us the Euclidean distance between two points. The Euclidean distance between point *p* and point *q* is the length of the line segment . In Cartesian coordinates, if *p* = (p_1, p_2 , ..., p_n) and *q* = (q_1, q_2, ..., q_n) are two points in Euclidean *n*-space, then the distance from *p* to *q* is given by (Euclidean distance):

2.6. Existent frameworks and platforms

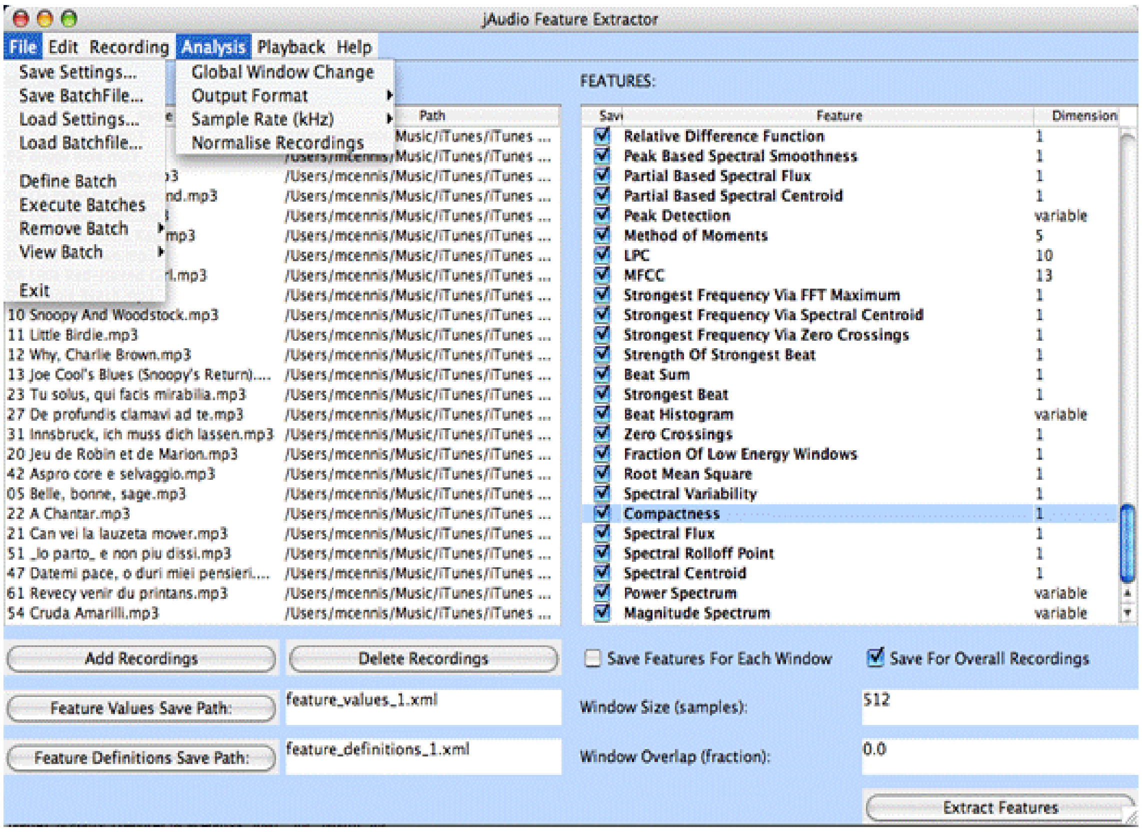
The most important and known platforms / frameworks available do perform feature extraction are now presented with more detail (jAudio, MARSYAS and MIRtoolbox).

2.6.1. jAudio

jAudio is a open source feature extraction system that can be used both with a GUI or in command line mode or as a library. jAudio uses a modular plugin interface that avoids core code modification or recompilation when new features are added.

One advantage of jAudio is automated "metafeature" extraction. Metafeatures are template-derived features that can be extracted from one or more other features. Examples of metafeatures implemented in jAudio include Running Mean, Running Standard Deviation and Derivative.

The software includes a wide set of features (28), which can be used with metafeatures and aggregators to expand this number, although a great part of this features are statistical results calculated from other basic features (mean, standard deviation, etc.). However, as mentioned before, this framework (as it is written in Java) is slower and computationally heavy. A screenshot of the application is now presented in Figure 20 (jAudio):



	Path
	Music/iTunes/iTunes ...
	/Users/mcennis/Music/iTunes/iTunes ...
3	/Users/mcennis/Music/iTunes/iTunes ...
nd.mp3	/Users/mcennis/Music/iTunes/iTunes ...
	/Users/mcennis/Music/iTunes/iTunes ...
mp3	/Users/mcennis/Music/iTunes/iTunes ...
	/Users/mcennis/Music/iTunes/iTunes ...
l.mp3	/Users/mcennis/Music/iTunes/iTunes ...
	/Users/mcennis/Music/iTunes/iTunes ...
10 Snoopy And Woodstock.mp3	/Users/mcennis/Music/iTunes/iTunes ...
11 Little Birdie.mp3	/Users/mcennis/Music/iTunes/iTunes ...
12 Why, Charlie Brown.mp3	/Users/mcennis/Music/iTunes/iTunes ...
13 Joe Cool's Blues (Snoopy's Return)....	/Users/mcennis/Music/iTunes/iTunes ...
23 Tu solus, qui facis mirabilia.mp3	/Users/mcennis/Music/iTunes/iTunes ...
27 De profundis clamavi ad te.mp3	/Users/mcennis/Music/iTunes/iTunes ...
31 Innsbruck, ich muss dich lassen.mp3	/Users/mcennis/Music/iTunes/iTunes ...
20 Jeu de Robin et de Marion.mp3	/Users/mcennis/Music/iTunes/iTunes ...
42 Aspro core e selvaggio.mp3	/Users/mcennis/Music/iTunes/iTunes ...
05 Belle, bonne, sage.mp3	/Users/mcennis/Music/iTunes/iTunes ...
22 A Chantar.mp3	/Users/mcennis/Music/iTunes/iTunes ...
21 Can vei la lauzeta mover.mp3	/Users/mcennis/Music/iTunes/iTunes ...
51 _Io parto_ e non piu dissi.mp3	/Users/mcennis/Music/iTunes/iTunes ...
47 Datemi pace, o duri miei pensieri....	/Users/mcennis/Music/iTunes/iTunes ...
61 Revecy venir du printans.mp3	/Users/mcennis/Music/iTunes/iTunes ...
54 Cruda Amarilli.mp3	/Users/mcennis/Music/iTunes/iTunes ...

Save	Feature	Dimension
✓	Relative Difference Function	1
✓	Peak Based Spectral Smoothness	1
✓	Partial Based Spectral Flux	1
✓	Partial Based Spectral Centroid	1
✓	Peak Detection	variable
✓	Method of Moments	5
✓	LPC	10
✓	MFCC	13
✓	Strongest Frequency Via FFT Maximum	1
✓	Strongest Frequency Via Spectral Centroid	1
✓	Strongest Frequency Via Zero Crossings	1
✓	Strength Of Strongest Beat	1
✓	Beat Sum	1
✓	Strongest Beat	1
✓	Beat Histogram	variable
✓	Zero Crossings	1
✓	Fraction Of Low Energy Windows	1
✓	Root Mean Square	1
✓	Spectral Variability	1
✓	Compactness	1
✓	Spectral Flux	1
✓	Spectral Rolloff Point	1
✓	Spectral Centroid	1
✓	Power Spectrum	variable
✓	Magnitude Spectrum	variable

☐ Save Features For Each Window ✓ Save For Overall Recordings

Feature Values Save Path: feature_values_1.xml

Feature Definitions Save Path: feature_definitions_1.xml

Window Size (samples): 512

Window Overlap (fraction): 0.0

Figure 20: jAudio screenshot, modified artificially to show two menus simultaneously (jAudio)

2.6.2. MARSYAS

MARSYAS is another open source software framework for audio processing with specific emphasis on MIR applications, being one of the first frameworks built in the area. It has been designed and written by George Tzanetakis, one the most well known and experienced names in the field, with help from students and researchers from around the world. MARSYAS has been used for a variety of projects in both academia and industry (Marsyas).

The basic goal of this framework is to provide a general, extensible and flexible architecture that allows easy experimentation with algorithms and provides fast performance that is useful in developing real time audio analysis and synthesis tools (Overview).

It is written in C++ (with several algorithms) and offers excellent classification times (Audio Music Mood Classification Results), as mentioned earlier. Although it does not come with a GUI it has base support for integration with Qt. These two advantages were crucial for choosing MARSYAS as the framework to be used in this project.

The main disadvantage of this framework is the fact that it apparently has less implemented features than the other two. Although, this flaw can be surpassed with the implementation of the lacking features, if necessary.

Finally, it is important to refer that a variety of existing building blocks that form the basis of most published algorithms in Computer Audition are already available as part of the framework and extending the framework with new components/building blocks is straightforward (Overview).

2.6.3. MIRtoolbox

MIRtoolbox offers an integrated set of functions written in MATLAB, dedicated to the extraction from audio files of musical features, with the objective of offering an overview of computational approaches in the MIR area. The design is based on a modular framework: the different algorithms are decomposed into stages, formalized using a minimal set of elementary mechanisms. These building blocks form the basic vocabulary of the toolbox, which can then be freely articulated in new original ways. These elementary mechanisms integrate all the different variants proposed by alternative approaches that users can select and parameterize. This synthetic digest of feature extraction tools enables a capitalization of the originality offered by all the alternative strategies. Additionally to the basic computational processes, the toolbox also includes higher-level musical feature extraction tools, whose alternative strategies, and their multiple combinations, can be selected by the user (Lartillot, Toiviainen, & Eerola, Department of Music: MIRtoolbox).

Briefly, it can be said that the main advantages of this toolbox are the integration with MATLAB (which can mean vast documentation, community help and in a sort of way ease of use) and the number of features implemented. However, the MATLAB dependency can also be a disadvantage, mainly because despite the toolbox is open source, MATLAB is not, and so dependency to the application is needed. Also, like jAudio, MATLAB is heavily based and implemented in the Java language, which makes it computationally heavy and slow when it comes to extract features / classify musical pieces.

2.6.4. Frameworks comparison

The three mentioned frameworks are now compared in what concerns to features, classifiers and distance metrics.

Features

Is now time to make a framework comparison (jAudio, MARSYAS and MIR toolbox and some papers / documentation) in what concerns to implemented features (Table 1).

Feature	Feature class	jAudio	MARSYAS	MIR toolbox	Others
Fraction Of Low Energy Frames	intensity	√			
Less-Than-Average Energy	intensity	√		√	
Root Mean Square Derivative	intensity	√			
Root Mean Square	intensity	√		√	
Root Mean Square Variability	intensity	√			
Pitch (Fo)	pitch		√	√	√
Strongest Frequency Via FFT Maximum	pitch	√			
Beat Sum	rhythm	√	√		
Rhythmic Fluctuation	rhythm			√	√
Strength Of Strongest Beat	rhythm	√		√	
Tempo	rhythm	√		√	√
Attack Slope	timbre			√	
Attack Time	timbre			√	
High Frequency Energy (Brightness)	timbre			√	√
Inharmonicity	timbre			√	
Linear Prediction Reflection Coefficients	timbre	√	[15]		
Mel-Frequency Cepstral Coefficients	timbre	√	√	√	√
Sensory Dissonance	timbre			√	√
Spectral Centroid	timbre	√	√	√	√
Spectral Flux	timbre	√	√	√	√
Spectral Peaks Variability (Irregularity)	timbre	√		√	√
Spectral Roll Off	timbre	√	√	√	√
Strongest Frequency Via Spectral Centroid	timbre	√			
Zero Crossing Rate	timbre	√	√	√	√
Zero Crossing Derivative	timbre	√	√		
Harmonic Change Detection Function	tonality			√	
Key (Tonal Center Positions)	tonality			√	
Modality	tonality			√	
Tonal Centroid	tonality			√	
Musical Content Features	-		√		

Table 1: List of features available in each framework (and papers)

Classification methods

Classfier	jAudio	MARSYAS	MIR toolbox
k-Nearest Neighbor		√	√
Gaussian Mixture Model		√	√
Support Vector Machines		√	

Table 2: Classification methods overview

[15] Linear Prediction Reflection Coefficients are estimated based on Linear Prediction Cepstral Coefficients, which are supported by MARSYAS

Distance metrics

Metric	jAudio	MARSYAS	MIR toolbox	Others
Euclidean distance		√	√	
Manhattan distance			√	
Euclidean distance in Thayer's model[16]				√
Membership feature vector distance[17]				√

Table 3: Distance metrics overview

2.7 Test collection

The musical dataset that is to be used in the application to be developed was kindly provided by Yi-Hsuan Yang, one of the authors of (Yang, Lin, Su, & Chen, 2006), after a personal request. The dataset is the same used in the mentioned paper, and consists of 194 musical pieces from popular japanese, chinese and western albums. Also, two MATLAB .m files (one with the arousal values and the other with valence values for each song in the dataset) were provided.

The musical pieces have common properties: all of the 194 samples have 25 seconds length, and were converted to a uniform format (22,050 Hz, 16 bits, and mono channel PCM WAV) and normalized to the same volume level. The 25 second segment was manually chosen to be representative of the dominant mood in the song (mostly the chorus part) (Yang, Lin, Su, & Chen, 2006).

It is important to notice that this is a provisory dataset, which can be modified and/or expanded in the future if proved necessary.

2.8. Followed approach

In this project, several approaches were studied and considered, some of them with similar aspects to this application main goal of this application: an automatic playlist generation system, based on mood analysis of a musical database.

[16] This distance metric was introduced in a paper but can be easily implemented
[17] To our knowledge, a new distance metric introduced in this work, also easy to develop

Options were made in order to achieve this, some by choice, others due to limitations, notably time constraints. As planned, Thayer's mood model, based on a continuous arousal-valence plan, was used.

Also as planned, feature extraction stage was developed using the MARSYAS framework. Although complex, poorly documented and sometimes unstable from version to version, this framework proved to be fast and powerful.

The approach developed in this thesis outputs a "continuous" classification, since it maps each song with a coordinate system (x, y), with values ranging from -1 to 1 in each axis, according to Thayer's model, as pointed out earlier. This implied training a regression model, since it's not a discrete classifier. To achieve this (and since MARSYAS only supports discrete classifiers), an additional classification library was used - libSVM[18], in conjunction with MARSYAS.

This way, after a train phase, similarity between two songs is calculated based on the Euclidean distance between the songs coordinates (points) in the Thayer's graphic model, as specified earlier in this document. This is done during the test phase and then playlist generation is achieved computing the closer songs to the one used as query. Both annotations and predicted values for arousal and valence coordinates in the test set are compared to evaluate the accuracy of the generated playlists.

[18] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

The implemented application consists of three main parts: the client, backoffice and server ones. The first two are more stable, since they have been being implemented for some time now, while the backoffice is in a alpha version, at the moment only allowing the user to login, logout and send a new song to the server (in order to be added to the database). The client application allows the user to see the songs distributed in the Thayer's model, generate a playlist based on a path designed by the user in the graph (and export it to a M3U file) and the see each song details, which includes a waveform visualization and highlighted mood changes in the song using different colors for each one of the Thayer's model quadrants. Finally, the server coordinates the communication with both the client and backoffice applications, performs song classification and interacts with the database and the musical dataset.

3. Implementation

In this section we present the main decisions taken throughout the development of this project. In what concerns to MARSYAS, this covers aspects related to feature extraction, data normalization, classification, playlist generation and evaluation. It also includes the back office part of the application, developed in Qt.

3.1. Playlists

As mentioned earlier, all the tasks related to playlist generation were developed on MARSYAS, the MIR framework chosen to achieve this goal. Based on the work done by my colleague Renato Panda in his thesis (mood tracking), some parts were improved, while others were developed from scratch. The following sections provide more detail about this.

3.1.1. Feature extraction

To achieve the main purpose of the developed application, the first step was clearly to extract the features of the all the songs on the dataset (the 194 songs samples annotated by Yang). All the features sets provided by MARSYAS were used (13 features sets), without selection, which comprises 454 features in total for each song to be extracted. This number includes statistical features derived from the core ones (means, standard deviations, etc), among others:

- Tempo
- Stereo Panning Spectrum Features
- Mel Frequency Cepstral Coefficients
- Chroma
- Spectral Flatness Measure
- Spectral Crest Factor
- Spectral Centroid
- Spectral Rolloff
- Spectral Flux
- Line Spectral Pair
- Linear Prediction Cepstral Coefficients
- Zero Crossings
- Beat

Since the objective was to classify each aforementioned sample with a single AV (arousal/valence) pair (which would represent the value for the whole song segment), and consequently a unique value for each feature depicting the entire song, a single feature of values was used (MARSYAS provides the possibility of using other kind of networks, which can extract many values for a given song feature – for instance, for mood tracking purposes, among others).

3.1.2. Data normalization

Between feature extraction and classification, classification data (*i.e.* feature extraction values) was normalized, to ensure that all values ranged between the same boundaries, preventing the classification results to become corrupted.

To achieve this, the following formula was used:

, where is the normalized value, *l* and *u* are respectively the upper and lower limits between which the features values will be scaled. In this case, the chosen feature normalization interval was [0, 1] (*u*=0 and *l*=1).

3.1.3. Classification

To properly classify the songs and estimate distances between them, both arousal and valence values needed to be known simultaneously. With this in mind, the initial method developed by my colleague Renato Panda was improved – so that the mentioned pair values could be predicted at the same time. Once again (as in Renato's thesis), the SVM classifier was used for this, through the libSVM library, on the Yang dataset (as referred earlier in this report, 194 samples of songs, annotated with arousal and valence values).

It was chosen to use *K*-fold cross-validation, with *K*=4, which means that 3 folds were used for the training phase (75% of the dataset), while the last fold was used for testing (25% of the dataset). This means that each fold would have 48 or 49 songs (considering the 194 that comprised the whole dataset). As this cross-validation method implies, all the 4 folds were rotated, to ensure that all of them were used for both training and testing purposes.

The aforementioned process was repeated 50 times, which means that 200 folds (4 x 50) were generated. It is also important to underline that all the folds were randomly generated.

Some statistical results can be measured by using the average predicted arousal and valence results from the original annotations, which are now presented in the next four sections.

Sum-Square Error (SSE)

Measures the total deviation of the predicted values from the original annotations, and is calculated by the following formula:

, where is the annotation and is the predicted value.

Total Sum of Squares (SST)

Measures the deviation of each annotation to the mean value of the annotations, and is given by:

, where is the annotation and the average of all annotation values.

Root Mean Square Error (RMSE)

Estimates the standard deviation of the predicted values, whose formula is:

R^2

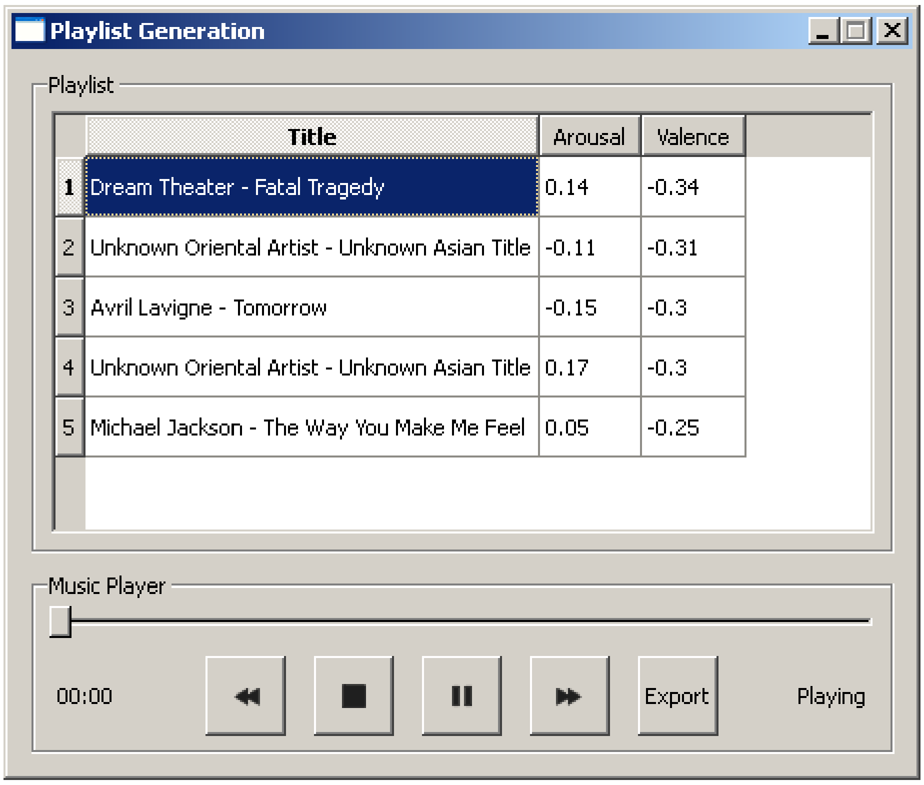
Is a statistic value used to measure how successful the trained model is and how it fits our training and test data. When results are near 1 it means that the model fits the data perfectly. It is given by:

—

Overall AV classification accuracy is provided by this statistic, so that it can be compared with Yang *et al.* results.

3.1.4. Playlist generation

In this part, after all the arousal and valence values had been predicted (for all the songs in the test fold), for each one of these, the Euclidean distances (based on the arousal / valence values) to all the remaining test set songs were computed, stored and ordered. This way, for each song in the test fold, both a top annotation and top prediction list was generated, both ordered by ascending distance (which means closest songs first). Based on this, some metrics were estimated, which are more accurately discussed in the next section (3.1.5. Playlist evaluation). Figure 21 shows the playlist generation window in the client GUI of the application:



Playlist Generation

Playlist

	Title	Arousal	Valence
1	Dream Theater - Fatal Tragedy	0.14	-0.34
2	Unknown Oriental Artist - Unknown Asian Title	-0.11	-0.31
3	Avril Lavigne - Tomorrow	-0.15	-0.3
4	Unknown Oriental Artist - Unknown Asian Title	0.17	-0.3
5	Michael Jackson - The Way You Make Me Feel	0.05	-0.25

Music Player

00:00 Export Playing

Figure 21: Playlist generation window (client application)

3.1.5. Playlist evaluation

In each test fold, every song in it is used sequentially as a query, and since it is guaranteed that all four folds are used as test ones, we guarantee that in every repetition all the songs are used as queries.

To efficiently evaluate the quality of the generated playlists (closest predicted songs playlist – basically the first *n* elements of the top annotation list, *n* being the playlist size - in comparison with the closest annotated songs playlist – the first *n* elements of the top prediction list, *n* being again the playlist size), four metrics or measures were calculated (for each song in the test set), which are explained after Table 4.

Next table (Table 4) shows an example of the top 20 annotations and top 20 predictions (*i.e.* closest songs) for one of the songs present in one of the (randomly generated) test folds. In this case the test fold had size 49. The chosen song (query) is Dream Theater's "Fatal Tragedy":

#	Top annotations (song name)	Distance	Top predictions (song name)	Distance
1	2.18	0.0482597	1.36	0.0545035
2	2.06.Dream_Theater.The_Answer_Lies_Within	0.118806	2.26	0.0658998
3	2.05.Dream_Theater.Pull_Me_Under	0.228473	2.29	0.0662149
4	2.30	0.234094	2.16.Alanis_Morissette.Jagged_Little_Pill_Forgiven	0.0733243
5	2.16.Alanis_Morissette.Jagged_Little_Pill_Forgiven	0.236764	2.39	0.0736593
6	2.38	0.260768	1.10.Bon_Jovi.08_Save_the_world	0.084574
7	2.45	0.269966	1.21	0.0886902
8	3.11.Angra.01_Deus_Le_Volt	0.270113	2.30	0.108635
9	2.48.Avril.07_Tomorrow	0.292746	4.13.Bill_Evans.My_Foolish_Heart	0.126201
10	2.29	0.339988	3.10	0.126384
11	2.39	0.348528	3.13.Jack_Johnson.01_Better_Together	0.1306
12	2.13.alanis_morissette.so_called_chaos_rns	0.394462	2.45	0.136185
13	2.26	0.411565	1.20	0.136735
14	3.10	0.446781	2.38	0.140954
15	3.17	0.44764	1.39	0.145313
16	1.25.Aerosmith.Dont_Stop	0.499992	2.48.Avril.07_Tomorrow	0.146856
17	1.43	0.538179	4.14.Damage.Forever	0.164022
18	4.26	0.545388	1.17	0.170183
19	1.38	0.560014	2.05.Dream_Theater.Pull_Me_Under	0.172588
20	4.35	0.575847	4.35	0.188898

Table 4: Top 20 annotations and top predictions lists for song "Fatal Tragedy" by Dream Theater

First playlist song match

This metric verifies if the first song in the closest predicted songs playlist is the same in the closest annotated songs playlist. For this reason, for each song, this percentage value can only be 0% or 100%. Just for curiosity (since obviously it doesn't affect the results), the chosen playlist size for this metric was 20 songs.

For instance, in Table 4, this metric is 0%.

Percentage of playlist songs match (size 5)

In this metric, both the closest predicted songs playlist and the closest annotated songs playlist with size 5 were compared, to see how many songs were common to both of them (the order being irrelevant).

For instance, in Table 4, if we consider the first five songs in each top, this metric is 20%, since only one song matches the two tops.

Percentage of playlist songs match (size 10)

This metric is identical to the previous one, with the only difference being the playlist size (in this case, 10 songs).

For instance, in Table 4, if we consider the first ten songs in each top, this metric is 30%, since three songs match the two tops.

Percentage of playlist songs match (size 20)

Again, this metric is similar to the previous two, with the slight difference in playlist size (this time, 20 songs for comparison).

For instance, in Table 4, if we consider the whole table (first twenty songs in each top), this metric is 55%, since eleven songs match the two tops.

3.2. Database

A SQL database is used by server, which stores all the information about classification and the songs. The server does all the operations required on the database (e.g. querying, update and addition of data). The use of Qt SQL libraries provides support for different database management systems, also supporting different engines, from a simple SQLite file to MySQL, PostgreSQL, Oracle or Acess DB files or any other Open Data Base Connectivity (ODBC) protocol. Currently, the prototype supports SQLite, with preliminary support for MySQL.

The database was planned and designed in a general and expandable way. It supports the current needs but also allows different mood models, including different types (both categorical and dimensional), user accounts, lists of artists, albums, genres, features and classification profiles, saving different classification and tracking values for the same song, based on different combinations of features and classifiers, for instance. Figure 22: Entity-Relationship modelFigure 22 represents the current Entity-Relationship Model:

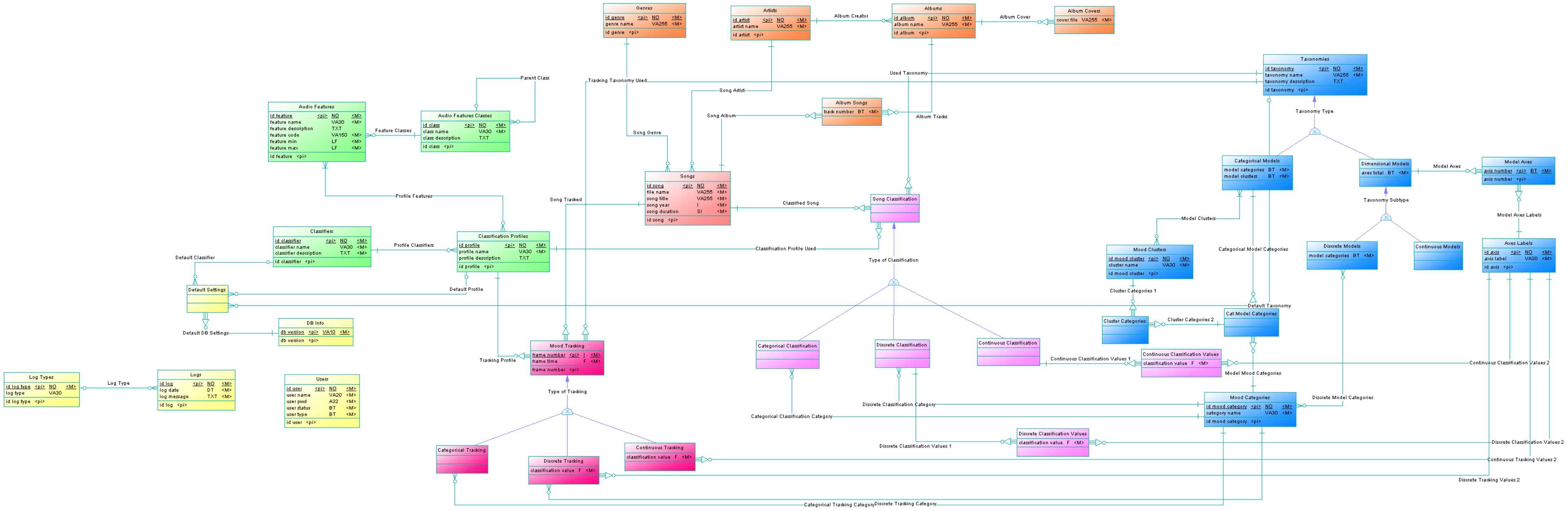


Figure 22: Entity-Relationship model

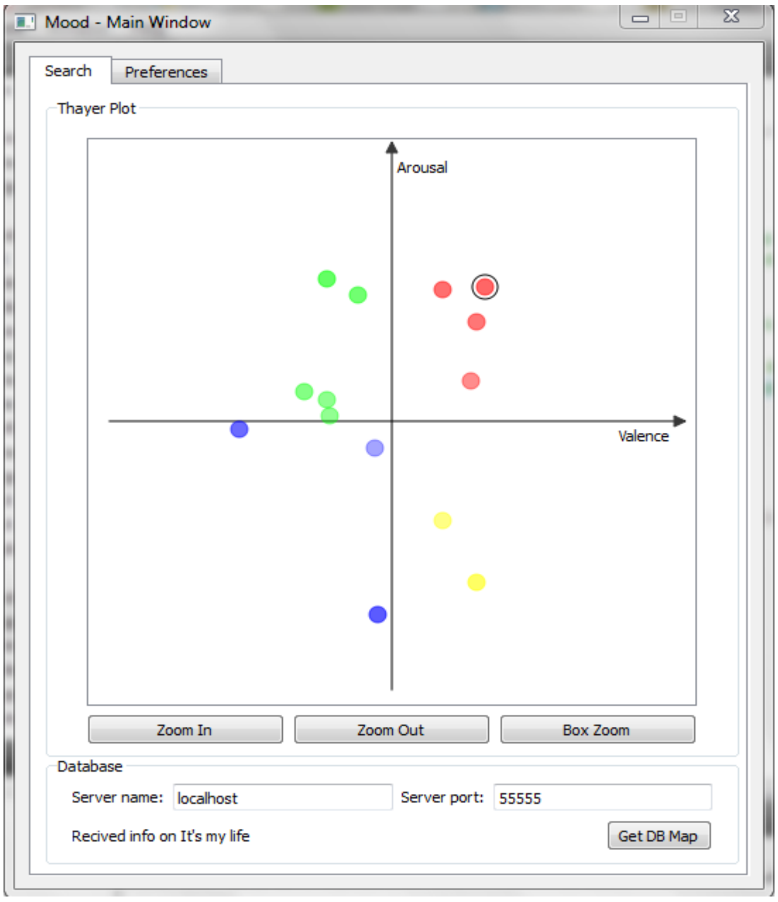
3.3. Client application

The client application was developed by Luís Cardoso, a Bsc student that also makes part of the team, with the requirements analysis (seeAppendix A) being developed by Professor Rui Pedro Paiva, Renato Panda and myself, using the Qt platform. Using Qt guarantees that the developed application is cross-platform and usable in any of the platforms supported by Qt.

Currently, the prototype supports both playlist generation (retrieving a playlist based on the draw of the desired path in the Thayer's model) and mood tracking features (showing the quadrant changes, highlighted in different colors in the sound wave graph), along with more basic features, such as playing a selected song, see its details or checking the distribution of the songs over the Thayer's model.

3.3.1. Main window

As it can be seen in Figure 23, after opening the application it is necessary to insert both the server address and the port number, in order to connect to establish the necessary connections. Once this is made, the user can request the visual map of all songs in the database ("Get DB Map" button), each point representing a song. The user can pass over each point with the mouse to check each song name. Additionally, the user can zoom in an out in the map, draw traces (a desired playlist path, for instance) and click on each song.



Database

Server name: localhost

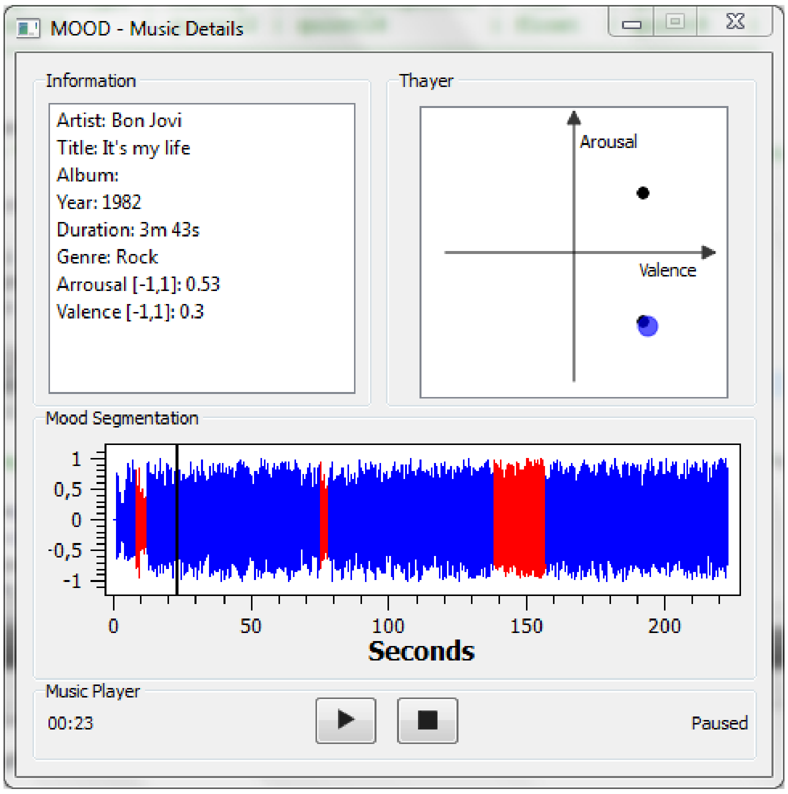
Server port: 55555

Recived info on It's my life

Figure 23: Client application main window

3.3.2. Song Details

After selecting one song in the database map, a request for its details is made to the server. The information is then received and displayed on the "Song details" dialog (Figure 24), which displays all the ID3 tag information data available for the specified song, as long as the correspondent soundwave graph and mood tracking data (based on Renato's work). The song can be played and a vertical black line marks the progress of it, with different colors representing different mood quadrants identified in the song.



Artist: Bon Jovi
Title: It's my life
Album:
Year: 1982
Duration: 3m 43s
Genre: Rock
Arrousal [-1,1]: 0.53
Valence [-1,1]: 0.3

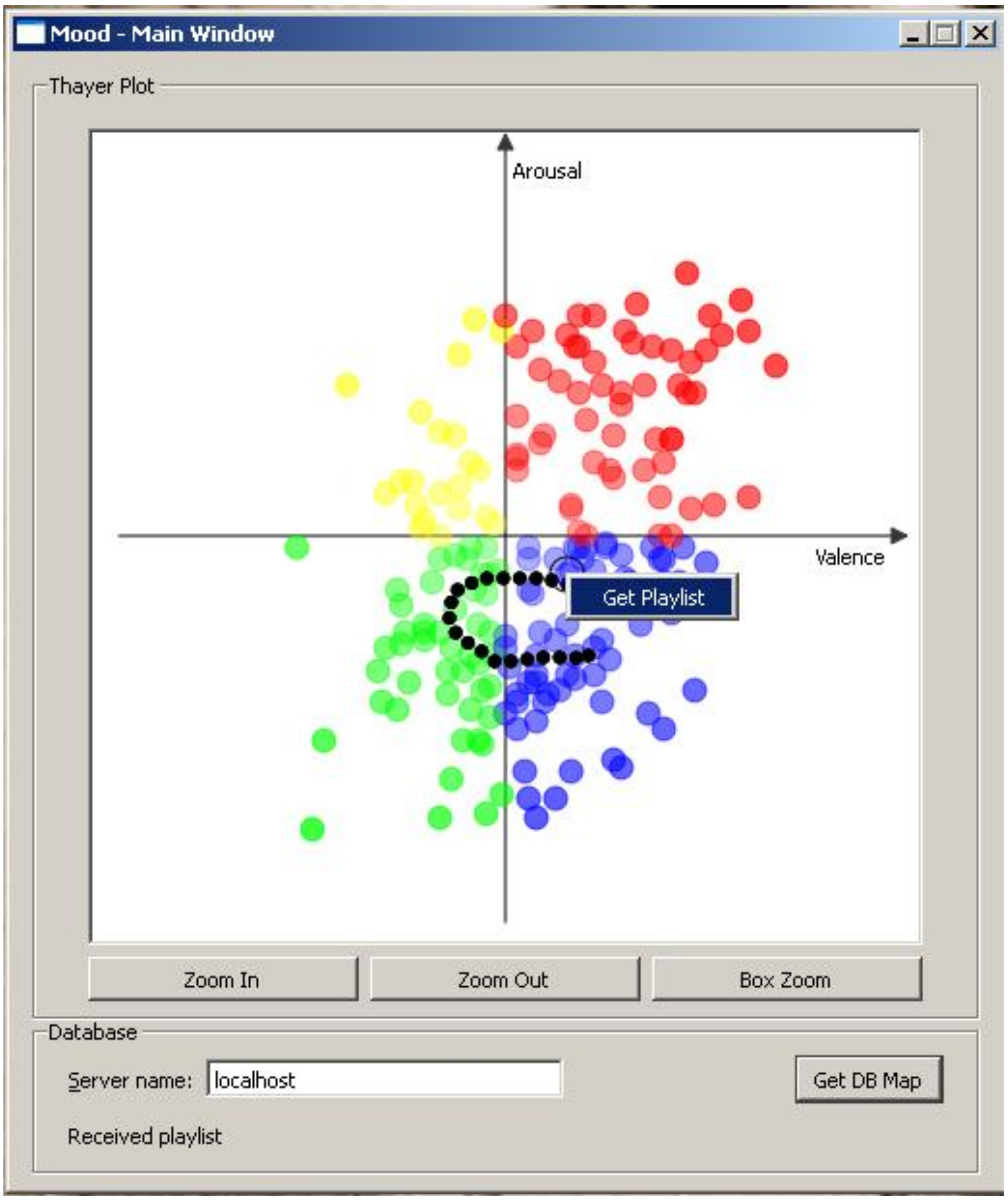
00:23

Paused

Figure 24: Client application song details dialog

3.3.3. Playlist details

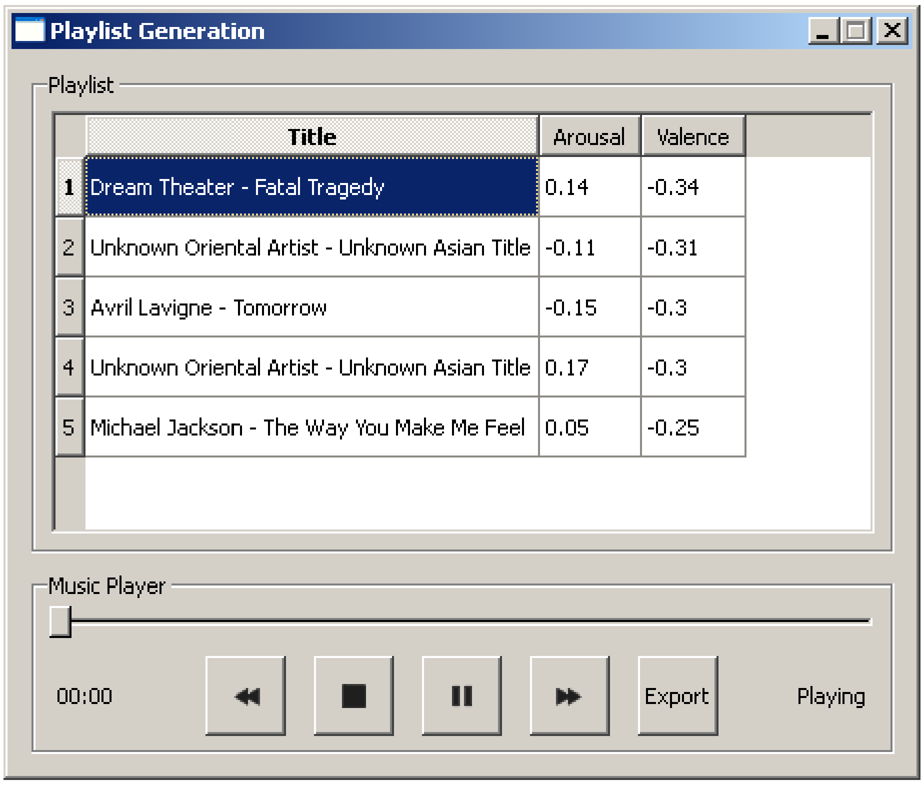
The playlist visualization purposes are satisfied after drawing the desired path in the main client window (Figure 25) and clicking with the mouse's right button in it ("Get Playlist" button). That will present the user with a new dialog, where the generated playlist information is displayed (Figure 26). Now, the user can export the playlist to the M3U format with a simple click on a button ("Export") or select any one of the songs present in the playlist, as shown in Figure 26:



Server name: localhost

Received playlist

Figure 25: Client application main window (with drawn desired playlist path)



	Title	Arousal	Valence
1	Dream Theater - Fatal Tragedy	0.14	-0.34
2	Unknown Oriental Artist - Unknown Asian Title	-0.11	-0.31
3	Avril Lavigne - Tomorrow	-0.15	-0.3
4	Unknown Oriental Artist - Unknown Asian Title	0.17	-0.3
5	Michael Jackson - The Way You Make Me Feel	0.05	-0.25

Figure 26: Client application playlist details dialog

3.4. Backoffice application

The backoffice application prototype provides administration means to the server. It works sending requests and receiving responses from the server (similar to the client application). Currently, it is an alpha version and so it only supports user login / logout, also allowing the administrator to send new songs to the server (after successful authentication), in order to be added to its database.

The communication is made through *QTcpSockets*, and different data blocks are assembled in order to make different requests to the server (login, logout, etc), which are made using an array of bytes (*QByteArray* class). The requests are then sent and the responses received using *QDataStream* objects (one for each purpose).

When the application starts, the login window is presented (Figure 27). After successfully filling the server address (if left empty, it will connect to the localhost address), username and password, a request is sent to the server with the user data. If the server response confirms that both the username and the password are valid, the login window is closed and the main window is presented (Figure 28), where the user can add a song to the database ("Choose file..." button) or logout the system (menu "File"->"Logout").

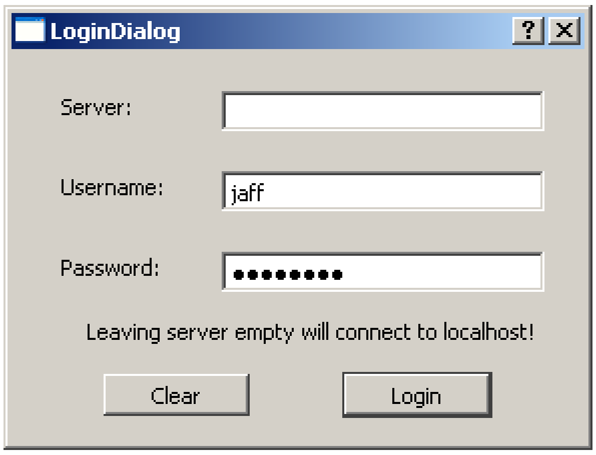


Figure 27: Backoffice application login dialog

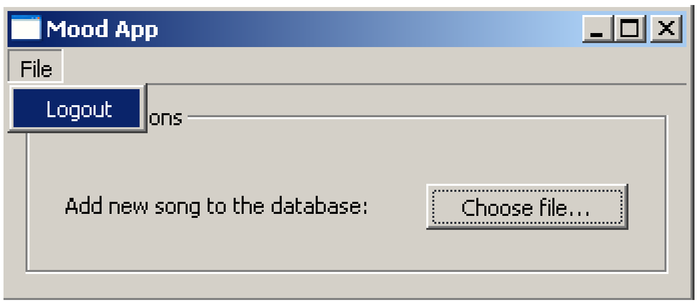


Figure 28: Backoffice application main window

3.5. Server application

This application was developed by my thesis colleague Renato Panda, and despite not having a GUI, it was also built using the Qt framework. The function of the server is to receive all the requests from both the client and backoffice applications and provide the correct responses to that requests. Also, it interacts with the database and processes audio files. The communication is made in a similar way as documented in the Backoffice application (section 3.4. Backoffice application), and threads are used to support multiple client connections simultaneously.

4. Experimental results

In this section, some information about the Yang dataset annotations are given, while in section 4.2. Global results the results achieved are presented and discussed.

4.1. Song annotations

As pointed out in previous sections of this report, the song dataset used was kindly provided by Yi-Hsuan Yang, one of the authors of (Yang, Lin, Su, & Chen, 2006), along with the arousal and valence annotations values for each one of the dataset songs. This dataset was the one use for training and testing purposes, and consisted of 194 songs from several genres and provenances, spanning different arousal and valence values, from all four Thayer's model quadrants.

However, as my colleague Renato Panda pointed out in his thesis, the annotations values revealed that they aren't 100% accurate and not as diverse as one would expect, which of course can negatively influence the final results. This will be addressed below.

4.1.1. Proximity to Thayer's model origin

After mapping the dataset annotations to the Thayer's model, it is obvious that the majority of the songs are close to the origin of both axes, which suggests that they don't denote very marked moods (if that was the case they would be much more distant from the axes origin, at least the majority). This can be seen both in Figure 29 and Table 5, below:

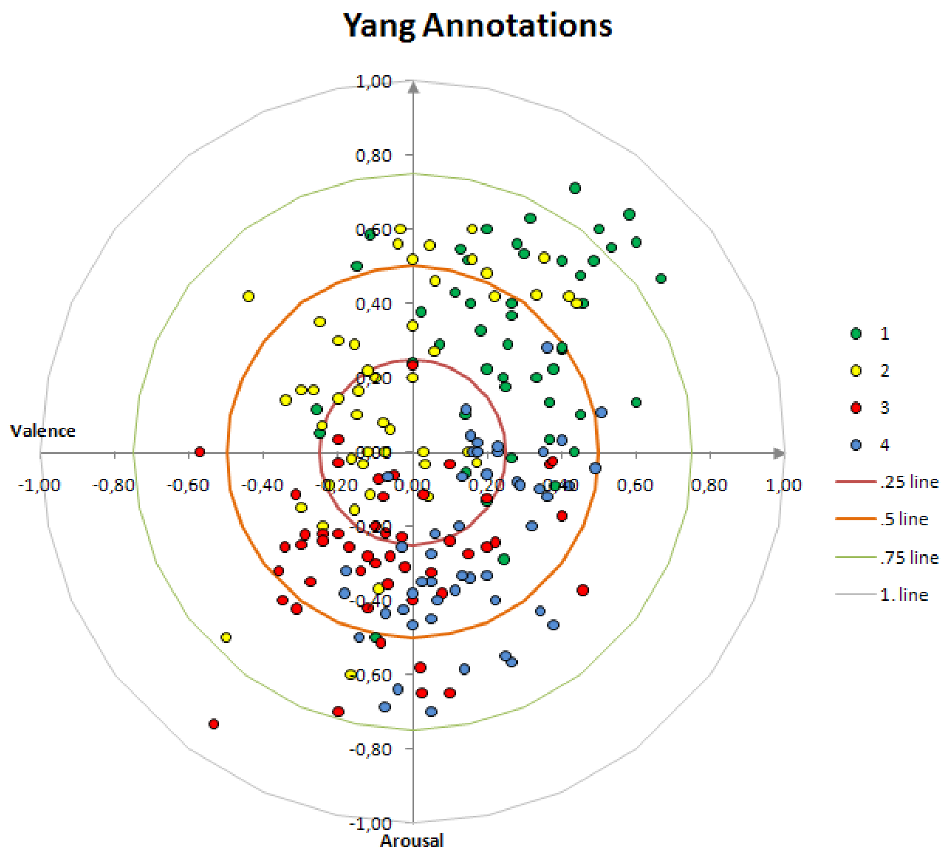


Figure 29: Yang annotations mapped into Thayer's model[19]

Distance from the origin	Number of songs	Percentage in the dataset	Sum
[0, 0.25]	47	24.23 %	47
]0.25, 0.5]	93	47.94 %	140
]0.5, 0.75]	47	24.23 %	187
]0.75, 1]	7	3.61 %	194

Table 5: Yang annotations distances to the model's origin[20]

As it can be seen in Table 5, nearly 25% of the songs annotations are within a distance of 0.25 of the center (red circumference in Figure 29), while nearly 75% of the dataset is at most at a distance of 0.5 from the graph's origin (orange circumference in Figure 29). This means that the majority of the songs are placed near the model's origin, which obviously leads to somewhat ambiguous moods.

[19] Image kindly provided by Renato Panda
[20] Information kindly provided by Renato Panda

4.1.2. Unbalanced song distribution

The Yang dataset aimed to achieve a balanced distribution of songs in the four quadrants that compose the Thayer's Model. Since the dataset has 194 songs, this would mean about 48 or 49 songs for each quadrant. After analyzing the annotations values, it is quite evident that this differ much from the initial quadrants (expressed in each song filename), which results in an unbalanced dataset.

Again, looking to Figure 29 (where each one of the four colors represents one of the four quadrants), it was expected to see the points with the same color (the initial annotation) together in the same quadrant. However, as it can be seen in Figure 29 and analyzed in Table 6, it is quite obvious that the songs are scattered all over the model's quadrants, which makes the dataset completely unbalanced. This is another issue that can also lead to distorted results.

Quadrant	Yang	Real	Real (%)	Matching	Matching (%)
1	48	54	27.84 %	36	75.00 %
2	48	22	11.34 %	17	35.42 %
3	49	51	26.29 %	16	32.65 %
4	49	49	25.26 %	9	18.37 %
Other	0	18	9.28 %	-	-
Total	194	194	100.0 %	78	40.21 %

Table 6: Songs per quadrant (Yang's vs. real annotations)[20]

In Table 6, the "Yang" column shows the number of songs in each quadrant, according to Yang's initial annotations, while the "Real" column shows how many songs were indeed in each quadrant, after the analysis of the annotated values (in this particular case, the "Other" row shows 18 songs where one of the values for arousal or valence was equal to zero, therefore not belonging to any particular quadrant). The "Matching" column shows how many songs were indeed correct (matching in both Yang's initial annotations and the annotated values).

In a quick glance, it can be seen that only nearly 40 % of the songs (78 songs) match both annotations, and that the second quadrant actually only has 22 songs in it.

4.2. Global results

The results obtained by the tests specified in section 3.1.5. Playlist evaluation are presented in Table 7. It is important to know that these results are the arithmetic means of each one of the metrics specified, based on all the measures made (9700 songs = 50 repetitions × 4 folds × all songs on each test fold – 2 folds have always 49 songs, while the other 2 have always 48):

Metric	Results (arithmetic mean)
Playlist first song match (size 20)	4.11 %
Percentage of playlist songs match (size 5)	19.03 %
Percentage of playlist songs match (size 10)	35.35 %
Percentage of playlist songs match (size 20)	58.73 %

Table 7: Experimental results

From this table, it is easy to understand why the exposed metrics tests results grow in percentage.

The first metric has undoubtedly a really low result, but that can be somewhat understand if we have in mind that it measures the average percentage of times the closest song in a top distance list of 48 or 49 songs is the same in the annotation and prediction ones.

The other three metrics are similar among themselves, the only variable being the playlist size. Since these three metrics measure the average percentage of common songs (therefore ignoring the order of appearance of them) between the top annotation and top prediction lists (with playlists sizes of 5, 10 and 20 songs), it is easy to understand that, the bigger the playlist size is, the most probable is to find common songs in the two top lists. To a playlist of size 20, in these conditions, the results are reasonable, with an average matching of almost 60%.

However, these results also predict that if the dataset (and consequently the fold) size is increased, these four metrics values will probably drop. The limitations that helped led to his results were already pointed out and some suggestions are made abaixo (section 5.1. Future work).

4.2.1. Classification results

However, it is noteworthy that the fact that feature selection was not performed was another factor that contributed to the somewhat low results achieved. In his thesis, using all the features available in MARSYAS, my colleague Renato Panda concluded that, as previous studies pointed out, valence values are easier to predict, in comparison with the valence ones. His tests used R^2 and RMSE statistics (see section 3.1.3. Classification for their definition) to verify this and indeed they revealed that the R^2 value for arousal reached 57.9% but only 3.24% for valence (Table 8). The arousal values are similar to the ones observed in (Yang, Lin, Su, & Chen, 2006), but in what concerns with the valence values, they reached 28.1% in the same paper, much more than the ones obtained here. This discrepancy is probably due to the fact that, in the aforementioned paper, several feature extraction frameworks were used, which provided the authors with a wider range of features, some of them not present in MARSYAS. This is underlined by the fact that three out of the four most important features pointed out by Yang in his paper are no present in MARSYAS.

We also conducted a pilot study with MIR Toolbox, which led to a R^2 value of 25% for valence, which confirms the absence of some meaningful features in MARSYAS. Another cause to this low R^2 results for valence probably had to do with the use of the entire MARSYAS feature set. If feature selection had been made, some features (with uninteresting results for mood classification) would be discarded, thus increasing both the results (especially for valence) and the model accuracy.

Arousal				Valence			
SSE	SST	RMSE	R^2	SSE	SST	RMSE	R^2
0.996816	2.34311	0.220108	0.57985	1.15312	1.1943	0.241297	0.0324472

Table 8: Global classification results (using all features)[21]

[21] Results kindly provided by Renato Panda

Also, looking at the placement of the predictions on Thayer's model it is easy to conclude that all songs are gathered within the 0.50 distance limit (orange circumference, see Figure 30) and that the valence values practically do not change (a very small variation almost places them all over the arousal axis).

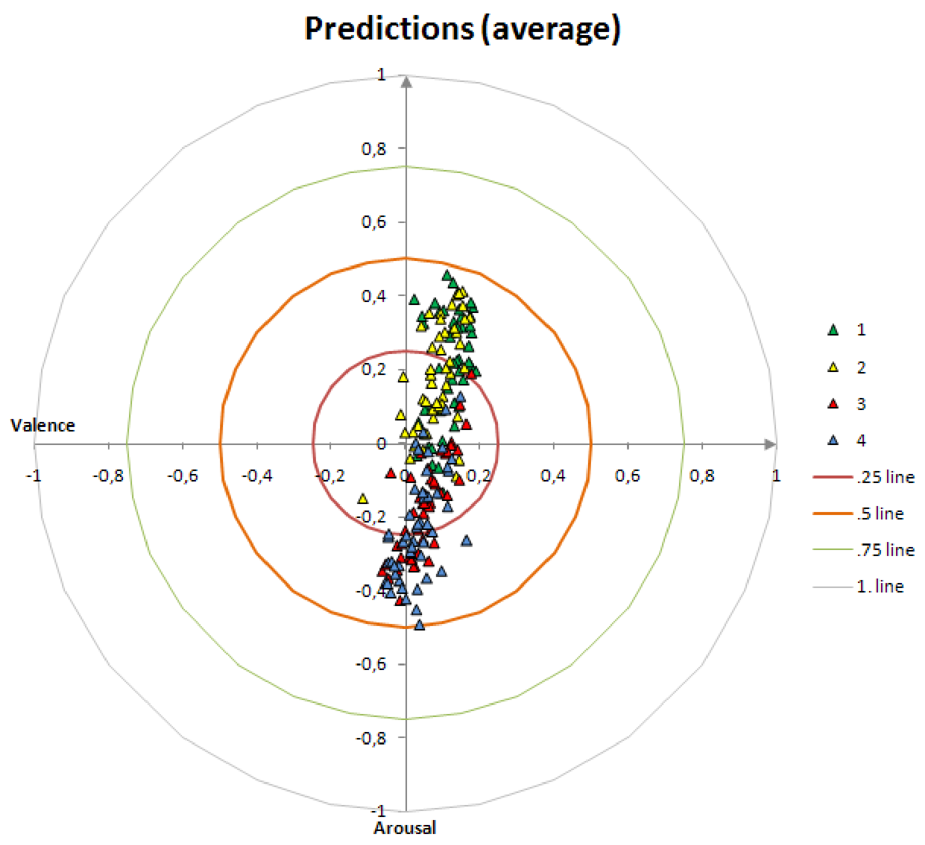


Figure 30: Global predictions in Thayer's model (using all features)[19]

As previously pointed out, improvements to these results can be achieved by adding meaningful features, such as tonality, multiplicity, spectral dissonance and chord, as mentioned in (Yang, Lin, Su, & Chen, 2006) (other suggestions are exposed in section 5.1. Future work).

5. Conclusions

This project revealed to be a valuable one, consisting in a valid study, with interesting and credible results. Personally, the main goals and purposes of this work were achieved, and one hint of the relevance of these results on the MIR and MER fields is the possibility of writing (or, at least, to take part on) a paper about the subject.

Of course that there were time limits and some difficulties, the main ones being the complexity of the MARSYAS core code (which increased the learning curve), and its instability, due to constant updates that sometimes generated memory leaks in some versions and so, the work done is a functional prototype of the planned mood application and not a final version.

Yet, the features that weren't developed so far are well documented in both section 5.1 and Appendix A. This way, it is easy to understand what is already done and what paths can be followed.

It is also noteworthy all the study and knowledge gained through the planning and development of this project, which is well documented in this thesis report. This consists in a solid base and introduction to the study of the MIR and MER fields. Also, despite being mainly a research project, software engineering techniques were also used, which ensured planning and eased team work and task distribution among all the members of the project.

In conclusion, this project was undoubtedly worthy and added value to the areas covered. Its results are another small but important step in the research of these recent fields (MIR and MER), and so, more data is available to all the researchers on the subject.

5.1. Future work

In a research project of this kind, usually many improvements can be done, and this one is no exception. Mainly due to time restrictions, some of the requirements planned weren't developed, although the main objectives were achieved.

In what concerns to MARSYAS and playlist generation, some of the improvements could include the use of a bigger, balanced dataset (instead of the Yang one), or, in alternative, to assure that both the training and test sets of the current dataset are balanced, including all the folds (with an equal number of songs for each quadrant). However, the latter option would mean that only 22 songs from each quadrant would be used in total (since the annotations analysis revealed that only 22 songs belonged to the second quadrant, the one with less number of songs).

Also, more metrics could be studied (which includes distance metrics, like the Euclidean and Manhattan distance between each songs feature vector, or a membership-like feature vector distance) – for instance, extract more playlist generation statistics by changing the number of repetitions and/or folds and comparing the results. Other important tool would be the use of FFS, to eventually increase the quality of the results obtained. FFS would possibly provide better results by choosing the optimal set of features (the ones that provided the best prediction results). Last, but not least, other classifiers (k-NN, GMM) could be used to compare results, and their parameters studied, compared and tuned (including the classifier used – SVM), to achieve better results.

In what concerns to the application (the client, backoffice and server), it could be improved to include all the options planned in the requirement analysis document (see Appendix A) that weren't developed. In what concerns to the database, the one currently specified and implemented is prepared for generalization and includes all the current necessary features.

Finally, at least the playlist generation experiments, metrics and results, could be documented on an article paper written on the subject.

6. References

Aucouturier, J.-J., & Pachet, F. (2002). Music Similarity Measures: What's The Use?

Audio Music Mood Classification Results. (s.d.). Obtido em 20 de January de 2010, de Audio Music Mood Classification Results - MIREX 2008: http://www.music-ir.org/mirex/2008/index.php/Audio_Music_Mood_Classification_Results#MIREX_2008_Audio_Mood_Classification_Run_Times

de Cheveigné, A., & Kawahara, H. (2002). YIN, a fundamental frequency estimator for speech and music. *Acoustical Society of America* .

Euclidean distance. (s.d.). Obtido em 22 de January de 2010, de Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Euclidean_distance

jAudio. (s.d.). Obtido em 19 de January de 2010, de jAudio: http://jmir.sourceforge.net/jAudio.html

Laar, B. v. (2005). Emotion detection in music, a survey.

Lartillot, O. (2008). MIRtoolbox 1.1 User's Manual. Jyväskylä, Finland.

Lartillot, O., Toiviainen, P., & Eerola, T. (s.d.). *Department of Music: MIRtoolbox*. Obtido em 20 de January de 2010, de Jyväskylä University: https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

Lu, L., Liu, D., & Zhang, H.-J. (2006). Automatic Mood Detection and Tracking of Music Audio Signals. *IEEE Transactions on Audio, Speech and Language Processing , 14* (1).

Marsyas. (s.d.). Obtido em 20 de January de 2010, de About: http://marsyas.info/about/overview

McKay, C. (s.d.). jAudio: Towards a standardized extensible audio music feature extraction system.

Meyers, O. C. (2007). *A Mood-Based Music Classification and Exploration System*.

Overview. (s.d.). Obtido em 20 de January de 2010, de Marsyas: http://marsyas.info/about/overview

Paiva, R. P. (2006). *Melody Detection in Polyphonic Audio*.

Pampalk, E. (2005). Tutorial: Music Similarity. *ISMIR*.

Pauws, S., & Eggen, B. (2002). PATS: Realization and User Evaluation of an Automatic Playlist Generator.

Ribeiro, B. (2009). Pattern Recognition Techniques slides.

Scherrer, B. (2007). Gaussian Mixture Mode lClassifiers.

Taxicab geometry. (s.d.). Obtido em 22 de January de 2010, de Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Manhattan_distance

Tzanetakis, G. (2002). *Manipulation, Analysis And Retrieval Systems For Audio Signals*.

Vossen, M. P. (2005). *Local Search for Automatic Playlist Generation*. Eindhoven.

Yang, Y.-H., Lin, Y.-C., Su, Y.-F., & Chen, H. H. (2006). A Regression Approach to Music Emotion Recognition.

Appendixes

Appendix A - Requirements analysis

The following sections present with some detail the requirements analysis of the application that is subject to be developed (objectives, functionalities and features), since this is a crucial part of the design of a software application. It is also presented some prototype screenshots of the client GUI interface, showing part of the playlist generation functionalities.

This requirement analysis was achieved through an extensive brainstorming involving Renato and the thesis supervisor, Professor Rui Pedro Paiva, which means that it is presented an extensive list of desirable features. Of course that, when hard work starts, some functionalities and details can change, while some of the software features can prove to be harder (or easier) than expected to fulfill.

A.1. Requirements description

After some discussion and brainstorming, it seems clear that there will be developed three applications: a client application, a server and an administration one. It is also required that both client and administration applications are able to run both in online and standalone mode (running both the client - or administration - and server application in the same host, and communicating via localhost). The client will be able to perform a query (in a variety of ways, as we will see later), while the server application will rely on doing the "heavy" stuff (extracting features, calculating arousal-valence coordinates, adding/removing/editing songs in the database and selecting songs to form the output playlist, retrieving the information about them afterwards to the client application). The architecture of these applications is graphically described in Figure 34. Its details will be discussed in the following sections.

A.1.1. Client application

On the client side, it will be developed a GUI that can offer to the user the possibility of choosing a song from a music file in his computer to add to the database, or perform a query in it. It will be also possible to choose from a two-dimensional valence-arousal axis graphic (based on the Thayer's model) – selecting a point on it, or several ones (in order to choose a trajectory for the playlist to be generated), or even selecting an area in the graphic, which would generate a playlist (retrieved by the server) with songs within the selected zone (or at least, the closest possible). It must be also possible to show all the songs of the database in the valence-arousal graphic (again, as points). Due to the fact that with this option enabled the graphic can easily become crowded (especially if the dimension of the database is big), it can be desirable to implement a zoom tool, where the user could choose an area to enlarge. Other possibilities include the use of filters to refine search on the database (by genre, musical attributes, features – like tone, rhythm, etc).

Besides all the aforementioned search mechanisms in the previous paragraph, the client application must also present the user with a number of configurable options, like the size of the generated playlist, the option to evaluate the mood of a song choosing what features (or categories of features) to use (or not) when retrieving the mood of a song. The GUI must present the features hierarchically (each one inside the feature category where it belongs). It must be also desirable to choose which type of distance will be use to compare the selected song with the rest of the database (by default, it is used the Euclidean distance). It will also crucial to have the possibility of choosing the classifier to use (SVM, k-NN, GMM, etc). It would be also desirable to have the option to export the playlist to a standard format (M3U[22], for instance). The application will also have mood tracking (analysis of mood change / segmentation during a song) functionalities, which are more related to Renato's work.

It will be necessary to implement a protocol as a guideline to the communication between the client and server application. All the users must have a username and a password in order to log in into the system and communicate with the server (they must register in the user's database). This way, it will be possible to track who is logged in the system. This communication should be encrypted (password, at least, as mentioned before).

After the query done by the user, the client application must receive from the server the ID of the song, the valence and arousal coordinate ant the level of similarity with the point chosen in the in the arousal-valence axis 2D graphic, the song chosen by the user or mood tracking information of a song, depending on which option was chosen. That results must be presented to the user, showing the name of the artist, the name of the song, the position in the arousal-valence 2D graphic and the distance to the point selected as input in a graphic way. It must be also possible to hear each song in

[22] http://en.wikipedia.org/wiki/M3U

the generated playlist through a link (streaming from the server, at least a portion of each song).

As mentioned earlier, the GUI should be implemented in Qt, which will interact directly with MARSYAS. A GUI prototype of the client application is presented in Figure 31 (section A.2. GUI prototypes).

To summarize what has been said, a table with functional requirements is now presented (Table 9):

Requirements	**Details**
User credentials	- Register user - Login - Logout - Edit profile / change password
Server communication	- Connect to server (local / remote) - Transmit queries - Receive / process results - Receive / download audio file or stream - Send song to be processed
Browse information	- View all database songs in Thayer's model (database map) - Create playlists by: - selecting/uploading a song file - a point in the arousal-valence 2D graph - playlist trajectory - selected area in the arousal-valence 2D graph - Apply zoom to the database map - Filter map view by: - musical genre - artist - album - year (or year interval) - other relevant musical attributes - a selected group of audio features - View mood tracking information (graph) for a database song - View mood tracking in real time (graph tracing) - View mood tracking information in wave form graph - Other methods of viewing results - Export playlist (M3U) - Exclude songs from suggested playlist - Download songs

Table 9: Client application functionalities

A.1.2. Server application

The server application (as seen on the architecture scheme in Figure 34), as expected, should answer the queries from the users. It should retrieve playlist data (based on the information provided in the query performed by a user from the client application), mood tracking information or streaming / download of a song. It will be divided in three different parts: a client daemon, an administration daemon and MARSYAS itself, besides the database containing the arousal-valence classification information for each song and the musical database, that will be composed by the test collection mentioned earlier (section 2.7 Test collection). As it was previously mentioned, at least the password should be encrypted in order to ensure security to the system. In order to achieve this it will be necessary to implement a protocol to successfully perform all the operations needed and provide authentication credentials (username, password).

Also, the server application should ensure all the tasks involving MARSYAS (processing songs and extracting features), interact with the database and provide all the communication with both the client and administration application, using the referred protocol to serve these purposes. The database could range from a simple set of files containing information to something more powerful and complex (a Database Management System (DBMS), e. g. SQLite[23]) – it will depend on the needs and size of the database when the application starts to be coded. It will provide methods to add, delete and edit song information in the database, although these will be accessed in the administration application. The same should be possible for classifiers (add, remove, edit) and users. Finally, it should implement methods to change the user's permissions (regular or administrator), create, process and drop the entire database.

It is now presented a table with functional and quality functionalities desired for the server application (Table 10):

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

Requirements	Details
User accounts	- Create account - Remove account - Block / Ban account - Edit profile - Encrypt sensible user information (password at least)
Client communication	- User authentication - Process user queries - Return query results (M3U, song information) - Stream / send songs - Receive songs from users / administrators - Remove song from database - Change server settings (administrators)
Database management	- Create database - Drop database - Delete database - Insert new songs information - Update / edit existent information - Select / browse songs information
Audio processing	- Downsample songs - Extract audio features - Apply classifiers

Table 10: Server application functionalities

The data model should be studied previously to increase query performance (on the case of using a DBMS, as mentioned earlier) – for instance, a table containing information about the songs connected through the song ID to a table with the arousal, valence and classification information of all the songs.

A.1.3. Administration application

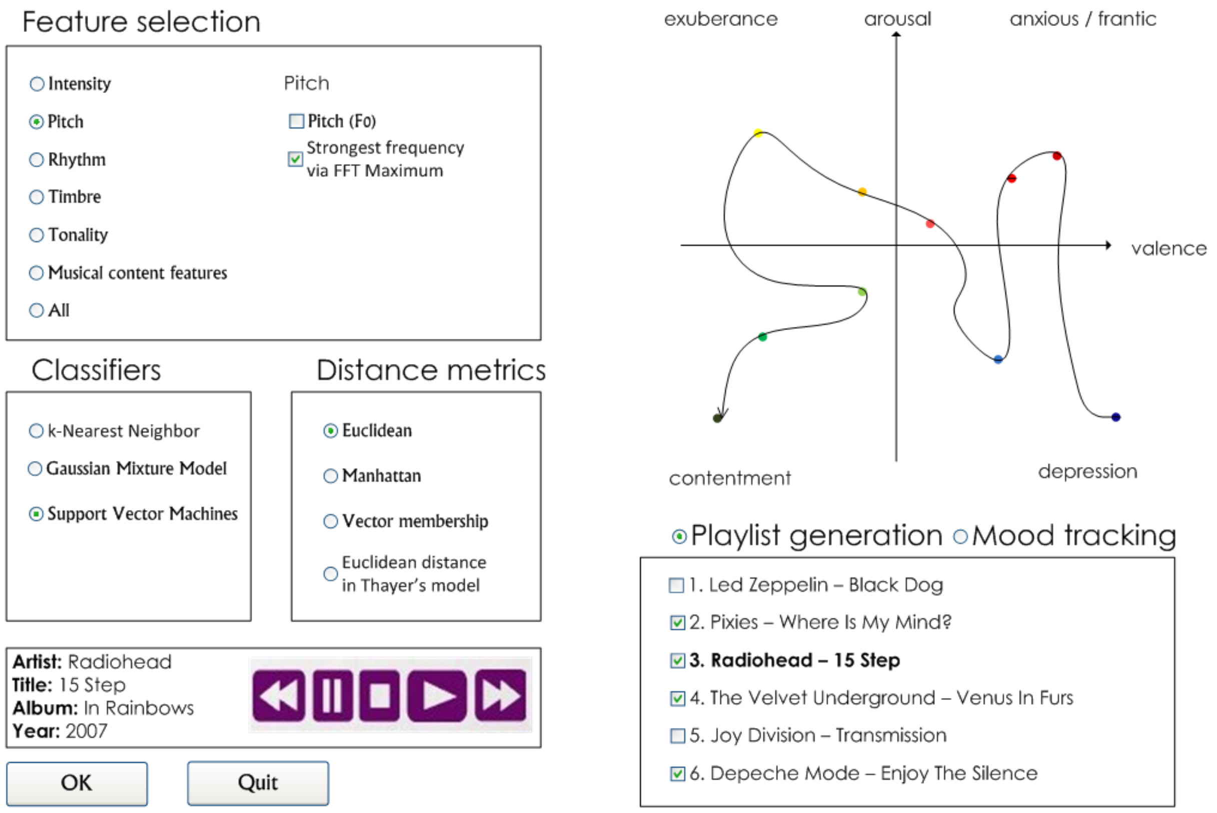
This application will function as the administrative tool. It will present a GUI to the administrator(s) where it could be performed all the operations described previously (add, edit and remove songs properties, classification methods, users and permissions from the database, along with methods to manipulate the database itself). The administrators must also have a username and a password to access the backoffice application and the information must be again encrypted and the aforementioned protocol sill be used to send information to the offline module, so it could reproduce in the database the changes made by the administrator(s). This application should also work in online or standalone mode, similar to the client application.

Again, a table with functional and quality requirements is now presented (Table 11):

Requirements	**Details**
User accounts	- Login (as administrator) - Edit user account (permissions, password) - Remove user account - Create account - Block user
Server communication	- Connect to server (local / remote) - Transmit queries - Receive / process results - Upload music files (with title/artist and other information)
Manage database	- Create a new songs database - Drop existent database - Edit database information - Remove song from database / server - Add song to database / server (and order feature extraction and classification) - Add song annotations (arousal-valence values and segmentation information, for validation tests) - Add song information
Manage settings	- View existent audio features (grouped by category) - Edit list of selected audio features - Change distance algorithm used (similarity) - Change default classifier (SVM, GMM, k-NN) - Edit classifier parameters - Change default taxonomy - Perform analysis of results (model accuracy) - Close users registration

Table 11: Administration application functionalities

A.2. GUI prototypes



Feature selection

- ○ Intensity
- [✓ Pitch]
- ○ Rhythm
- ○ Timbre
- ○ Tonality
- ○ Musical content features
- ○ All

Pitch

- ☐ Pitch (F0)
- [✓ Strongest frequency via FFT Maximum]

Classifiers

- ○ k-Nearest Neighbor
- ○ Gaussian Mixture Model
- [✓ Support Vector Machines]

Distance metrics

- [✓ Euclidean]
- ○ Manhattan
- ○ Vector membership
- ○ Euclidean distance in Thayer's model

Artist: Radiohead
Title: 15 Step
Album: In Rainbows
Year: 2007

[✓ Playlist generation] ○ Mood tracking

- ☐ 1. Led Zeppelin – Black Dog
- [✓ 2. Pixies – Where Is My Mind?]
- [✓ **3. Radiohead – 15 Step**]
- [✓ 4. The Velvet Underground – Venus In Furs]
- ☐ 5. Joy Division – Transmission
- [✓ 6. Depeche Mode – Enjoy The Silence]

Figure 31: Prototype screenshot example for the client application GUI

As it can be seen above (Figure 31), the client will have the possibility to choose the feature category that will classify the song, and specify which features will be used inside each category. The same approach is made with classifiers and distance metrics. Above this part of the GUI, Figure 31 shows a simple music player, with the common commands (play, pause, stop, skip forward, skip backwards) and information (song title, artist, album and year of release). A pair of buttons performs the classification according to the specified options or cancels it. On the right side of the GUI it can be seen a representation of the Thayer's emotional model, with dots connected, forming a playlist.

It is important to notice that each point has a different color or tonality, according to its position in the bi-dimensional axis: the points in the depression quadrant all have a blue color, with the tonality changing from light blue ("soft depression", somehow near the origin of the axis) to dark blue ("deep depression", with almost minimal arousal and maximum valence). The other colors include red (anxious/frantic quadrant), yellow (exuberance) and green (contentment). It is desirable that each song (point) can be clicked, showing information about the respective song. Finally, below this model is information panel that can switch from showing information related to the playlist generation (the example in Figure 31) and mood tracking.

Also, as the name suggests, a prototype is only a conceptualized interface of the future application, and for this reason the final application may have little in common in what concerns to the GUI interface (the design of the final application GUI will be created "by the book").

Appendix B - Software design

After describing with some detail the application to be developed, the requirement analysis generated some use cases diagrams, which are now presented and can give us a clearer, better idea of the features to implement.

B.1. Use cases

Use cases are part of the Unified Model Language and represent system behavior, describing the interactions between one or more actors and the system itself (from the actor's point of view).

In this phase, only a couple of use cases were developed, since it is too soon to anticipate exactly how the system will work. With the application development, surely more use cases will be designed and the ones presented below will be improved.

The following use cases illustrate the main actions of the mood tracking application. The first one (Figure 32) illustrates how a "normal" user interacts with the system, detailing the actions he can perform when using the client application, while the second one enlightens the actions that a administrator is able to do by using the administration application (Figure 33).

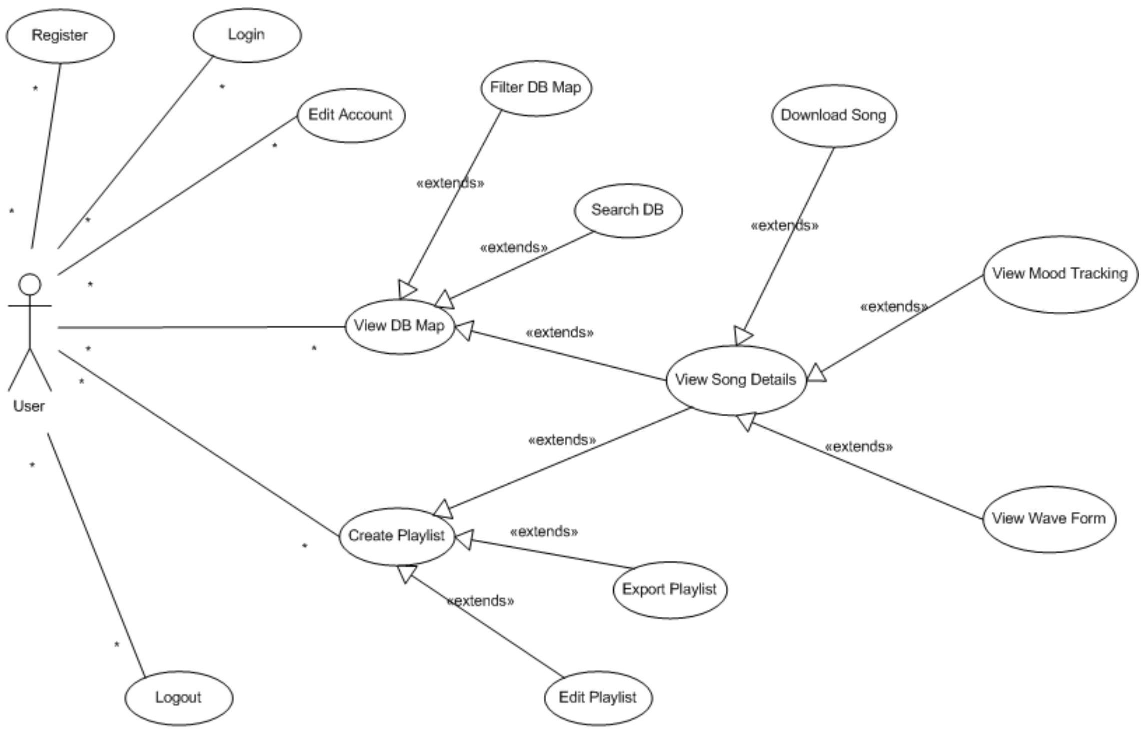


Figure 32: Use case nº 1 ("normal" user)

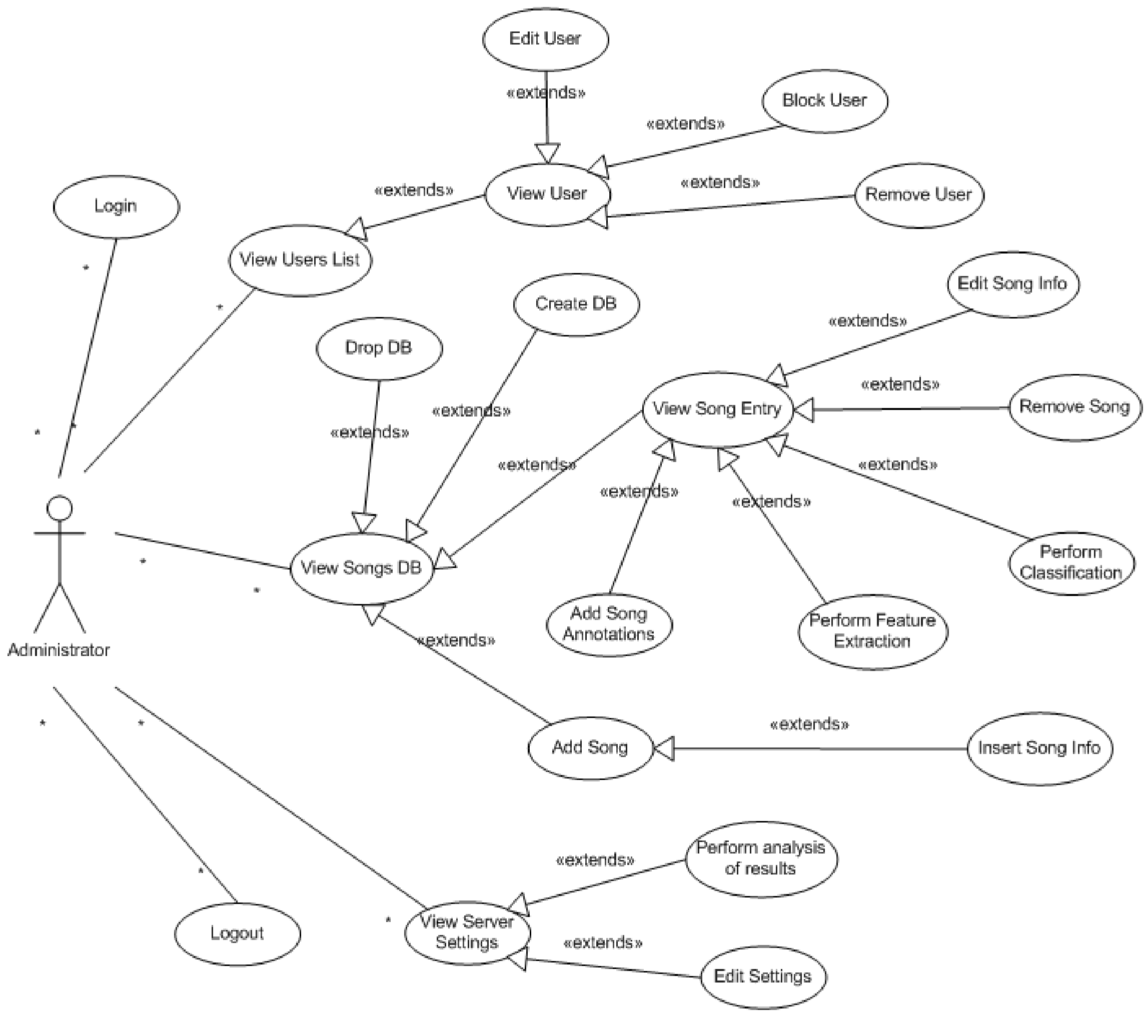


Figure 33: Use case nº 2 (administrator user)

B.2. System architecture

The system architecture represents graphically the requirements analysis as it was depicted in section A.1. Requirements description. The three main applications are represented, as also are the most important internal modules of the server application (Figure 34).

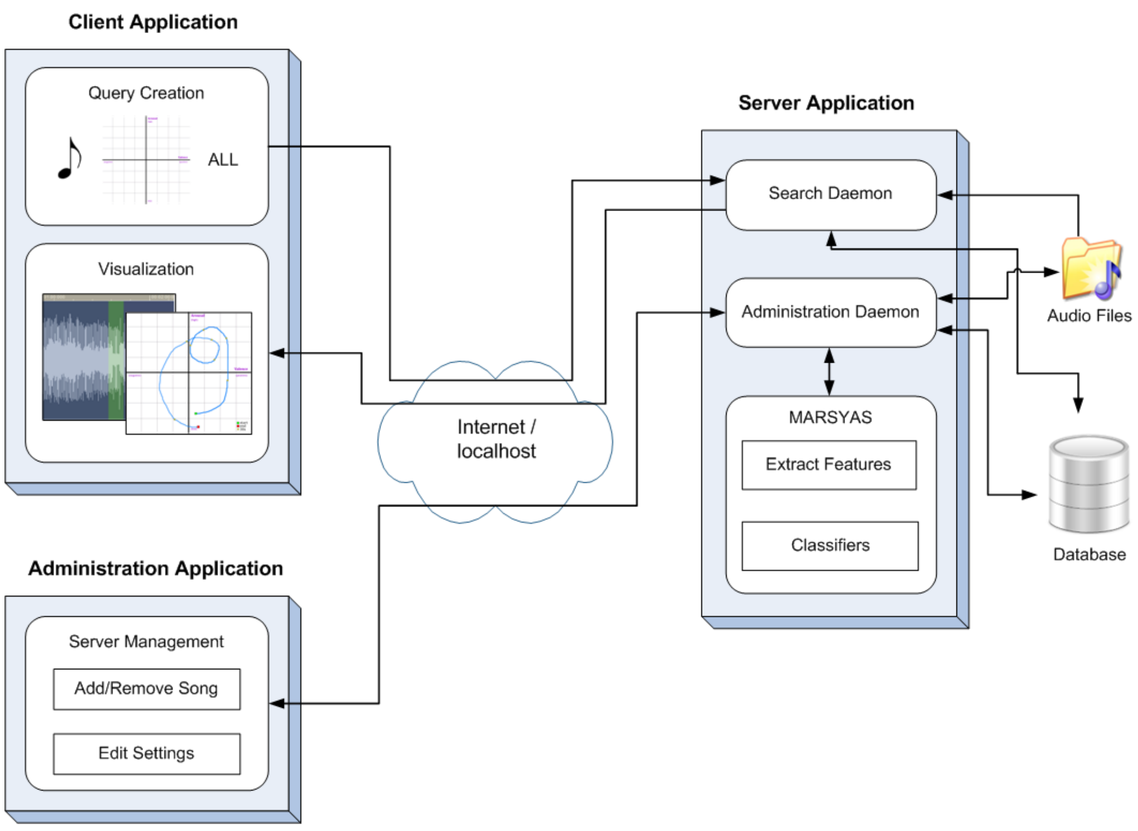


Figure 34: System architecture

B.3. Data storage

It is now time to specify how information will be stored in our server application. Since the initial brainstorming, several options were addressed, from simple encrypted text files to something more sophisticated, like XML files or even “real” databases. After realizing that the expected size of the musical database reaches easily hundreds or thousands of files and consequentially large amounts of data (the mood tracking data, for each second per song is the main responsible for this situation), it became clear that simpler solutions like text or XML files would not have the best possible performance (heavy and slow basic operations – e.g. search, edit and delete). On the other hand, using a full DBMS (Postgres or MySQL, for instance) would be too complex for the purpose we have in hands, since it would be imperative to have the system installed in every application and always running, making it much more computationally heavier. Also, the fact that both the client and the administration applications must be usable in both online and offline mode affected this decision - if the application was only to be used in online mode, probably a DBMS (more specifically MySQL) would be chosen.

Dealing with these circumstances, SQLite was chosen, since it is a software library implementing a self-contained, serverless, zero-configuration, transactional SQL database engine. It can be seen as an embedded SQL database engine, but unlike the majority of the other SQL databases, it does not have a separate database engine, reading and writing information directly to ordinary disk files. Also, the generated database file is cross-platform, storing all the information in a unique, small footprint file.

B.3.1. Data model

In this section is presented the Entity-Relationship diagram, which represents the data in a abstract and conceptual way. This scheme is very important since, if well-defined, it can assure minimum storage space an optimal performance when accessing the database. To guarantee it, the database will be normalized to at least the 3rd normal form (and if possible, to the Boyce-Codd form, at least for the tables expected to be bigger / more used.

The database should keep all the critical information about songs (ID, title, artist, album, year, filename, among others), mood information (arousal-valence values), usernames and respective encrypted passwords. A tentative preliminary version of the aforementioned diagram is now presented. It already covers some of the needs of the desired database and prevents duplication of values null cells, although some improvements to the mood information part are expected in future diagrams. Figure 35 presents the Entity-Relationship diagram:

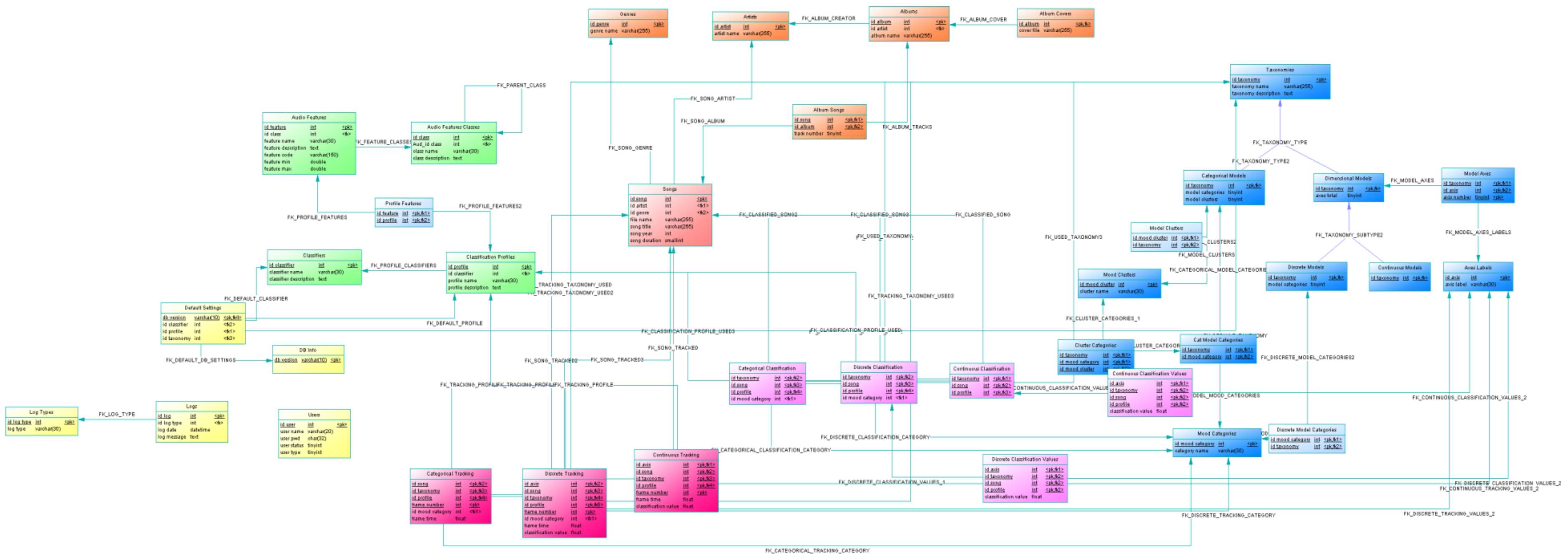


Figure 35: Entity-Relationship diagram