U N I V E R S I T Y O F T A R T U
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Anastasia Semyonova

Automatic Personalized Playlist Generation

Master's thesis (30 ECTS)

Supervisor: Konstantin Tretyakov, M.Sc.

Autor: .... "....." juuni 2011

Juhendaja: .... "....." juuni 2011

Lubada kaitsmisele

Professor: .... "....." juuni 2011

TARTU 2011

Acknowledgements

This thesis describes the graduation work for my study of Institute of Computer Science at the University of Tartu. The project is done on the topic of automatic personalized generation of music playlists.

Firstly, I would like to thank my supervisor, Konstantin Tretyakov. His assistance in all phases of the project was of great importance to me, as we discussed all aspects of the problem and the solutions. And I would like to commemorate the many hours Konstantin has spent reading and giving detailed comments on this thesis and helping to solve technical problems. I am specially grateful for his challenge to propose and supervise the completely new for the University of Tartu research topic and flexible attitude to my personality and life decisions.

My thanks also go out to the Institute of Computer Science of the University of Tartu (UT), and especially to Jaak Vilo, the professor of the UT, the leader of BIIT research group who was supporting me with advices and gave a chance to take part in Estonian Summer School in Computer and Systems Science (ESSCaSS) 2009, 2010; Utrecht Summer School of Music Information Retrieval (USMIR) 2010; and the conference of the International Society for Music Information Retrieval (ISMIR) 2010. It was a great impact in my level of education.

Also, I am especially grateful for Archimedes Foundation that sponsored my participation in thematic USMIR 2010 summer school and ISMIR 2010 conference that helped me to get a broad overview of the research topic I was working on, meet interesting people working in this field and choose the topic of my master thesis research.

I would also like to thank all students, collegues and my friends for inspiring and not giving up on me, especially Thomas Prätzlich, Tom Collins, Anastassia Kozlova, Aleksei Suharev, Anna Leontjeva, Jekaterina Štšogoleva, Aleksei Lissitsin, Angelina Timoshkova, Alina Kravchenko and Nadezhda Plishkina.

Contents

Introduction

Music in digital form is widely spread nowadays. Traditional methods of listening to and discovering music, such as radio broadcasts and record stores, are being replaced by personalized ways to hear and learn about music [CVG+08]. In recent years, due to the broad adoption of digital music and personal digital music players, there has been increasing interest in automatic generation of playlists. A playlist may be defined as a finite sequence of songs which is played as a complete set [RBBC].

The easiest and the most effective way for constructing a playlist is to do it manually. However, this is a time-consuming task that, at the same time, allows a certain degree of freedom. The huge size of music collections nowadays exceeds a person's ability to recall which compositions might comply best to current mood or situation. Moreover, many social contexts, such as, for example, being at work, or driving a car, do not allow to waste too much time on hand-picking music. In our work we focus on the development of an algorithm for automatic construction of personalized playlists, primarily targeting casual users rather than professional DJs. Such a playlist consists of a start and an end seed songs, both chosen by a user. The choice and order of songs inbetween should correspond to the user's personal taste, extracted from pre-labeled "good" and "bad" playlists. This algorithm can be further used for constructing more complex personalized playlists.

The solution contains two main parts: the choice of the playlist generation method and the construction of the evaluation function for discriminating good playlists from the bad ones. For the first task we consider *shuffle*, *randomized* and *genetic* generation algorithms. For the second task we use the Naïve Bayesian classifier. The input features for the classifier are comprised of audio content-based features of the songs and their normalized variability over the playlist. As a result we develop an algorithm for automated personalized playlist generation (APPG) that, according to our evaluations, satisfies the user expectations better than random shuffling. However, we should assume that algorithm has a high potential to be improved.

The organization of the thesis is the following. Chapter 1 provides the preliminary background in digital audio processing and machine learning necessary to understand the rest of the work. Chapter 2 provides the formulation and motivation of automated personalized playlist generation problem (APPG), as well as reviews some related work in this area. The description of the proposed algorithm for APPG together with some implementation details is given in Chapter 3. Chapter 4 describes the results of performance evaluations. The thesis is concluded by a Conclusion section which summarizes the results and suggests ideas for future work. The source code used in the experiments is supplied on a separate CD (Appendix A).

Chapter 1

Preliminaries

In this chapter we present brief definitions of the basic concepts used further in the thesis.

1.1 Mathematical Treatment of Music

Before constructing the algorithm to solve the problem of automatic personalized playlist generation (APPG) we must clarify what a music playlist is, what parameters influence the subjective value of playlist goodness, and how they can be computed.

1.1.1 Digital Representation of Sound

Music playlists consist of musical compositions. Music, is a melodic type of sound. Sound is, in turn, an oscillation in air pressure, produced by a vibrating object, such as a musical instrument. Moreover, sound is a travelling wave (sound wave) that can be represented as a continuous periodic function of time (Figure 1.1).

In order to process sound waves on a computer, the corresponding function needs to be converted to digital form – a process known as *sampling* (Figure 1.2). Sampling is performed by measuring the continuous signal at regular intervals, thus converting it into a stream of numbers. These numbers may then be stored on a computer. Each measurement is referred to as a *sample*.

Once we have converted the sound wave into a stream of numbers we can store and process this information on a computer. The data of a music file can be stored in many different ways. One of the main parameters is *sampling frequency* of the stored signal – the number of times per second samples were taken. This attribute, also known as *sampling rate*, is measured in Hertz (1 Hz = 1 measurement per second).

The optimal sampling frequency may be different, depending on the situation. If we store data for listening, it is always preferable to choose the composition with the highest reasonable sampling frequency. For the purposes of

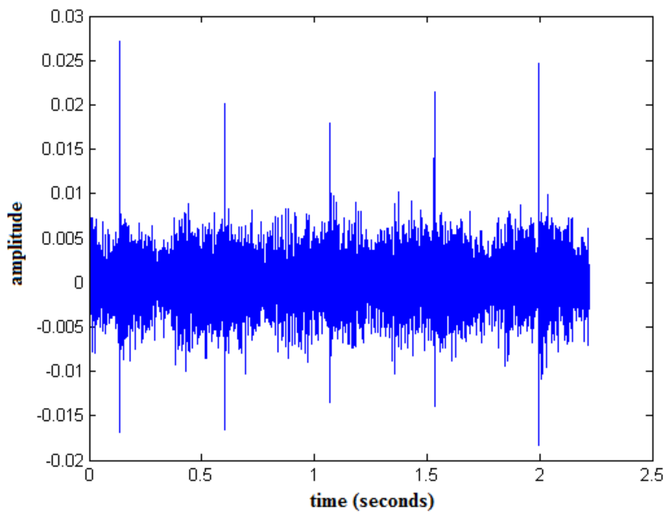


Figure 1.1: Sound wave can be represented as a function of time.

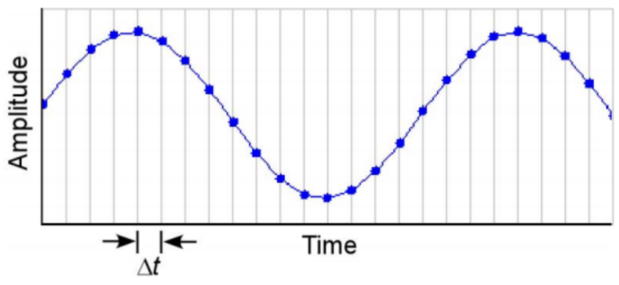


Figure 1.2: The sinusoidal curve represents the continuous analog waveform being sampled. Measurements of the instantaneous amplitude of the signal are taken at a sampling rate of Δt [ft].

analysis we usually select some golden middle, so that the calculations would be faster and yet no important information is lost.

1.1.2 Spectral analysis

One of the classical methods of extracting the properties of a sound wave is known as *Fourier series*, *Fourier transform* or *spectrum*.

Any 2π-periodic function may be represented as an infinite sum of simpler functions – sine waves (Figure 1.3):

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)], N \geq 0$$

The coefficients a_i and b_i of the Fourier series can be found as

$$a_i = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \cos(nt) dt,$$

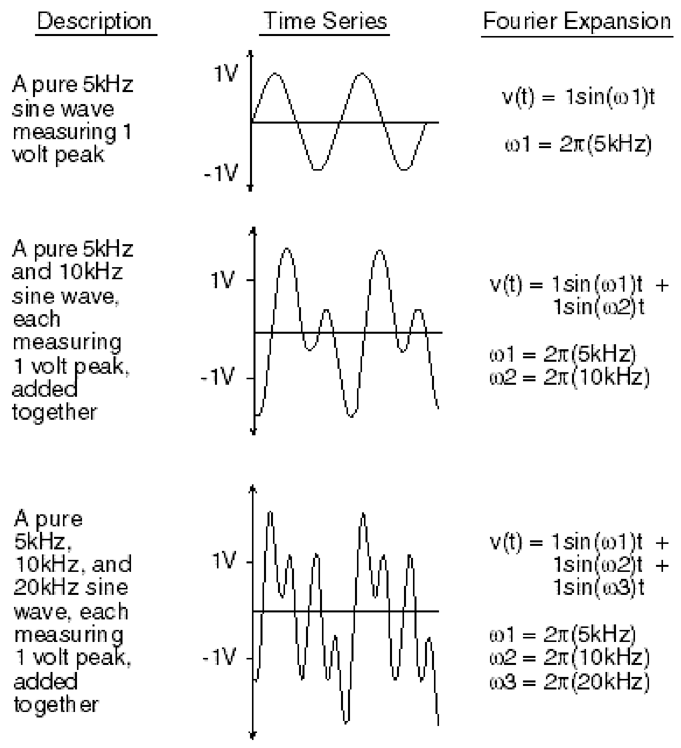

Description
Time Series
Fourier Expansion
A pure 5kHz sine wave measuring 1 volt peak
1V
-1V
v(t) = 1sin(ω1)t
ω1 = 2π(5kHz)
A pure 5kHz and 10kHz sine wave, each measuring 1 volt peak, added together
1V
-1V
v(t) = 1sin(ω1)t + 1sin(ω2)t
ω1 = 2π(5kHz)
ω2 = 2π(10kHz)
A pure 5kHz, 10kHz, and 20kHz sine wave, each measuring 1 volt peak, added together
1V
-1V
v(t) = 1sin(ω1)t + 1sin(ω2)t + 1sin(ω3)t
ω1 = 2π(5kHz)
ω2 = 2π(10kHz)
ω3 = 2π(20kHz)

Figure 1.3: Complex sound as a sum of sinewaves.

$$b_i = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\sin(nt)dt,$$

The practical meaning of the Fourier transform for sound processing is that any sound can be represented as a combination of *frequencies*. The spectrum provides the intensities of those frequencies. The importance of Fourier representation lies in the fact that human sound perception is also based on spectral analysis, performed by the biological machinery of our ears [ear].

For representing digital sound, a discrete version of the Fourier series is commonly used, the *discrete fourier transform (DFT)*. The DFT can be computed effciently using the algorithm known as the *fast fourier transform (FFT)*. We refer the reader to the book of D. Benson [Ben06] for a thorough coverage of this and related topics. For practical purposes it suffices to know that the application of FFT to a discrete signal (i.e., a vector of numbers representing signal intensities at different timepoints) produces a discrete spectrum (i.e., a vector of numbers representing frequency intensities).

1.1.3 Higher-level Audio Features

The raw frequency intensities can be further transformed into yet more informative higher-level sounding characteristics. A comprehensive overview of numerous such features is provided in [mir]. In the following we provide a description of the most important of them.

Spectral Centroid (`centroid`)

The spectral centroid is a measure used in digital signal processing to characterise a spectrum. It indicates where the "center of mass" of the spectrum is. It is calculated as the weighted mean of the frequencies present in the signal, with their magnitudes used as the weights by the following formula:

$$\texttt{centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)},$$

where $x(n)$ represents the frequency value (magnitude of the frequency bin number n), and $f(n)$ represents the central frequency of that bin.

Spectral Flatness (`flatness`)

Spectral flatness, measured in decibels, provides a way to quantify how tone-like or noise-like a sound is. High spectral flatness indicates that the spectrum has a similar amount of power in all spectral bands – this would sound similar to white noise. Low spectral flatness indicates that the spectral power is concentrated in a relatively small number of bands – this would typically sound like a mixture of sine waves, and the spectrum would appear "spiky".

Spectral flatness is defined as follows:

$$\texttt{flatness} = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{\sum_{n=0}^{N-1} x(n)}{N}}.$$

Mel-Frequency Cepstral Coefficients (`mfcc`)

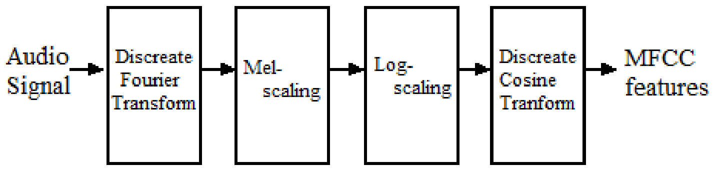


Figure 1.4: Mel-Frequency Cepstral Coefficients. After taking the log-amplitude of the magnitude spectrum, the FFT bins are grouped and smoothed according to the perceptually motivated Mel-frequency scaling. Finally, in order to decorrelate the resulting feature vectors a discrete cosine transform is performed.

Mel-Frequency cepstral coefficients (MFCC) are a set of perceptually motivated features that have been widely used in music information retrieval applications, such as speech recognition, audio similarities measures and genre classification. MFCC provide a compact description of the spectral shape of the sound, such that most of the signal energy is concentrated in the first coefficients. MFCC take human perception sensitivity with respect to frequencies into consideration, and therefore are best for speech/speaker recognition. Taken simply, the MFCC are computed by taking the spectrum of the signal, log-rescaling its both axes, and taking the spectrum again. The resulting vector contains the mel-cepstral coefficients (see Figure 1.4).

Spectral Flux (`flux`)

The spectral flux is a measure of the amount of local spectral change, i.e. the change in the spectra of nearby audio excerpts (windows). It can be used, among other things, to determine the timbre of an audio signal, or used in onset detection. Spectral flux at time t of the composition is usually calculated as the 2-norm (Euclidean) distance between the normalized (with uniformly increased (or decreased) amplitude of an entire audio signal) spectra at two nearby time frames t and $t-1$:

$$\texttt{flux}_t = \sum_{n=0}^{N-1} (\tilde{x}_t(n) - \tilde{x}_{t-1}(n))^2, \text{ [TC02]},$$

where $\tilde{x}_t$ and $\tilde{x}_{t-1}$ are the normalized magnitudes of the Fourier transform at the time frames t and $t-1$, respectively.

To calcalate `flux` we take mean value of all $\texttt{flux}_t$ within the texture window.

Spectral Irregularity (`irregularity`)

Spectral irregularity measures noisiness of the audio signal [Jen04]. The irregularity of a spectrum is the degree of variation of the successive peaks of the spectrum. Peaks are local maxima of the function values being bigger than all the points in a given neighbourhood (Figure 1.5).

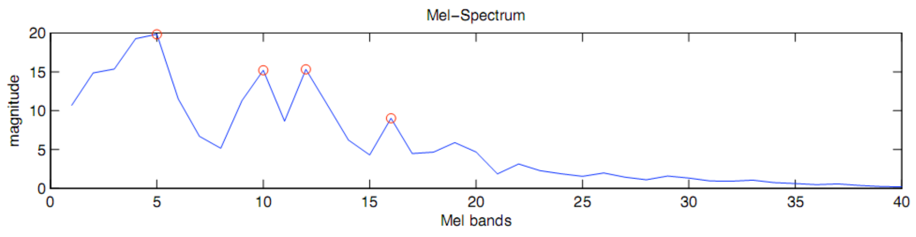


Figure 1.5: If x is a curve, peaks are represented by red circles.

The irregularity is measured as the sum of the squares of the differences in amplitudes between adjoining peaks:

$$\texttt{irregularity} = \frac{\sum_{k=1}^{N} (a_k - a_{k+1})^2}{\sum_{k=1}^{N} a_k^2},$$

where a_k is a local peak value.

1.2 Naïve Bayes as a Machine Learning Technique

Once an object (e.g. a playlist or a musical composition) is represented as a vector of numbers (*feature vector*), we would like to find a classification rule that can devise the category (e.g. goodness) of each object based on its features. Although it might be possible to design such an algorithm manually, by specifying some common sense rules of thumb, this is often inconvenient or complicated. An alternative approach is to collect a data set of labeled objects and *train* a classifier that detects the most appropriate rule by generalizing the information in the data. This approach is commonly referred to as *machine learning.* In the following, the process of constructing the dataset and training a particular classifier is provided.

Numerous algorithms for training classifiers exist. In our work we use the *Naïve Bayesian classifier.*

Let $F_1, F_2, \dots, F_n$ denote the feature values of an object (e.g. sounding characteristics of a song, or certain features of the whole playlist) and G be its class (e.g. goodness). By the classical Bayes approach, the object should be classified according to the most frequent class among the objects with same values of the feature variables in the training set:

$$\text{argmax}_G \Pr(G|F_1, F_2, \dots, F_n)\,.$$

When the number of variables is large enough, this would require a training set of unrealistically large size, where all possible combinations of values of the predictor variables would be available. The Naïve Bayesian method overcomes this practical limitation by using the assumption that predictor variables are independent for each class:

$$\Pr(F_1, F_2, \dots, F_n|G) \approx \Pr(F_1|G)\Pr(F_2|G)\dots\Pr(F_n|G)\,.$$

Now, by applying the Bayes theorem:

$$\Pr(G|F_1, F_2, \dots, F_n) = \Pr(G)\Pr(F_{1,F}2, \dots, F_n|G)/Z \approx$$

$$\approx \frac{1}{Z}p(G)\prod_{i=1}^{n} p(F_i|G)$$

where Z is a scaling factor dependent only on $F_1, F_2, \dots, F_n$, i.e., a constant if the values of the feature variables are known.

In spite of the "naïve" assumptions, the Naïve Bayes classifiers often works well for many complex real-world situations. An advantage of the Naïve Bayes classifier is that it requires a relatively small amount of training data to estimate the parameters necessary for classification.

1.3 Algorithm Validation

1.3.1 Cross-validation

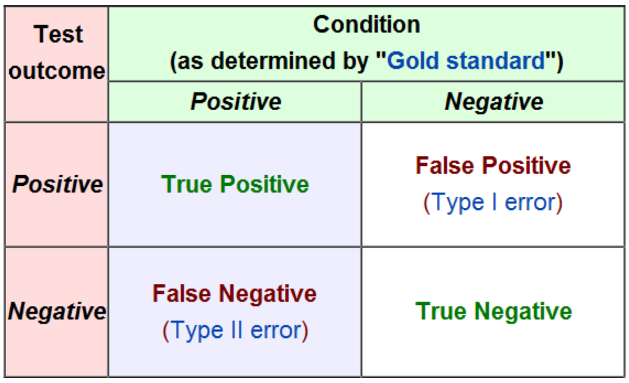
To validate the performance of a classification algorithm we used the *10-fold cross validation* technique. This means, we split the training set into 10 parts, use 9 for training and 1 for testing (measuring the precision), then repeat this procedure for each of the 10 parts and finally take the average of the 10 obtained precision estimates. This validation method is known to produce reasonably accurate precision estimates even when the training set is small [Wit05].

1.3.2 Evaluation Measures

To evaluate correctness of the classifier, we used such metrics as *classification accuracy*, *precision* and *recall*, all computed from the values of the *confusion matrix*.

Confusion Matrix

Each object in the dataset has an assigned "true" class (the so-called *gold standard* or *ground truth*). In addition, we can observe for each object, which class is assigned to it by the classification algorithm (*test outcome*). If we split all the objects in the dataset into four groups according to these two dimensions and count the number of objects in each group, we obtain a *confusion matrix* (Figure 1.6).



Test outcome	**Condition (as determined by "Gold standard")**	
	Positive	***Negative***
Positive	**True Positive**	**False Positive** (Type I error)
Negative	**False Negative** (Type II error)	**True Negative**

Figure 1.6: Confusion matrix for binary classification parameters

In our case, the cells of the confusion matrix correspond to the following objects:

- *True positive* (tp) – the instances of 'good' playlists are labeled as 'good'
- *True negative* (tn) – the instances of 'good' playlists are labeles as 'bad'

- *False positive* (`fp`) – the instances of 'bad' playlists are labeles as 'bad' (also called Type I error)
- *False negative* (`fn`) – the instances of 'bad' playlists are labeles as 'good' (also called Type II error)

Although the confusion matrix neatly summarizes the classification results into four numbers, sometimes it is useful to get a single-number summary.

Accuracy

Classification accuracy denotes the fraction of instances that were classified correctly:

$$Accuracy = \frac{tp + fp}{tp + tn + fp + fn}$$

An accuracy of 100% denotes perfect classification.

Precision

Precision is the probability that a positively-classified playlist is indeed a 'good' playlist:

$$Precision = \frac{tp}{tp + fp}$$

Precision measures the exactness of a classifier. A higher precision means less false positives, while a lower precision means more false positives.

Recall

Recall is the probability that a randomly selected good playlist will be detected as 'good' by the classifier.

$$Recall = \frac{tp}{tp + fn}$$

Recall measures the completness, or sensitivity of a classifier. Higher recall means less false negatives, while lower recall means more false negatives.

1.4 Combinatorial Optimization

Let there be a function $f(x) : A \to \mathbb{R}$. In our case this is a classifier, assigning a "goodness" measure to a playlist x. The problem of finding a good playlist is then equivalent to the problem of *maximizing* the function f. Considering the discrete structure of a playlist object, such problem is commonly referred to as *combinatorial optimization.*

Various techniques of combinatorial optimization exist. In this work we tried a simplistic *randomized search* technique and a more involved *genetic algorithm*.

1.4.1 Randomized Search (RS)

Randomized search is perhaps the simplest approach to combinatorial optimization. The idea is to generate N random values x uniformly and select the one which achieves the largest value of $f(x)$. The only important aspect of this algorithm is in the method of generating random instances. In our case the instances are random playlists with predefined start and end songs. Each playlist is thus generated by randomly sampling k songs (with replacement), then prepending the start song, and appending the end song to the resulting list.

1.4.2 Genetic Algorithms (GA)

A genetic algorithm can be viewed as an extension of the randomized search technique. The main idea is to start with a random set of objects (playlists in our case), then mutate and "cross-breed" the best of them during several populations in order to receive better results in future generations. Genetic algorithms work very well on mixed (continuous and discrete) combinatorial problems, but they tend to be computationally expensive.

To use a genetic algorithm, we must represent the solution to the problem as a *chromosome*. In our case the chromosome is the playlist itself. A population of *chromosomes* will then be "evolving" with various *genetic operations* being applied to them. The most common genetic operations are *mutation* (replacing a random element of the chromosome (i.e. a song) with another random element) and *crossover* (creating a new chromosome by combining the head of one chromosome with the tail of another).

The performance of the genetic algorithm can sometimes be influenced by tuning the frequency at which each of the genetic operations is performed – the *mutation rate* and *crossover rate* parameters, selecting appropritate *population size* and *iteration count*.

Chapter 2

Motivation

In this chapter we present the motivation for the automatic personalized playlist generation problem as well as give an overview of related work.

2.1 The Power of Music

To appreciate the importance of the chosen research topic, we should understand the power and the role of music in our lives. All life long, irrespectively of our desire, we are surrounded by different sounds. One of the most pleasant types of sounds we hear is music. Soon after birth we listen to the first melodies as our mothers hum soothing lullabies to help us sleep. As we get older we discover that music is an inseparable part of the society. We can hear it everywhere: at home, in shops, cars, gyms, theatres, churches, parades. We can not imagine movies, games, sport trainings, etc without music. Thus, in the modern Western culture, the main purpose of music is to entertain or to accompany an entertainment. However, music has more power and influence on our lives than we used to expect.

The main advantage of music compared to other types of media is that it can convey emotional state and sometimes does it more effectively than words or pictures. Music can relax or make us dance, keep spirits up or induce nostalgy. Familiar songs help us recall special moments in our lives, we sing church hymns to help build our spiritual being, and patriotic songs to give us a sense of national identity. Some researches try to prove that music even has healing power. As one can see, music evokes emotions, portrays moods and affects our perceptions. It helps each one of us find our unique social niche, bringing us together with other folks who share similar interests.

Everyone is sensitive to music and everyone can understand music language independent of nationality or education. Music can be regarded as a universal language for the whole humanity. That is why people like music so much despite of the underestimation of its power. With the advent of digital technologies the accesibility of music increased significantly and music is becoming ubiquitously available.

Nowadays, traditional methods of listening to and discovering music, such as radio broadcasts and record stores, are being replaced by personalized ways to hear and learn about music [CVG+08]. In recent years, there has been increasing interest in the automatic generation of playlists, partly due to the broad adoption of digital music and personal digital music players [Oli].

2.2 Music Information Retrieval

The increasing availability of music in digital format, the growing size of personal music libraries and the freely available online music streaming services such as Spotify [spo] or Last.fm [las02] encourage the development of tools for music access, filtering, classification, and retrieval. The interdisciplinary research area that covers all those and other problems connected with digitally available music data is called Music Information Retrieval (MIR).

The concept of information need is central in MIR. Users interact with software to retrieve information that is relevant to their wishes. The fact that audio information is perceived differently by each personality, introduces the complication in how well the users can describe their personal music needs and preferences.

2.2.1 MIR Users

According to the degree of knowledge and involvement, the potential users of MIR systems can be divided in three categories. *Casual users* want to enjoy music, listen and collect the music they like, also discover new music. *Professional users* need music suitable for particular usages related to their activities, which may be media production, advertisements or entertaintment industry. *Music scholars, music theorists, musicologists, and musicians* - typically interested in studying or producing music [Ori06].

2.2.2 MIR Approaches

Research in MIR comprises a broad range of topics including user interfaces for audio collections, vizualization, search, classification, clustering, modelling, segmentation, etc. Different approaches to music processing can be applied to solve the mentioned problems. Depending on the form and format in which musical documents are instantiated and on the dimensions of interest, the techniques used in MIR researches can be divided into three main groups: *metadata analysis* (year, tempo, genre, mood [Lan09]), *listening model* (rating, skips, replays [CTLjH10, ZGMRMF10]) and *content-based learning* (MFCC, flux, rolloff, crossover, average energy [TC02]).

Metadata Analysis

The simplest approach, called *metadata analysis* focuses on metadata provided with a musical composition, such as names of the artist and composition, year of release, genre, *tags*, etc. Despite the fact that metadata-driven systems are quite popular nowadays due to their simplicity and acceptable effectiveness, a combination of metadata analysis with other techniques currently dominates MIR research.

Content-based Analysis

The basic assumption behind *content-based* approaches, the most time and resources consuming, is that metadata are either not suitable, unreliable, or missing. Consequently, an object is described by a set of features that are directly computed from its content. In a case of music the content can be music wave itself or some type of symbolic data – lyrics, notes, chords, etc. As it can be expected, attributes such as pitch, timbre, intensity, rhythm, melodic sequences, instrumentation, and others can be computed from the audio using signal processing techniques. Other features, such as chords, keywords, key chord sequences, are mainly extracted using different types of text algorithms. The analysis of publications on feature extraction shows a clear drift from symbolic toward audio forms [Ori06]. First experiments on content-based audio retrieval were reported in [Foo99] – were focused on automatic genre classification [TC02]. Nowadays, most of the research in MIR is either purely content-based or uses content-based approaches in a combination with other techniques.

Listening Model Analysis

The third approach, *listening model*, does not take into account neither the content nor the metadata, but rather the data about how the object is perceived or described by the users (feedback). This data for analysis contains an information about songs' ratings, skips, replays etc. The listening policy is irregular, but some clues can be inferred from an everyday experience: it follows changes of mood, which are indeed unpredictable, but are rarely totally chaotic. Therefore skipping or replaying behaviour mirrors the interest in the song in particular context.

One of the approaches to detect overal music preference is to analyze feedback data – collaborative filtering (CF). CF is the process of "filtering" information from very large datasets based on data obtained from a collaboration among multiple agents, viewpoints, data sources, etc. CF is widely used for making automatic predictions about the interests of a user by collecting information from many users. In the recent years a lot of data for CF is taken from social networks.

To sum up, freely available audio content continues to become more accessible, listeners require more sophisticated tools to for discovery and organization of new music that they find enjoyable and the main goal of MIR research is to find a way for solving those problems using different information retrieval techniques.

2.3 Basic Parameters of a Music Playlist

This section provides a definition of a playlist and discusses the subjective factors, influencing playlist preferences.

2.3.1 Playlist Definition

Although the concept of a *playlist* is intuitively clear, it is surprising that there is no “standardized” definition of a playlist (with respect to its content, structure and purpose) in the related literature: some authors suggest particular, but differing, definitions [CBF06, Lan09, RBBC], some rely on the reader’s own interpretation of the term [AH06, RBH05, Oli], some provide latent explanation, such as “radio playlists” being the ground truth [MEDL09] or attempt to construct a sequence of songs with an inherent order defined by smooth transitions between neighbours [BCT08]. For the purposes of this work, a playlist may be defined as a finite sequence of songs which is played as a complete set [RBBC].

Some playlists are created for personal use by oneself or a few close friends, primarily as background for another activity (e.g. music to listen to while traveling, studying, or exercising at the gym). A playlists may be created to reflect a particular mood or emotion of the creator, such as depression, angst, or cheerfulness.

Playlists are often confused with DJ-created mixes, which involve cutting, changing or merging of the original compositions into basically a new piece of art. The main controversies of playlist definition are related to such aspects as existence of a principle and importance of the songs order. Cunningham et al. [CBF06] make the distinction between playlists and “mixes”. So, a playlist is a collection of songs grouped together under a particular principle. The principle could be objective, such as “rock songs from the 70’s” or subjective, such as “songs that remind me of Melanie” [Lan09].

2.3.2 Songs, Order, Length

There are three main aspects to a playlist that are of interest: the songs in the sequence, the order in which these songs occur, and the length of the playlist [vos].

Songs

People do not always know which specific songs they want to hear: they either do not remember the title or the artist or even do not know them at all, but they remember their emotions, mood connected with those songs. However, it is usually hard to explicitly express the characteristics of the songs they want to hear in a certain situation. It is therefore essential that each song contained within the playlist satisfies the expectations of the listener. These expectations are formed based upon the listener's mood, which in turn is influenced by the environment [RBBC].

Order

The order in which the songs are played provides the playlist with a sense of balance which a randomly generated playlist can not produce. In addition it can provide a sense of progression such as, a playlist is changing from slow to fast or from loud to soft. [RBBC]. In other words, not only the grouping of the pieces is of importance, but also in which order they were listened to.

It is hard, however, to find an explanation as to what are the rules to good ordering. "It's been said that there is only one rule. . . There are no rules" [CBF06]. S. J. Cunningham et al. performed an extensive study analyzing typical suggestions given by the user. The most important were: a) no more than two songs from the same artist or genre in a row, b) consecutive songs should have complementary styles or sounds so that the mix does "not clash one song up against another" – the first song should be good, but not the best, d) particular care should be taken in selecting the final song, as "they will remember the last song easily, e) trying to avoid boring repetition and excessive change: not too many slow songs, hard rock, sad songs together. To sum up, overall, people prefer playlists containing variable songs – of different artist, genre, tempo, mood, but at the same time not too random. One of the most valuable properties of a good playlist, confirmed by expert DJs is that there should be 3-4 songs of the same genre, tempo etc in a row, after which those parameters should change.

Number of Songs, Length

The number of songs in a playlist primarily determines the time duration of the playlist. An understanding of the length of a playlist is important, as song ordering and song balancing of the playlist is unachievable otherwise [RBBC].

The length of a piece is important also in the sense that it is quite rare that one chooses two pieces of very different lengths in the same collection. The length of a piece in fact can be a clue to determine some other properties. Longer pieces tend to be more serious and involving, whilst the short ones should be more informal [AH06].

2.3.3 Musical Taste, Preferences and Listening Habits

In the situation considered in this work, playlists are not meant for music discovery. All the individual songs are known by the user and assumed to be liked by the listener. Songs in the music collection used for a playlist generation should mirror the users' musical taste. The notion of musical taste can be defined as a person's slowly evolving long-term commitment to a particular music idiom [Pau02]. Its development is assumed to depend on the cultural environment, the major consensus [Fur88], peer approval, musical training [Ger82], age as an indirect factor [HS89, Rub88] a particular period or social context [AH06] and other personal characteristics.

Hence, the music collection represents the user's taste in long-term, and a playlist – the current preference. Moreover, each song in the music library answers the global information need of the owner. The music preference is the one that interests us for predicting the future playlists.

The factors that can influence the user's musical preference are very complex, and they are both subjective and objective. An incomplete list of these factors could be as follows [AH06]:

- education,
- environmental effects, social context (place, people around),
- factors of the moment (mood, attention level, etc),
- social and cultural influences (ethnical issues, membership of social/ subculture groups),
- individual differences (age, gender, introversion/extraversion, independence, sensitivity, anxiety),
- real world applications (clinical and therapeutic uses of music, consumer behavior and music education).

In addition, music preference changes over time. More regular criteria that can be used for personalized playlist generation are listening habits. Consequently, if a piece is played often quite recently, it is expectable that it will be played in the near future. If a group of pieces is played together a number of times recently, it is quite likely for the whole group to be played in the same order in the near future as well [AH06]. If playlists contain songs of different genres seems like a person likes variability etc. it is important to accept these and other observations while modelling an automatic personalized playlist generator.

2.3.4 Criteria for 'Good' and 'Bad' Playlists

As we mentioned before, the playlist generation task is a user's information need satisfaction problem. The general wisdom for information retrieval system design is that users wish to achieve their information goals efficiently and with a minimum of interaction with the system [CBF06].

Category	No	%
Artist/Genre/Style	29	25.2%
Event or Activity	29	25.2%
Romance	22	19.1%
Message or Story	19	16.5%
Mood	19	16.5%
Challenge or Puzzle	12	10.4%
Orchestration	8	7.0%
Characteristics of Recipient	7	6.1%
Cultural References	7	6.1%
Other	3	2.6%

Table 2.1: Categorization of organizing principles for mix requests [artes]. As many proposed mixes had more than one theme, the percentages total to more than 100%.

A good playlist has a central theme or organizing principle: it tells a story, shares a mood, "gives a perspective into the individual songs that you would not have had without seeing them in that idea" [CBF06]. Table 2.1 presents a categorization of the organizing principles for 115 mix requests posted to the Art of the Mix Discussion Forum [artes].

In the research of music seeking criteria [Ada08] experiment participants mentioned that emotion, mood, and listening context are important attributes for music seeking.

2.4 Approaches to Playlist Construction

There are three main parameters to each existing playlist generation method: the *level of automation*, the *time needed to construct a first good playlist*, the way of *modelling user's preference.*

2.4.1 Level of Automation

By the level of automation, all methods can be divided into three main categories: made manually, automatically or manually with support (semi-automatically). The easiest and the most effective way for constructing a playlist is to do it manually. However, this is a time-consuming task that, at the same time, allows a certain degree of freedom. However, the huge size of music collections nowadays exceeds a person's ability to recall which compositions comply best to the current mood or situation. Semiautomatical playlist generator is used to help users to orient in their music collections by vizualizing or by proposing some variants for the next song in a sequence satisfying some predefined constraints.

If a user just needs background music for studying, (s)he is not very demanding to the exact choice of the songs. None the less, the transition from Shakira to Bach or Metallica provided by uniformly random shuffle is probably still not desirable. Moreover, many social contexts, such as, for example, being at work, or driving a car, do not allow to waste too much time on hand-picking music. In these cases automated playlist generation can be used.

Earlier music retrieval work regarding playlists has focused on 'automatic playlist generation' [CTLH10, RBBC, FRCJ08], and to a lesser extent on supporting users in more easily constructing their own playlists [CBF06]. In our work we focus on creating an algorithm for automated personalized playlist generation (APPG), primarily targeting casual users rather than professional DJs, who are interested in generating "good enough" background playlists by providing just a couple of sample playlists.

2.4.2 Time Resources

Another parameter of playlist construction task is the time needed to construct a first good playlist. Some algorithms do it on the fly [BK09, age10] and the level of playlist's goodness is constant if muisic library and constraints stay the same, some methods based on "learning" algorithms provide better and better results the more they are used and rated – such approach usually have big problems with first playlists. Moreover, the type of data used for playlist construction defines the time needed to process it: metadata (artist name, song title, year of release, genre etc) [GV05], skipping behaviour [CTLjH10] and collaborative filtering [BPK] based approaches usually require less time to process the data than algorithm that use audio feature extraction [BCT08, TPW05] to solve the problem.

2.4.3 Modelling User's Preference

Many different approaches are possible to automatically generate playlists from a music collection. Besides the basic random (shuffle) method a *constraint satisfaction*, *similarity-based* and *landscape* methods can be identified [CVER07]. *Constraint satisfaction*-based playlist composition uses several constraints including the desires of users [Pau02, AP02, RBBC]. Creation of playlists satisfying user constraints can be based on rich metadata [RBBC], temporal order of the playlists (e.g. rising tempo, change of genre), notions of audio similarity [CTLH10]. *Similarity*-based approaches try to guess the users' interest and mood and propose lists of similar sounding characteristics using seed songs [MEDL09, BCT08], observed user interactions (e.g. collaborative filtering) [CVER07] or some kind of metadata [Pau02, GPB$^+$01, RBH05, FRCJ08, TPW05]. Cunningham et al. confirmed the relevance of the seed-song approach analyzing the user study reports and stating the fact that 50 percent of requests for help in creating playlists included a song as an example. The problem of seed-based creation of playlists is that the result is often too uniform and too dependent on music database size [BCT08]. New methods have recently been proposed that are based on a *music landscape*. The playlists to be composed are specified, either with an implicit path like in [CVER07] or an explicit path that goes through the landscape. Few authors report about generating playlists with an inherent sequential order.

Chapter 3

The Algorithm for APPG

In this chapter, we describe an automatic personalized playlist generation (APPG) algorithm that generates music playlists according to the taste of a concrete user learned on a set of provided sample 'good' and 'bad' playlists. First, we set constraints and define a personalized classifier for distinguishing 'good' and 'bad' playlists. Next, we describe how randomized and genetic algorithms can be used to solve the optimization problem.

3.1 Algorithm Description

The main steps of the algorithm are shown on the scheme below (Figure 3.1). The input for the algorithm should be a large collection of music pieces and a set of playlists rated by the user as 'good' or 'bad'. The output, in turn, should give a list containing some of the music files from the dataset. The criterion according to which the lists should be assembled is to maximize user's satisfaction.

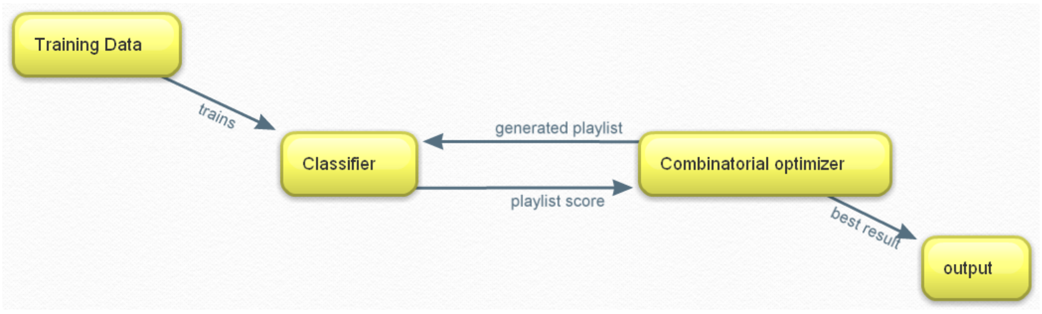


Figure 3.1: The main steps of the algorithm are: a) train a classifier to distinguish good and bad playlists b) use the classifier to measure the goodness of a generated playlist c) apply optimization algorithm to find the best playlist.

3.2 Playlist Constraints

We set some constraints to our dataset and playlists. Firstly, the dataset should not contain duplicates of music files and should not be a collection of one genre/style only.

Secondly, the generated playlist must satisfy the following criteria: a) the length of the playlist is n, b) variability of styles and artists (i.e. not all songs must be of the same style/artist), c) start and end songs are chosen by the user, d) no duplicate songs in the playlist.

The reason for disallowing playlists of the same style and artist is mainly the following. It is quite simple to construct a coherent background playlist of songs just by choosing them randomly from a subset of some style or a single artist. To exclude this trivial case and motivate a more involved algorithm, we exclude this possibility from our algorithm.

3.3 Dataset Construction

The dataset of playlists was constructed manually from a personal music collection of 500 MP3 files of various styles and genres (manually labeled). To reduce the size of data and save on computation, we decreased the sample rate of all compositions from 44 kHz down to 8 kHz and selected a 10 second piece from the middle part of each song. The acoustic characteristics of each song were computed from this downsampled middle-piece.

Next, we manually constructed 50 playlists, each of length $n = 10$ songs and satisfying the abovementioned constraints, such that 25 would provide examples of 'good' playlists and 25 would be 'bad' playlists.

3.4 Learning Subjective Playlist Preference

One of the main parts of our algorithm is a playlist goodness classifier. The description is provided in the following sections.

3.4.1 Song and Playlist Representation

Most classical machine learning techniques require data in the form of vectors. As described in Chapter 1, we represent all songs in the musical collection in terms of their sounding characteristics such as `centroid`, `flatness`, `mfcc`, `flux` and `irregularity`, which can be computed from the audio signal. Therefore, each song i gets assigned a finite feature vector $s_i = (v_{i1}, \ldots, v_{iK})$.

Each playlist is a sequence of songs $p = (s_1, s_2, \ldots, s_\ell)$. Similarly to the song representation, we aim at representing the playlist using a finite set of features, where the main idea is to capture the type of transitions between neighbouring songs (smooth or sharp transitions).

Consequently, we propose to define the following playlist feature representation:

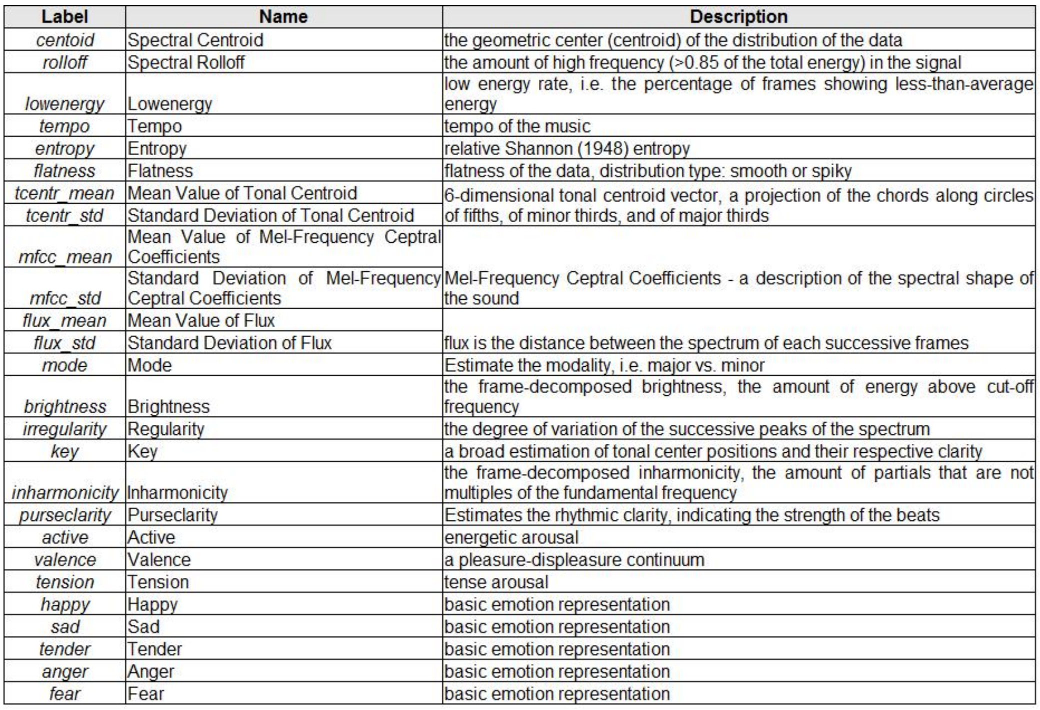
$$p = (v_1, \dots, v_K),$$

where each v_k is calculated as the normalized sum of absolute differences in the corresponding song features:

$$v_k = \frac{\sum_{i=1}^{n-1} |v_{i+1,k} - v_{i,k}|}{\ell |v_{n,k} - v_{1,k}|},$$

where $v_{i,k}$ – is a k-th feature value of the i-th song in the playlist.

Alltogether we had constructed an initial feature vector consisting of 26 elements representing each playlist(Figure 3.2). It included *classical spectral* features, such as mfcc, rolloff, lowenergy, centroid, as well as *emotional* features: sadness, tender, tension, anger, happiness, etc (the full list corresponds to features supported by the MIRtoolbox [mir]).



Label	Name	Description
centoid	Spectral Centroid	the geometric center (centroid) of the distribution of the data
rolloff	Spectral Rolloff	the amount of high frequency (>0.85 of the total energy) in the signal
lowenergy	Lowenergy	low energy rate, i.e. the percentage of frames showing less-than-average energy
tempo	Tempo	tempo of the music
entropy	Entropy	relative Shannon (1948) entropy
flatness	Flatness	flatness of the data, distribution type: smooth or spiky
tcentr_mean	Mean Value of Tonal Centroid	6-dimensional tonal centroid vector, a projection of the chords along circles of fifths, of minor thirds, and of major thirds
tcentr_std	Standard Deviation of Tonal Centroid	
mfcc_mean	Mean Value of Mel-Frequency Ceptral Coefficients	Mel-Frequency Ceptral Coefficients - a description of the spectral shape of the sound
mfcc_std	Standard Deviation of Mel-Frequency Ceptral Coefficients	
flux_mean	Mean Value of Flux	flux is the distance between the spectrum of each successive frames
flux_std	Standard Deviation of Flux	
mode	Mode	Estimate the modality, i.e. major vs. minor
brightness	Brightness	the frame-decomposed brightness, the amount of energy above cut-off frequency
irregularity	Regularity	the degree of variation of the successive peaks of the spectrum
key	Key	a broad estimation of tonal center positions and their respective clarity
inharmonicity	Inharmonicity	the frame-decomposed inharmonicity, the amount of partials that are not multiples of the fundamental frequency
purseclarity	Purseclarity	Estimates the rhythmic clarity, indicating the strength of the beats
active	Active	energetic arousal
valence	Valence	a pleasure-displeasure continuum
tension	Tension	tense arousal
happy	Happy	basic emotion representation
sad	Sad	basic emotion representation
tender	Tender	basic emotion representation
anger	Anger	basic emotion representation
fear	Fear	basic emotion representation

Figure 3.2: MIRtoolbox audio content-based features tested in our experiments.

Finally, when the feature set is fixed, all features are extracted and the data collected, all information could be represented as a matrix with each row corresponding to a playlist: each column corresponding to a feature and each cell assigning the feature value to a given playlist in turn (Figure 3.4). The last column is the label of the playlist category.

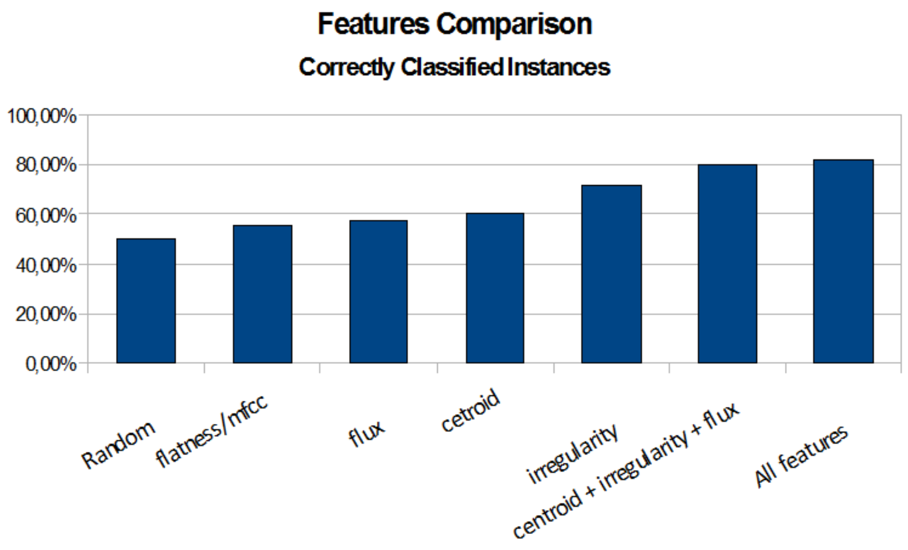
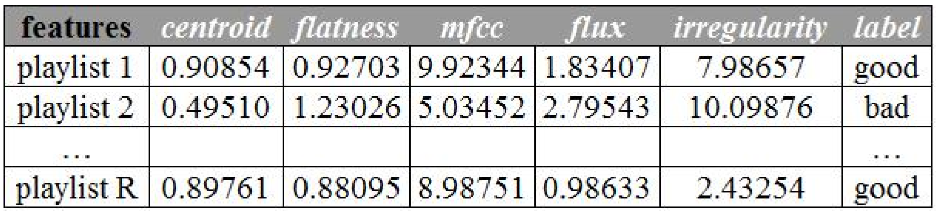


Figure 3.3: Percentage of correctly classified compositions obtained by 10-fold evaluation strategy on the dataset of 50 playlists of two classes (‘good’ and ‘bad’) using different algorithms. The feature set contains 5 spectral features described in Chapter 1



features	*centroid*	*flatness*	*mfcc*	*flux*	*irregularity*	*label*
playlist 1	0.90854	0.92703	9.92344	1.83407	7.98657	good
playlist 2	0.49510	1.23026	5.03452	2.79543	10.09876	bad
...						...
playlist R	0.89761	0.88095	8.98751	0.98633	2.43254	good

Figure 3.4: Features representation: in a supervised learning scenario each training example (musicplaylist) consists of an object to be classified presented by its features, as well as the correct category (style) to which it should be assigned.

3.4.2 The Choice of Classifier

Once we collect a dataset of labeled playlists and represet all the playlists as features, we are free to apply standard machine learning techniques, which are otherwise independent of the specific application area [TC02].

In the present paper we selected the Naïve Bayes algorithm for automatic playlist goodness classification mostly due to its conceptual simplicity and comparably good effciency. Our preliminary studies have confirmed that its performance is better than other algorithms, such as SMO, J48, NBTree (Figure 3.5).

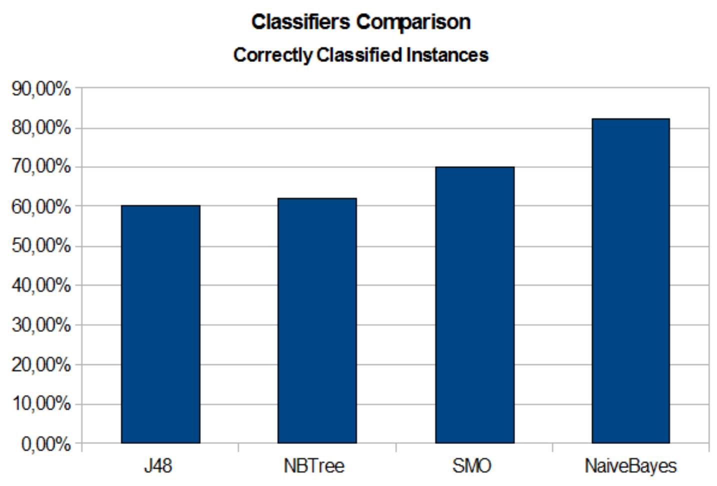


Figure 3.5: Percentage of correctly classified playlists obtained by 10-fold evaluation strategy on the dataset of 50 playlists of two categories (‘good’ and ‘bad’) using different. The feature set contains features of music surface proposed in the paper

3.5 Automatic Playlist Generators

Having trained Naïve Bayes classification algorithm we are able to assign goodness probability to any other playlist. This makes it possible to view the playlist generation problem as a combinatorial optimization task. We use the randomized search technique and the genetic algorithm to perform this optimization and compare our results with the baseline “shuffle” approach.

Shuffle

The most trivial approach to playlist generation is generating playlists purely randomly. In our experiments we used this approach as the baseline to compare with the results of the more clever algorithms.

Randomized Search (RS)

For random search, 1000 different playlists were randomly generated and evaluated by the classifier function. One of the playlists that had a goodness score > 0.995 was returned to the user. The distribution of the evaluation function values (Figure 3.6) shows that it is, in fact, fairly “easy” to generate a good playlist. We later found out that most of the “good” playlists were highly homogeneous in their style. When more stringent constraints were used, the proportion of high-quality candidates within a random population drops drastically.

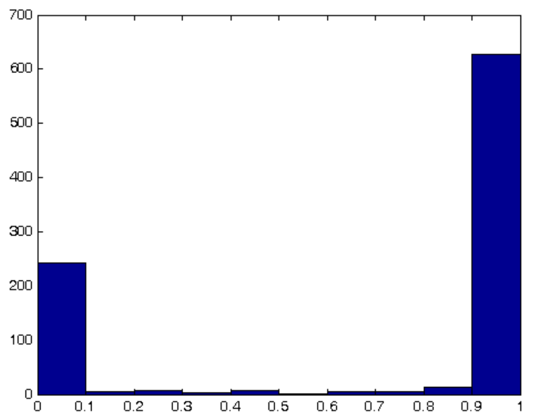


Figure 3.6: Distribution of playlist goodness probability. 1000 playlists generated by randomized algorithm.

Genetic Algorithm (GA)

In the case of automatic playlist generation problem, the chromosomes are represented by a sequence of songs identical numbers. Each chromosome will be also checked/tested for the goodness probability by the playlist classification algorithm presented above making clear how good it is at solving the problem at hand and assign a fitness score accordingly. The fitness score is a measure of how good that chromosome is at solving the problem. In our case it is a playlist goodness probability returned by the Naïve Bayes classifier.

Next step is to select two members from the current population choosing members from the population of chromosomes in a way that is proportional to their fitness – Roulette wheel method. It does not guarantee that the fittest member goes through to the next generation, merely that it has a very good chance of doing so. In our case the algorithm looks as follows:

1. Randomly generate an initial playlist population $V(0)$
2. Compute and save the goodness $u(v)$ for each individual (playlist) v in the current population $V(t)$
3. Define selection probabilities $p(v)$ for each individual (playlist) m in $V(t)$ so that $p(v)$ is proportional to $u(v)$
4. Generate $V(t+1)$ by probabilistically selecting individuals from $V(t)$ to produce offspring via genetic operators
5. Repeat step 2 until satisfying solution is obtained.

We use the `PyEvolve` library [Per09] and its default GA parameters: evolve for 100 generations with a population size of 80 individuals, the mutation rate of 2% and a crossover rate of 80%.

3.6 Implementation details

In the implementation we used the following packages:

Data processing and feature extraction was done by using MIRtoolbox [mir] – a toolbox for music information retrieval that offers an integrated set of functions written in MATLAB [mat], dedicated to the extraction from audio files of musical features such as tonality, rhythm, structures, etc. We tried 26 different acoustic features that potentially could help to clarify an individual playlist taste of the user.

As MIRtoolbox [mir] prefers to read sound files in WAV format and does it much faster than others, we used the Sound eXchange [sox] sound converter software to convert MP3 files to WAV.

For initial analysis of playlist feature vectors we used Weka [HDW94] – a suite of machine learning software which supports several standard data mining tasks, such as clustering, classifiation, visualization and feature selection that we have used in our work.

To implement the playlist generation algorithm we used Python packages *GNB* and *pyevolve*. *GNB* is an implementation of the Gaussian Naïve Bayes Classifier that allows to train a classifier and test the input playlists for the probability of being 'good'. *Pyevolve* was used to run the genetic algorithm.

Chapter 4

Algorithm Evaluation

In this chapter we evaluate the performance of the algorithm and present the results. For evaluation we used the dataset of manually constructed and labeled playlists described in the previous chapter.

4.1 Classification Performance

To illustrate the usefulness of each separate feature, we shall illustrate its ability to discriminate 'good' and 'bad' playlists that we used in our experiments.

The overall performance of the Naïve Bayes classification algorithm for playlists is presented in the confusion matrix in Table 4.1. Those results correspond to accuracy 82%, recall 90% and precision 44%.

Table 4.1: Confusion Matrix

a	b	<–classified as
18	7	a = good
2	23	b = bad

To sum up, the classification algorithm for 'good' and 'bad' playlists is quite precise for being used as a part of automatic playlist generator.

4.2 Generation Performance

In our work we tried a genetic generation algorithm and compared its result with a simpler randomized search and a baseline shuffle algorithms.

4.2.1 User Evaluation

In solving the APPG problem one of the most important tests that can be done is user evaluation, i.e. getting a feedback of the potentioal user of our algorithm. The test itself and its results are described below.

Test Description

For the user test, the same music collection was used as for the other tests that are described in this chapter. From the collection of 500 music files, 150 different playlists were automatically generated using three algorithms: shuffle, randomized search (RS) and genetic (GA).

Playlists were generated in triples, so that each playlist in the triple was generated using a separate algorithm. The task of the user was to choose the one of the three that seems better than the other two. The test was blind – the user did not know which of the three playlists was generated by which algorithm.

As the work is about the automatic *personalized* playlist generation algorithm and the original dataset was created by the thesis author, all tests were also done by the author of the thesis. Thus, the present user evaluation assessed the subjective quality of music playlists.

Results

Playlist quality was measured by the proportion of the playlists that were chosen by the user as the best ones out of the provided couple generated by the same constraints and the same start and end seed song. As a result, playlists generated by the RS were judged to be of higher quality than those generated by the GA algorithm and shuffle method in the case of longer 10-songs playlists. The same situation was detected for shorter playlists of 5 songs. In both cases the RS algorithm outperformed without significant advantage - just 4% of more user choices (Figure 4.2).

Type of test	Shuffle (SH)	Randomized (RS)	Genetic (GA)
10-song playlists	10	19	21
5-song playlists	12	18	20

Table 4.2: Algorithm quality was measured by the proportion of the preferred playlists. As the work is about the automatic personalized playlist generation algorithm, all tests were done by the authors of the thesis - same people rated the test dataset.

So we can make an assumption that the quality of playlist generator is more or less the same for playlists containing 5-10 compositions. In turn, comparing the playlists constructed by Randomized Search (RS) and Shuffle, we see the positive impact of audio content-based features into the APPG algorithm (Figure 4.1).

4.2.2 Run Time Evaluation

In this section the genetic playlist generation algorithm is compared to the randomized approach in a sence of run time. We measured how long does it

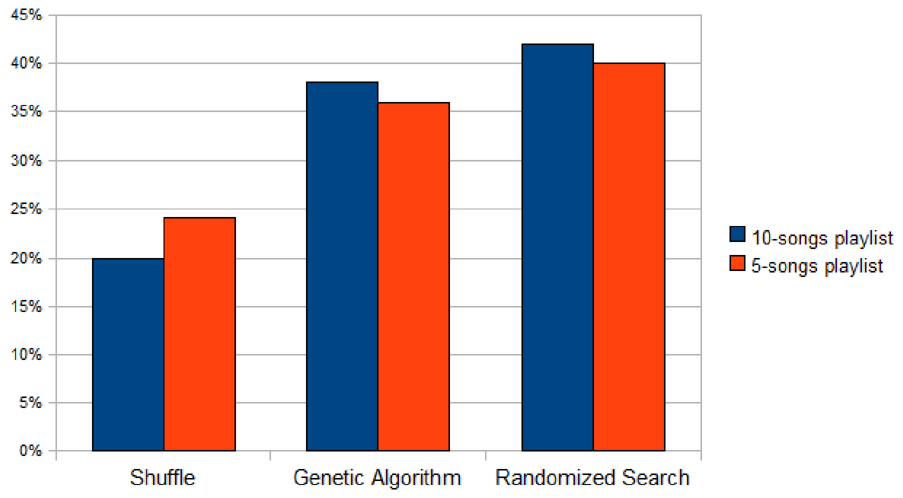


Figure 4.1: The result of the following experiment: for each couple of start and end seed songs three playlists were automatically generated using three algorithms: shuffle, randomized search (RS) and genetic (GA). The user chose (blindly) the one of the three that seems better than the other two.

take to generate playlists of different length with given constraints. The algorithms were tested on the machine with following parameters: Intel Core^{TM}2 Duo CPU T7300 2 GHz 2GB RAM, 32-bit OS.

As we see from the table (Figure 4.3) – the time needed for genetic algorithm is much greater than for randomized.

Genetic algorithm (GA)	Randomized Search (RS)
18,634 s	$<$ 1 s
618,910 s	$<$ 1 s
155,642 s	$<$ 1 s
409,441 s	$<$ 1 s
351,283 s	$<$ 1 s
430,282 s	$<$ 1 s
580,342 s	$<$ 1 s
9,406 s	$<$1 s
320,501 s	$<$ 1 s
98,234 s	$<$ 1 s
mean:	mean:
299,269 s	$<$ 1 s

Table 4.3: Run Time of the Algorithms measured in seconds (s).

With a known dataset it takes just a couple of seconds to generate a random set of playlists, test them by our category classification algorithm and choose some of the best of them to propose to the user.

Summary

In this thesis we presented a study of an approach to automated personalized playlist generation. Besides a brief overview of the theoretical background, we documented our approach, the experiments we performed and the results we obtained. In the practical part of our work we extracted features proposed by MIRtoolbox and mostly defined in the paper of [TC02] and evaluated the work on our data set using different approaches. Afterwards, we fixed the classifier, made some proposals for the feature set, tested them, constructed a feature set of 5 elements that classify playlists for 'good' and 'bad' with the accuracy of 82% that is much better than random (50 %). Finally, we tried diferent playlist generation methods and came to a conclusion that Randomized Search is an optimal solution for a given problem by both parameters: quality and run time.

There are several directions for future research. One of the obvious is to continue work on the improvement of 'good' and 'bad' playlist classification algorithms, searching for new valuable features that could be extracted from the audio signal, as well as adding new metadata features and information for the collaborative filtering. From the other point of view, we could learn human playlist generation strategies and constraints and accept them in the generation algorithm. This could provide an easier search for a particular set of music compositions that will help people to construct their background playlists automatically according to their music preferences and playlist taste with minimal time consumption. Moreover, we believe that such an algorithm could simplify the work of professional DJs; they will be able to get intelligent proposals of music compositions for the next track by their artificial characteristics.

Automaatse personaliseeritud esitusloendi generaator

Magistritöö(30 EAP)

Anastassia Semjonova

Resümee

Tänapäeval hoitakse muusikat peamiselt digitaalvormis. Viimasel ajal suure digitaalmuusika ning isikliku muusika mängijate kättesaadavuse tõttu on üha rohkem huvi automaatse põlvkonna muusika esitusloendite vastu. Muusika esitusloend või pleilist võib olla defineeritud nagu lõplik muusika kompositsioonide järjestus, mis on mängitud ning tajutud nagu tervik kogum.

Kõige lihtsam viis on konstrueerida esitusloendi käsitsi. Kuid, see on väga aeganõudev ülesanne, mis aga samal ajal lubab mingil määral vabadust. Samuti, personaalsed muusika kogumid on tänapäeval nii suured, et paljud inimesed ei suuda kiiresti valida muusika faile, mis vastavad nende tujule antud hetkel ja olukorral. Ka paljud sotsiaalsed kontekstid, nagu töökohal või autos, ei anna võimalust käsitsi muusikat valida. Seepärast käesolevas magistriöös me fokuseerime automaatse personaliseeritud taustamuusika esitusloendi generaatoril, mille peamisteks kasutajateks on mitte professionaalsed DJ-d. Kasutaja määrab pleilisti pikkust ning esimest ja viimast kompositsiooni. Meie algoritm omakorda täidab järjestust vastavalt kasutaja maitsele, mis on kavandatud eelnevalt märgistatud 'hea' ja 'halva' esitusloendite järgi. Kõikide meie esitusloendite jaoks on keelatud laulja ning kompositiooni kordumine. Käesolevas magistritöös on esitatud automaatse personaliseeritud esitusloendi generaatori probleemi lähenemisviiside uuring. Lisaks teoreetilise tausta lühiülevaatele me dokumenteerisime oma lähenemist: meie poolt tehtud katsed ning nende tulemused.

Meie algoritm koosneb kahest põhiosast: esitusloendi hindamisfunktsiooni konstrueerimine ning pleilisti genereerimisstrateegia valik. Esimese ülesande lahendamiseks on valitud Naïve Bayes klassifitseerija ning 5-elemendiline MIRtoolbox tööristakasti poolt kavandatud audio sisupõhiste attribuutide vektor, mis klassiitseerivad pleilisti heaks või halvaks 82% täpsusega - palju parem kui juhuslik klassifitseerija (50%). Teise probleemi lahendamiseks proovisime kolm genereerimisalgoritmi: lohistus (Shuffle), randomiseeritud otsing (Randomized Search) ning geneetiline algoritm (Genetic Algorithm).

Vastavalt katsete tulemustele kõige paremini ja kiiremini töötab randomiseeritud otsingu algoritm. Kõik katsed on tehtud 5 ning 10 elemendilistel esitusloenditel.

Kokkuvõttes, oleme arendanud automatiseeritud personaliseeritud esindusloendi generaatori algoritmi, mis vastavalt meie hinnangutele vastab ka kasutaja ootustele rohkem, kui juhuslikud lohistajad. Algoritmi võib kasutada keerulisema pleilistide konstrueerimiseks.

On olemas mitu suunda edasisteks uuringuteks. Üks variant on jätkata tööd 'hea' ja 'halva' esitusloendite klassifitseerija parandamise valdkonnas, uurides milliseid atribuute ning klassifikaatorit kasutada. Samal ajal võib uurida, kuidas automaatset personaliseeritud muusika pleilisti genereerijat paremaks muuta: testida teisi genereerimisstrateegiaid ning algoritme. Väga kasulik oleks ka mingi üldise andmekogumiku koostamine, mis oleks võimalik testimiseks kasutada.

References

[Ada08] Matt Adamo. Looking with their ears: The information seeking behavior of music consumers. 2008.

[age10] Moodagent - a commercial service from syntonetic that combines digital signal processing and ai techniques to create music profiles that incorporate characteristics such as mood, emotion, genre, style, instrument, vocals, orchestration, production, and beat/tempo, 2010.

[AH06] Andreja Andric and Goffredo Haus. Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools Appl.*, 29:127–151, June 2006.

[AP02] Jean-Julien Aucouturier and Francois Pachet. Scaling up music playlist generation, 2002.

[artes] User-friendly and cost-free, simply promoting a communal place (webpage) to talk about some of the best mixes.

[BCT08] Juan Pablo Bello, Elaine Chew, and Douglas Turnbull, editors. *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, 2008.

[Ben06] Dave Benson. *Music: A Mathematical Offering.* 2006.

[BK09] Klaas Bosteels and Etienne E. Kerre. A fuzzy framework for defining dynamic playlist generation heuristics. *Fuzzy Sets and Systems*, 160(23):3342 – 3358, 2009. Theme: Computer Science.

[BPK] Klaas Bosteels, Elias Pampalk, and Etienne E. Kerre. Evaluating and analysing dynamic playlist generation heuristics using radio logs and fuzzy set theory.

[CBF06] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. 'more of an art than a science': Supporting the creation of playlists and mixes. In *ISMIR*, pages 240–245, 2006.

[CTLH10] Chung-Yi Chi, Richard Tzong-Han Tsai, Jeng-You Lai, and Jane Yung-jen Hsu. A reinforcement learning approach to emotion-based automatic playlist generation. In *Proceedings of the 2010 Conference on Technologies and Applications of Artificial Intelligence (TAAI2010)*, November 2010.

[CTLjH10] Chung-Yi Chi, Richard Tzong-Han Tsai, Jeng-You Lai, and Jane Yung jen Hsu. A reinforcement learning approach to emotion-based automatic playlist generation. *Technologies and Applications of Artificial Intelligence, International Conference on*, 0c:60–65, 2010.

[CVER07] Michel Crampes, Jean Villerd, Andrew Emery, and Sylvie Ranwez. Automatic playlist composition in a dynamic music landscape. In *Proceedings of the 2007 international workshop on Semantically aware document processing and indexing*, SADPI '07, pages 15–20, New York, NY, USA, 2007. ACM.

[CVG$^+$08] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges, 2008.

[ear] Physics tutorial - http://www.ddart.net/science/physics/ (2009).

[Foo99] Jonathan Foote. An overview of audio information retrieval, 1999.

[FRCJ08] B. Fields, C. Rhodes, M. Casey, and K. Jacobson. Social Playlists and Bottleneck Measurements: Exploiting Musician Social Graphs Using Content-Based Dissimilarity and Pairwise Maximum Flow Values. In *ISMIR*, pages 559–564, 2008.

[ft] Raven 1.2 user‘s manual - http://www.birds.cornell.edu/brp/pdf-documents/appa-digitalsound.pdf.

[Fur88] Duke R. A. Furman, C. E. The effect of overtly categorizing music on preference for popular music styles. 1988.

[Ger82] J.M. Geringer. Verbal and operant music listening in relationship to age and musical training. *Psychology of music (special issue)*, pages 47–50, 1982.

[GPB$^+$01] For Automatically Generating, John C. Platt, Christopher J. C. Burges, Steven Swenson, Christopher Weare, and Alice

Zheng. Learning a gaussian process prior. In *In Advances in Neural Information Processing Systems*, pages 1425–1432. MIT Press, 2001.

[GV05] Rob Gulik and Fabio Vignoli. Visual playlist generation on the artist map. In *in Proc. of the ISMIR Intl. Conf. on Music Information Retrieval*, pages 520–523, 2005.

[HDW94] G. Holmes, A. Donkin, and I. H. Witten. Weka: a machine learning workbench. pages 357–361, August 1994.

[HS89] M.B. Holbrook and R.M. Schindler. Some exploratory findings on the development of musical tastes. *Journal of Consumer Research*, 16 (June):119–124, 1989.

[Jen04] Kristoffer Jensen. Kristoffer Jensen Irregularities, Noise and Random Fluctuations in Musical Sounds 3.1. Introduction. 2, 2004.

[Lan09] Luke Barrington; Reid Oda; Gert Lanckriet. Smarter than genius? human evaluation of music recommender systems. In *In 10th International Society for Music Information Retrieval Conference*, 2009.

[las02] Music website, founded in the united kingdom - http://last.fm, 2002.

[mat] Matlab - the language of technical computing - http://www.mathworks.com/products/matlab.

[MEDL09] F. Maillet, D. Eck, G. Desjardins, and P. Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR 2009)*, 2009.

[mir] Mirtoolbox - an innovative environment, on top of matlab, for music and audio analysis - http://users.jyu.fi/ lartillo/mirtoolbox/.

[Oli] Nuria Oliver. Papa: Physiology and purpose-aware automatic playlist generation.

[Ori06] Nicola Orio. Music retrieval: a tutorial and review. *Found. Trends Inf. Retr.*, 1:1–96, January 2006.

[Pau02] Steffen Pauws. Pats: Realization and user evaluation of an automatic playlist generator. In *In ISMIR*, pages 222–230, 2002.

[Per09] Christian S. Perone. Pyevolve: a python open-source framework for genetic algorithms. *SIGEVOlution*, 4:12–20, November 2009.

[RBBC] Gordon Reynolds, Dan Barry, Ted Burke, and Eugene Coyle. Towards a personal automatic music playlist generation algorithm: The need for contextual information.

[RBH05] R. Ragno, C.J.C. Burges, and C. Herley. Inferring similarity between music objects with application to playlist generation. In *Proc. 7th ACM SIGMM international workshop on Multimedia information retrieva*, 2005.

[Rub88] Rahhal T. A. & Poon L. W. Rubin, D. C. Things learned in early adulthood are remembered best. *Memory & Cognition*, 26:3–19, 1988.

[sox] Sox - sound converter software - http://sox.sourceforge.net/.

[spo] Swedish drm-based music streaming service offering streaming of selected music from a range of major and independent record labels, including sony, emi, warner music group, the orchard, and universal - http://www.spotify.com.

[TC02] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. on Speech and Audio Processing*, 10(5):293–302, 2002.

[TPW05] Elias Pampalk Tim, Tim Pohle, and Gerhard Widmer. Dynamic playlist generation based on skipping behavior. In *Proc. of the 6th ISMIR Conference*, pages 634–637, 2005.

[vos] Local search for automatic playlist generation, m.p.h. vossen.

[Wit05] Eibe Frank Ian H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques.* 2005.

[ZGMRMF10] Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Natasa Milic-Frayling. Statistical models of music-listening sessions in social media. In *19th International World Wide Web Conference (WWW)*, 2010.

Appendices

Appendix A. Program code (on a compact disc)