Automatic Movie Genre Classification Based On Musical Descriptors

Boix Luna, Pau

Curs 2013-2014

Director: EMILIA GÓMEZ

GRAU EN ENGINYERIA EN SISTEMES AUDIOVISUALS

Treball de Fi de Grau
Acknowledgements

First, I would like to thank my supervisor, Emilia Gómez, for encouraging me through all the process of this project, and for helping me in all the bad moments and the struggles that happened during it. For paying me attention, through all these months, even when I was abroad, and answering and revising things on weekends.

Thanks to Professor Stefania Serafin and the people from Medialogy at Aalborg University, for helping me define the project and giving me advice and ideas for starting it.

I would also like to thank my family, for giving me the strength to keep on, and having a lot of patience through my struggle and up and downs through all these years.

Thanks to the professors at Universitat Pompeu Fabra who were interested in this project and allowed me to discussed some of the issues with them. Also thanks to Lluís de las Heras, who advised me at the beginning of this project and gave me numerous suggestions.

Thanks to my friends, specially to Francina Careta, who has supported me, has advised me, and has endured me through all the nerves and stress of these months; and to Ferran Mesas, who helped me in countless times when compiling Gaia, and many other things that I got stuck to.

Thanks to all of them for being here during all these years.
Abstract

Although cinema is an art with a history of just over a 100 years, its evolution has been much faster than any other form of art, both in technology and creativity. However, the essence of cinema has been preserved over the years, and its most essential elements has remained over the huge changes.

Regarding soundtracks, these are a key feature to cinema, very defined for the different existing genres. If we focus specially on the most classic genres, we can easily find characteristics that help us link the song to its corresponding soundtrack.

This project was conceived with the idea of study the evolution of soundtracks in order to understand its characteristics, and using “Music Information Retrieval” methods, being able to gather the necessary information from songs of different soundtracks and compare them, obtaining relevant common characteristics of a genre.

This process will allow to determine the characteristics of a genre, and with the use of machine learning algorithms, be able to classify in real time new songs, comparing the obtained data with the saved one.

Finally, the results of this project aim to contribute to the future development of the analysis of movie soundtracks, allowing not only the automatic classification of genres, but also understanding its characteristics.

Keywords: Music Information Retrieval, machine-learning, film, soundtrack, genre, automatic classification, descriptors, Weka, Essentia, real time classification.
Resum

Tot i que el cinema és un art amb una història de poc més de 100 anys, la seva evolució ha estat molt més ràpida que la de qualsevol altre art, tant pel que fa a la seva tecnologia com a la vessant creativa. Tot i així, la essència del cinema s’ha conservat al llarg dels anys i els seus elements més essencials s’han mantingut per sobre els grans canvis.

Pel que fa a les bandes sonores, aquests són uns elements clau en el cinema i estan molt definides i en els diferents gèneres existents. Si ens fixem sobretot en les bandes sonores dels gèneres més clàssics, podem trobar fàcilment característiques que fan que identifiquem ràpidament la cançó amb el gènere corresponent.

Aquest projecte neix precisament amb la idea estudiar l’evolució de les bandes sonores per tal d’entendre’n les seves característiques, i mitjançant mètodes d’extracció de dades, i de “Music Information Retrieval” poder obtenir la informació necessària de les cançons de diferents gèneres per tal de poder-les comparar entre elles i obtenir així dades rellevants sobre les característiques comunes en un mateix gènere.

Aquest procés ens permetrà determinar les característiques dels gèneres, i mitjançant algorismes de machine learning, classificar en temps real noves cançons, comparant les dades obtingudes amb les dades guardades.

Finalment, els resultats d’aquest projecte tenen com a intenció contribuir en el futur desenvolupament de l’anàlisi de bandes sonores, permentent no només classificar gèneres, sinó entendre’n les característiques.

Keywords: Music Information Retrieval, machine-learning, pel·lícula, banda sonora, gènere, classificació automàtica, descriptors, weka, Essentia, classificació en temps real.
INDEX

Acknowledgements ........................................................................................................................................ iii
Abstract ..................................................................................................................................................... iii
Resum ......................................................................................................................................................... v
INDEX....................................................................................................................................................... 7
1. INTRODUCTION ........................................................................................................................................ 11
  1.1. Motivations ........................................................................................................................................... 11
  1.2. Goals of this project ............................................................................................................................. 12
2. BACKGROUND ........................................................................................................................................... 13
  2.1. The Silent Era (1900s) ....................................................................................................................... 13
  2.2. The Sound Era ..................................................................................................................................... 14
  2.3. The Modern Era (1950s) .................................................................................................................... 15
  2.4. Contemporary Era ............................................................................................................................... 16
      Romance .................................................................................................................................................. 17
      Animation .............................................................................................................................................. 17
      Action .................................................................................................................................................... 17
      Adventure ............................................................................................................................................ 17
3. MUSIC INFORMATION RETRIEVAL ....................................................................................................... 19
  3.1. Dataset .............................................................................................................................................. 21
  3.2. Musical Descriptors ............................................................................................................................. 21
      3.2.1. Low-level Descriptors .................................................................................................................. 21
      3.2.2. Mid and High-Level Descriptors ................................................................................................. 22
  3.3. Classifiers .......................................................................................................................................... 23
4. METHODOLOGY ...................................................................................................................................... 25
  4.1. Dataset Building ................................................................................................................................ 25
      4.2.1. Preparing the Data ......................................................................................................................... 26
  4.3. Descriptors .......................................................................................................................................... 27
      4.3.1. Essentia ....................................................................................................................................... 27
      4.3.2. Descriptors .................................................................................................................................. 28
      4.3.3. File conversion ............................................................................................................................. 30
4.4. Weka.................................................................................................................................34
  4.4.1. Classifiers ......................................................................................................................35
5. CLASSIFICATION RESULTS.................................................................................................36
  5.1. Accuracy.............................................................................................................................36
    ZeroR .......................................................................................................................................37
    Decision Tree (J48) ..................................................................................................................38
    Support Vector Machines (LibSVM) .......................................................................................39
    Support Vector Machines (SMO) ...........................................................................................39
    PART .......................................................................................................................................40
    K-Nearest Neighbors ...........................................................................................................41
5.2. Error Analysis .....................................................................................................................43
5.3. Improving the results..........................................................................................................43
  5.3.1. Gaia ...............................................................................................................................44
  5.3.2. Results with Gaia ..........................................................................................................45
  5.3.2. Binary Datasets ..............................................................................................................47
5.4. Results Analysis ..................................................................................................................50
6. REAL TIME CLASSIFICATION..............................................................................................52
  6.1. Data Preparation .................................................................................................................52
  6.2. Process Design ..................................................................................................................52
    6.2.1. Data Gathering .............................................................................................................53
    6.2.2. Data Processing ..........................................................................................................55
    6.2.3. Weka Core ...................................................................................................................55
    6.2.4. Outputs ........................................................................................................................56
6.3. Prototype ............................................................................................................................57
7. EVALUATION ........................................................................................................................59
  7.1. Results and Limitations ......................................................................................................59
  7.2. Applications .......................................................................................................................60
8. CONCLUSIONS .....................................................................................................................61
  8.1. Future Work Development ...............................................................................................62
9. REFERENCES ........................................................................................................................64
ANNEX ........................................................................................................................................66
Annex I. Glossary ................................................................. 66
Annex II. Movie List ................................................................. 68
Annex III. Descriptors ................................................................. 70
Annex IV. Code ................................................................. 71
1. INTRODUCTION

1.1. Motivations

Nowadays there is a huge expansion of movies, not only in the field of the industry itself, but also in the field of the creativity. More and more movies try to accomplish impressive achievements in the field of Technology, visual effects, and even audio effects, with amazing results. Even when it comes to the creativity and narratives, the stories are more elaborated and experimental.

But there’s one field that is more static: the soundtracks. The field of the music scores, in a composition point of view, hasn’t changed much. The Technology might have changed, we have improved a lot in the way we shoot films, but in general terms, most of the music composed and arranged for a movie follows a very defined pattern.

This means that, despite having innovation and explore new ways of moviemaking, we can find evident similitudes in the field of the soundtracks from movies of the same genre.

If we start thinking about it, when we listen to some score from a movie, we can determine quite fast in which genre this movie belongs to. The same happens with photography. If we watch a movie, or a scene, we can see, only by the treatment of color and light, which is the genre of the movie. If we start to focus on different genres, we can find that most of the music of the movies from similar genres has some “things” that we find characteristic, although we don’t quite know why this is like this.

This arises some questions: Which are the elements that allow us to determine the genre of a film by its music? And by its images? Could we use that to automatize the genre classification of movies?

Trying to answer all these questions is the principal motivation of this project. With this project, we would like to understand the elements and features that we can extract from a sound or an image, understand how do they relate between each other, and how we could use them to determine the genre of a movie.

Once we know the genre of the movie, we can determine which are the elements that characterize a genre, and extract all this data in order to create an automated system which given the movie, it compares it with the genre database, and can be able to determine which is the correct genre for the movie.

The motivation for this project is to precisely understand which are the elements that characterize different genres in a numerical way, allowing us to create an automatic genre classifier for movies in real time.
1.2. Goals of this project

As we mentioned before, the motivation of this work is to understand the importance of a film soundtrack in order to extract their key features.

In this project we will try to accomplish three main goals: The first one will be to understand why film soundtracks are so important in film, and which are the main characteristics of a soundtrack that determine the genre of a movie. The second one will be to analyze a dataset of soundtracks, obtaining their key features and try to understand why these features are present in all the movies of the same genre, understanding which is the relation between the genre and the soundtracks. Our last goal will be to get all these extracted features and create an automated system which will try to determine the genre of a movie given a random soundtrack.

This last goal will help us to create a system that, given the right feature analysis, could allow us to determine the genre of a movie with the use of only its soundtrack, obtaining results with the minimum number of errors as possible.
2. BACKGROUND

First, and before entering in the details of the technical characteristics of musical feature extraction, we must understand the relationship between film and music, and their historical background in order to comprehend why is the music so important in a film, and which are the main characteristics that we can find in different genres through film history.

We are going to analyze the use of music in film, and the evolution of both towards what we know today. If we look closely, there are three main stages that are relevant to music film industry: The Silent Era, The Sound Era, and the Modern Era. Of course there are more stages in film history, but as this is just a brief introduction and background, we will just cover the basics.

2.1. The Silent Era (1900s)

Since the origins of cinema, in the late 1800s and the beginning of 1900s, cinema didn’t have any sound, which is known as the silent era. But the first projections had a major problem: the projector machines that were used were very noisy, which made the projections very annoying. That’s why live music was added to the projection, in order to try to hide the sound of the projector.

But soon, it was realized that the use of music could add much more to the film experience than only covering the sound of the machines used in the projections. The music that was being played could add a new level of immersion to the viewer. Following the steps of Opera or Concerts, they realized that music could also tell a story in cinema, or add subtle things that could not be added at the moment by the actors themselves.

This phenomenon led to the apparition of full original scores written for specific films. One of the first examples that we can find is ‘La mort du Duc de Guise’, directed by Charles Le Bargy in 1908 (Figure 1). Camille Saint-Saëns composed the score especially for the movie, one of the most prestigious French composers of that era, and with a huge experience in music in theatre.

Such was the success of the first score that in 1909, Max Winkler published ‘Suggestions for Music’, which included film music cue sheets for silent movies, institutionalizing the use of music in films, and accepting its huge importance.

Since then, music evolved, from small pieces to full-orchestra scores, played during the projections of the film. We can see that directors started to see music not only as a thing that accompanied the
film, but it was something that had life by its own. Music wasn’t just something to help the audiences immerse in the movie, but it had been gaining importance.

In this era though, the predominant music is pianos and small chamber groups for the small production movies, and also big orchestras that played during the projections at the major pictures. Although the difference between the musical resources between big and small productions, the content of the music was strongly tied to the action that was happening in the image.

2.2. The Sound Era

Then, cinema was not silent anymore, and actors could finally talk. This was a huge change in the film industry, and a revolution. Most of the rules were changed, and the film industry changed again.

In the beginning, there was quite a receding to the beginnings of cinema. Shots were very static, acting was again very static, and music was left in a second position. This was because directors, actors and technicians were not used to shooting with sound. Actors didn’t know how to act and speak, and technicians didn’t have the right knowledge about most of the new material, leading to poor and sound recordings.

Again, music was left behind, as directors didn’t know how to combine music and speech inside the film. This lead to the thought that music in film was finished, as the main purpose until that date was to complement what the actors couldn’t say in words, but now that actors could speak, music wasn’t required in the film anymore.

This thought, though, was quickly proven to be wrong, as films without music became static and boring, and cinema makers soon realized that music was essential to films. Now that the orchestra didn’t have to be playing live at the cinema, there was room for further editing, and experimentation with music.

This led to the use of musical genres that weren’t previously used in cinema, and the experimentation of music in movie soundtracks, specially in the decade of the fifties. Pioneer examples of the new use of movies can be found in Cobweb (1955), by Leonard Rosenman, which was the first movie to experiment with atonal music; Forbidden Planet (1956), by Louis and Bebe Barron, which was the first movie to use a complete electronic score.

There were also a few attempts at the usage of stereophonic sound in movies, but it was played in a separate track than the film. For example, in Walt Disney’s Fantasia (1940), a three-
channel stereophonic is used by reproducing three different tracks simultaneously with the movie projection, focusing specially in the quality of the sound, as it could be seen in the promotional posters of that time (Figure 2).

These innovations started to shape what we know nowadays as Original Soundtracks, and set the bases for the actual soundtracks in genres.

2.3. The Modern Era (1950s)

During the fifties and sixties, Hollywood had one of the worst decades of the industry, as it registered significant losses in their revenues. The responsible of this loss was the apparition of a new competitor that could make the cinema disappear: the apparition of the television in homes.

Television was bringing cinema home, meaning that people didn’t have to go to the cinemas in order to watch their favorite movies. For years, movie studios tried to fight television in courts, but seeing that their fight was a lost one, they realized that they had to cope with television and coexist.

In order to coexist, cinema tried to innovate, and give the people something that television couldn’t give them. Cinema had two important things that played on its favor: the size of the screen and the technical advancements. While television was still in its beginning, in black and white and with a bad quality of image, cinema had decades of color, lens, recording and sound techniques that television still didn’t have in its technology.

This meant that, while television was still in black-and-white, with a poor quality of sound, etc.; cinema had full color big screens, with proper sound, and impressive visual effects. During this time, directors found out that they could tell different stories using all these technologies, hence the apparition of new genres. It was time for innovation, not only in the field of music, but also in the field of the art itself.

One of the innovations, and maybe the most relevant and lasting, was the Cinemascope, which meant the introduction of wide screen recording and later projection.

The creators of cinemascope thought that if they offered a stunning picture, they should also offer an equally impressive audio for their movies. This meant that cinemascope included true stereophonic

![Figure 3 - Disposition of the speakers and the screens in Cinemascope System. Source: widescreenmuseum.com](image-url)
sound in the same film through 3 channels, giving a three-dimensional effect, as we can appreciate in Figure 3. This was quite an accomplishment, as, as we mentioned earlier, playing the sound in a different track so far used stereophonics.

Another accomplishment in the decade of the fifties was the popularity of songs from movies. The first success case was the song *Do not Forsake me Oh My Darlin’*, by Dimitri Tiomkin. This song was composed for Fred Zinnermann’s *High Noon* (1952). The song became so popular that producers started to realize that they could also sell the soundtracks of the movies independently. During these years, Dimitri Tiomkin became one of the referents when recording soundtracks. In Figure 4, we can see the expansion of the soundtrack, in the recording of *55 Days at Peking* (1963), 

The sixties continued to innovate with the use of different scores in films, like including existing songs in movies, such as rock, jazz, etc. Soundtracks were not tied to an orchestra anymore, and any sound and song could be arranged or composed to fit in the movie.

This is where we can start to find clear differentiations in the soundtracks of different genres, such as action movies, drama movies, horror, comedy, etc. Music was used to enhance the genre itself, and it had a huge characterization part. Music had to characterize the genre.

In the seventies, there is a golden age for music soundtracks, going back to the orchestration in movies, with the ‘inflection point’ of *Star Wars: A New Hope* (1977), by George Lucas. The score and orchestration of the movie set the bases for the contemporary cinema score, and bringing them to the golden age of music composition and orchestration. From then, we can find both orchestration music and also existing songs from different genres such as rock, pop, jazz, etc.

**2.4. Contemporary Era**

Nowadays, we can find both orchestrated scores and pieces taken from existing music. We can find music from almost any genre into any film. However, most of the genres still follow a ‘classic’ approach to its soundtrack.

As our goal is to focus our work in defined soundtracks, we will study four well-defined genres, which follow a more classical approach to the genre, and are the ones that have maintained its
structure through the years, the most classical ones. We will have a short introduction on them, based on the content of the movie, and its musicalization.

**ROMANCE**
Romance movies are based on romantic love stories that focus on passion, emotion, and affectionate romantic involvement of the main characters. Romance films make the romantic love story or the search for strong and pure love and romance the main plot focus.

Its music is usually very orchestrated, with a slow tempo and melodic themes. Usually, we often find string instruments (such as violins).

A ‘classic’ romance movie in the terms of its soundtrack can be Cold Mountain (2003).

**ANIMATION**
Animation movies are the recording of a sequence of drawn images, creating an illusion of movement. Animation in film is usually focused on child audiences, meaning that most of the pictures will have childish motives.

Some of the characteristics that we can find in its soundtracks are melodic childish tunes, usually using other instruments than conventional ones, such as bells and rattle sounds. Their songs will also feature fast paced tempos, with high pitches and very orchestrated timbres.

Animation is a genre that have changed quite a lot through the years, as it has passed from drawn pictures to computer animated movies. Its soundtracks have also changed to the new technologies, but the main essence of them still remains.

One example of a ‘classic’ drawn movie could be ‘The Lion King’ (1994), while a modern animation movie could be ‘Brave’ (2012).

**ACTION**
Action is a film genre in which one or more heroes are thrust into a series of challenges that typically include physical feats, extended fight scenes, violence, and frantic chases. Action films tend to feature a resourceful character struggling against incredible odds, which include life-threatening situations, a villain, or a pursuit that generally concludes in victory for the hero.

Its key features are a fast tempo, usually with high-energy peaks, with dissonant tunes and not very melodic. Its music is usually orchestrated, but with a high use of rock music and electronic themes.

One example of a ‘classic’ Action movie is Die Hard (1988).

**ADVENTURE**
Adventure films are a genre of film which include quests for lost continents, a jungle and/or desert settings, characters going on a treasure hunts and heroic journeys for the unknown. Unlike action films, they often use their action scenes preferably to display and explore exotic locations in an energetic way.
Its main characteristics are a fast tempo, with very tonal music. It is very orchestrated, and most of its tracks are variations of a ‘main’ track.

Some classical examples from a soundtracks point of view that fit this description could be the Lord Of The Rings (2001)
3. MUSIC INFORMATION RETRIEVAL

As J. Stephen Downie stated in *Music Information Retrieval*\(^2\), we can define the following problem:

“Imagine a world where you could walk up to a computer and sing a song fragment that has been plaguing you since breakfast. […] The computer suggests to you that “Camptown Races” is the cause of your irritation, and you confirm that by listening to one of the many MP3 files that the computer has found.”

This problem finds its solution through the use of Music Information Retrieval (MIR). MIR is an interdisciplinary science that tries to retrieve information from music. It involves background combinations from musicology, psychology, signal processing, and machine learning among others.

Its main use is by businesses and academics to classify, categorize and manipulate music, having a large variety of existing systems on the market.

Nowadays, state of the art technologies allow us to automatically compute descriptors from music signals in terms of timbre and instrumentation, rhythm (tempo, beats and rhythmic pattern) and tonality (key, chord progression). State of the art algorithms do not always work perfectly, but have been used to compute similarity between songs and to classify songs according to mood, genre or culture.

There are many existing examples of MIR applications, such as recommender systems, instrument recognition, automatic music transcription, genre classification, mood classification, similarity, and even music generation.

Some recommender systems, such as *Pandora Radio*\(^3\), haven’t been using MIR techniques, and have mainly based their recommendations on experts to tag music, and to compare the listening history between other users with similar histories. MIR methods are starting to be used and incorporated to such existing systems, e.g. through technologies provided by companies such as *Gracenote*\(^4\) or *BMAT*\(^5\).

On the other hand, automatic categorization, music transcription, mood classifier, etc. relies completely on Music Information Retrieval methods, such as statistics and machine learning. These systems usually take a given track (in WAV or MP3) and then extract features from it. These features contain information from the signal, such as beat, loudness, amplitude, frequency, etc. This information will later be compared to a large dataset, with thousands of other tracks that have already had their descriptors extracted. These descriptors will be compared to the other descriptors and will obtain results depending on the similarity between these descriptors.

This procedure is known as Supervised Learning, and consists of inferring a function from labeled training data. The training data consist of a set of training examples, which in our case will be the descriptors of different songs from different genres. These trained examples are analyzed and then produce an inferred function, which can be used in order to map new examples (or predicted sets), as we can see in the following figure:
In this figure we can see the explanation of a Supervised Learning system. In this case, we can see that we have two different steps: Train and Predict.

During Train, we extract the features from different songs, and then we label them manually. Once we have done this, we use machine-learning algorithms, such as Support Vector Machines, in order to obtain a trained model, which we are going to store.

During Predict, we repeat pretty much the same steps. We extract the features from the track we want to test. Then, we send it to the classifier model, in which we compare the features from the file we want to predict and the features we trained before. Then, we will obtain the Label from the file we wanted to predict.

As our main goal in this project is to analyze the tracks, and obtain information that will help us see which are the characteristics of different genres, we will use some of the methods explained before in order to create a train dataset, which will be functional when analyzing new test data.

Once we’ve seen the different characteristics of the genres that we will work with, we will see which are the best techniques that will be used in order to obtain all the necessary information that will be used in order to determine the genre of the movie.
In order to determine the elements that are present in the songs, and that will be compared in the scores of the same genre, we will use different descriptors, based on low, mid, and high level features, as we will explain in further detail later. In order to obtain these descriptors, we will have to analyze the songs and focus on different aspects of the tracks.

### 3.1. Dataset

In order to guess the genre of a movie by its soundtrack, we compare the song with a library of existing songs. In order to obtain results, we extract relevant information from the existing songs, and save them in a dataset.

Once we've built this dataset, we are able to process it, and find similarities between the different songs. Once we've found similarities, we are able to have some conclusions, and group songs depending to the genres. For example, it is expected that songs from the same movie genre have similar descriptors values. These values are saved, and grouped into the desired genres.

Then, when one song is introduced, its descriptors are compared to the ones that are already in the dataset, and it tries to see which is the genre that is most similar to the song. We obtain a percentage of each genre, meaning that the song is closer to a genre than another.

The goal of the system is that it will be able to learn. The more is used, the more will learn, and more precise it becomes.

### 3.2. Musical Descriptors

In order to obtain the necessary information to achieve our goals, we will have to work with different musical descriptors. Musical descriptors will have to be extracted from every track of our dataset, and they will contain all the necessary and relevant information from these tracks in order to determine patterns from them. In our case, we will want to extract descriptors that can be useful to the classification of genres.

We will work with low-level, mid-level and high-level descriptors:

#### 3.2.1. Low-level Descriptors

Low-level descriptors are the descriptors that are closely related to the signal of the score itself[^6]. These descriptors depend on the duration of the analysis. We can have instantaneous descriptors, segment descriptors, and global descriptors, depending on the duration of the acquisition of the information.
INSTANTANEOUS DESCRIPTORS:
These descriptors extract features at any point of the signal, usually take about 50ms of it, and are very useful to obtain information about the energy or the amplitude of the signal.

SEGMENT DESCRIPTORS:
These descriptors work on small regions of time -usually from 1s to 10s-, such as spectrum related features. These frames are useful in order to segment the track into regions that are easily classifiable in terms of their content. This is useful in order to identify and extract region attributes that will give a higher level control over the sound.

These descriptors allow us to compute attack, steady and release regions, as these regions are identified by the way the instantaneous attributes change in time.

TEMPORAL DESCRIPTORS:
These descriptors are the ones that take features along the whole duration of the signal.

3.2.2. MID AND HIGH-LEVEL DESCRIPTORS
Unlike the low levels descriptors, that only extract acoustic information (information from the signal), high-levels descriptors can carry semantic or syntactic meaning.

Syntactic descriptors refer to features that can be understood by an end-user without previous signal processing knowledge but do not carry semantic meaning. Syntactic high-level descriptors can be sometimes computed as a combination of low-level descriptors. They can be seen as attributes of our sound classes but, by themselves, cannot be used to identify objects and classify them. For that reason, the computation of syntactic descriptors (either low or high-leveled) is not dependent on any kind of musical knowledge, symbolic or real-world knowledge

3.2.2.1. SEMANTIC DESCRIPTORS
These descriptors refer to meaningful features of the sound and are understandable for the end-user [6]. We therefore need to apply more “real world” knowledge. The degree of abstraction of a semantic descriptor though has a wide range, labels such as ```scary``` or more concrete such as ```violin sound``` can be considered semantic descriptors.

Some of the high level features that we can find are descriptors that represent timbre, tempo and Tonality.

TIMBRE: Timbral features describe the quality of the sound that allows us to differentiate particular musical sounds. For example, timbre will allow us to differentiate between a trumpet and a piano, when both are playing at the same tone and loudness.
This feature is useful in our scenario, as it will help to determine which are the predominant instruments that are present in the tracks corresponding to the same genre.

**TEMPO**: This feature is described as the repetition of ‘beat’ of the music. This means that we can determine the pace of the tracks by calculating its “peaks” by finding the onsets of the track. These onsets determine the beginning of a new note, detecting the its attack.

**CHROMA**: This feature can be described as the pitch content of the analyzed piece. When we talk about pitch, we are talking about the “note” of the music, or the fundamental frequency. This feature can be useful in order to determine which are the predominant notes that are present in the tracks of the same genre, allowing to determine which are the predominant notes.

This feature will be useful in order to determine which is the average pace that the tracks of the same genre have in common.

### 3.3. Classifiers

In order to work with the dataset and train it, we will have to analyze it with the use of different classifiers. These classifiers will allow us to which of a set of categories a new observation belongs, mapping this new instance to a category.

There are many classifiers; the most used being neural (perceptron), support vector machines, k-nearest neighbors, etc. From these classifiers, we will study the following ones.

**ZEROR** is the simplest form of classification, as it simply predicts the class with more number of instances [7]. It is very useful in order to predict a baseline for determining a basic prediction, and a benchmark to initialize more advanced predictors.

**DECISION TREE** builds classification models in the form of a tree structure. It breaks down the dataset into smaller subsets, while the tree is developed. The result will be a tree with decision nodes and leaf nodes. Decision nodes will have two or more branches, while leaf nodes will represent a classification or decision.

**SUPPORT VECTOR MACHINES** is a supervised learning which tries to classify data given a trained data source. In order to determine into which group belongs the sample, it try to find a hyperplane that maximizes the distance between the classes.

In our case scenario, where we have a large dataset, and all of the instances are labeled, we can use a variation of SVM, called Sequential Minimal Optimization. This algorithm requires less processing power and it is less expensive than SVM, and thus, it is faster.

**K-NEAREST NEIGHBOUR** algorithm consists of the k closest training examples in the feature space. This means that the instances that are closer to themselves will be classified in one group, while the instances that are close to another group, will belong to the other group.
In order to decide which is the best classifier for our case scenario, we will have to choose the one that gives us a better accuracy rate, as we will see in the following steps.
4. METHODOLOGY

4.1. Dataset Building

As stated before, the dataset will be built with the scores from the movies. The purpose is to accomplish a large database in order to have enough data from each genre to obtain results as averaged as possible. When choosing the ‘candidate’ movies, the criterion has been to try to get the most characteristic movies of a genre. For example, in action movies, we will try to get ‘Action classics’: Movies that are purely action. The same criteria have been applied for the rest of the tracks in the other genre.

All the movie information about the movies and the genres has been obtained from the Internet Movie Database. The music has been downloaded using Youtube [8] and ListenToYoutube’s [9] services. All the rights to the music correspond to their respective owners, and the use of these tracks in this project is merely for an educational and research purpose.

In order to create our dataset, we have searched for a large number of representative movies from our dataset are composed by a total number of 769 tracks from 184 different movies. Each movie has more than one track. This is because one only track is not a representative sample of the entire movie. This is because a movie has different parts and an only track is not relevant in the movie. If we want to determine the genre of the movie, we will have to analyze different tracks from the movie, obtaining a much general view of the whole movie, and how its soundtrack determines the genre.

The dataset created is as it follows:

<table>
<thead>
<tr>
<th>Genre</th>
<th>Number of movies</th>
<th>Number of tracks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romance</td>
<td>40</td>
<td>190</td>
</tr>
<tr>
<td>Adventure</td>
<td>52</td>
<td>181</td>
</tr>
<tr>
<td>Action</td>
<td>46</td>
<td>184</td>
</tr>
<tr>
<td>Animation</td>
<td>46</td>
<td>214</td>
</tr>
<tr>
<td>Total</td>
<td>184</td>
<td>769</td>
</tr>
</tbody>
</table>

*Figure 6.- Table with the number of films and tracks used in every genre*
As we can see, the total number of tracks is 769. These tracks are obtained from 184 different movies. As we can see, the number of tracks of animation is quite bigger than the number of tracks in the other genres. The number of tracks in Adventure is slightly smaller than the number of tracks in the other genres. But we can see that the number of movies in Adventure is slightly higher than the number of other movies, meaning that we have a slightly bigger variety of tracks.

In the case of Animation, we have a larger number of tracks, because the animation genre is not a very definite genre. It encompasses a large variety of ‘sub-genres’, and all of them have different types of music. We have tried to focus on the ‘Child Animation’, which are movies that have a more childish plot, and the soundtracks are more similar and have a more “childish” tune.

4.2.1. Preparing the Data

As we mentioned before, we have chosen quite a lot of tracks in order to obtain a model as accurate as possible. But the problem is that usually, songs have different parts within their structure. If we want to get representative information about the songs, we can try to analyze only the part in the middle of the song, which will probably be the part with the most orchestrated part, and the chorus.

**DATA OBTENTION**

![Workflow for obtaining the data](image)

*Figure 7. - Workflow for obtaining the data*

In order to do so, we have decided to analyze only 60 seconds of each track, starting at instant 30s, and finishing at instant 90s. In order to cut the songs, we have used FFMPEG’s libraries.

FFMPEG is a multimedia framework capable of encoding, decoding and operate with multimedia files, accepting pretty much every existent format. We have used FFMPEG because it allows the use of mp3 files, and the operation of cropping a song is already implemented, and seamless easy.

In order to crop the song, we have to call FFMPEG as it follows
Where “$f” is the variable of every file, -ss makes a reference of where to start, and -t indicates how many seconds are we going to take. the parameter -acodec makes reference to take the audio file, unaltered. Finally, the parameter ‘copy’ makes reference to copy the audio to a new output file. So, with this script, we are saying that we want to create a new mp3 file, called “$f” (the name of each file), which will be the original file, taken from second 30 to second 90.

In order to automatize the process, and crop all the songs in one step, we can create a simple script that takes all the mp3 files in our soundtracks folder, and invokes the previous FFMPEG function for each one of them:

```bash
for f in *.mp3; do ./ffmpeg -ss 00:00:30.00 -t 60 -i "$f" -acodec copy "$f" ../**New folder**/$f; done
```

As before, with this script we will have all the files cropped to 90 seconds and saved in a new folder, keeping the original name of the file.

### 4.3. Descriptors

#### 4.3.1. ESSENTIA

In order to extract the necessary descriptors, we will use one of the built-in extractors in Essentia [11].

Essentia is an open source library created by the UPF for audio analysis and music information retrieval (MIR). The goal of using Essentia is to analyze in a serialized way all the tracks, and extract all the desired descriptors. The advantage of using Essentia is that it decreases significantly the amount of time required to extract a pool of tracks.

Once we have installed all the dependencies required, compiled and built essentia, we can call it and use it for all the existing tracks in a folder.

The extractor that we have used is `streaming_extractor_archivemusic`. This extractor can be used as an executable command line instruction, and computes a large set of low, mid and high level descriptors. This set of descriptors are suited for large music computations, which is perfect for our objectives, and given our large dataset.

The process in order to call Essentia’s extractors is to give it two parameters: an input file, and an output file where the results will be written. In our case, we will use the cropped mp3 files as input
files, and we will write the results in JSON files, which is the format in which Essentia returns the results. So, for a single track, we will go to Essentia’s built folder, and we will call the extractor as it follows:

```
/essentia/build/src/examples/streaming_extractor_archivemusic
"filename.mp3" "filename.json"
```

Where the first parameter is the path to the mp3 file that we want to analyze, and the second parameter is the path to the output file where the results are going to be written.

Again, if we want to automatize the process, we can create a simple bash script that will take all the cropped mp3 files, and extract all the features at once, as it follows:

```
for f in *.mp3;
do
    /essentia/build/src/examples/streaming_extractor_archivemusic
    "$f" "$f.json";
done
```

The output will be a JSON file for each mp3 cropped track, which we will have to use in order to analyze all the results.

### 4.3.2. DESCRIPTORS

The used descriptors in our project are the ones that follow:\[12\]:

**LOW-LEVEL DESCRIPTORS**

**BARK-BANDS:** Calculates bark bands energies. Bark bands are calculated by the bark scale, which is a psychoacoustic scale based on subjective measurements of loudness.

**DISSONANCE:** Tries to estimate the level of ‘harmonicity’ or ‘inharmonicity’ of an audio frame of the piece.

**ERBBANDS:** Calculates the energies in bands, calculated on an equivalent rectangular bandwidth scale. Equivalent Rectangular Bandwith is also a measure used in psychoacoustics, which gives an approximation to the bandwidths of the filters in human hearing.

**HFC:** Computes the high frequency content measure. HFC is a measure that can be used to characterize the amount of high frequencies in the signal. It can be useful for onset detection.
**Melbands:** compute the mel bands energies. Mel scale is a perceptual scale of pitches, judged by listeners to be equal in distance from one another.

**MFCC:** computes the mel frequency cepstrum of a frame. The mel frequency cepstrum is the representation of the short-term power spectrum, which describes the variance of the data distribution over the frequencies.

**GFCC:** similar to the MFCC, it calculates the cepstrum coefficients of a frame, but using the gammatone feature.

**High-Level Descriptors**

**Beats-Loudness:** it computes the loudness of the signal based on windows centred on the beats locations.

**BPM:** this calculates the beats per minute of the track, meaning the repetition of the percussion of the sound (or the beat). This will allow us to determine the pace of the tracks, by calculating its peaks found on the onsets of the signal.

**Danceability:** this descriptor returns whether a song is ‘danceable’ or not, based on its tempo.

**Tonal Chords:** returns the sequence of chords found on the score.

**Key Strength:** returns the key and the scale of the song.

**Tuning:** returns the frequency in which the song is tuned and the number of cents to 440Hz

**HPCP:** computes the Harmonic Pitch-Class Profile of a spectrum.

The reason why these features were chosen is because they can allow us to obtain information regarding the main parameters, which are the rhythm, the melody, the pitch and the timbre.

With the use of these features and their descriptors applied to the dataset that we have created, we will try to obtain useful information and see which are the patterns that are common in music from the same genre, and what distinguishes the genres between them.

In order to extract the defined features from the sounds, we will use different descriptors, which will be later stored and processed.
4.3.3. File conversion

Once we have calculated all the descriptors for each track, Essentia provides all the results formatted in a JSON file containing all the descriptors of one song. As we will explain later, we will want to import this data into Weka in order to analyze the results and see which is the classification accuracy. The problem here is that we will have to adapt the output files from Essentia into a format that is accepted by Weka. So, we will have to do two different things: first, we will have to change the format of each output file into an accepted Weka format, and second, we will have to concatenate all the files into one single file, with all the descriptors of each file.

Weka works with its own format, which is ARFF (Attribute Relation File Format) \(^{[13]}\), although it provides converters for other formats to the ARFF format. The ARFF format follows some rules regarding the order of the attributes and the data. We can take examples from the ARFF documentation:

```plaintext
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa

@ATTRIBUTE sepallength NUMERIC
@ATTRIBUTE sepalwidth NUMERIC
@ATTRIBUTE petallength NUMERIC
@ATTRIBUTE petalwidth NUMERIC
@ATTRIBUTE class{Iris-setosa,Iris-versicolor,Iris-virg.}
```

As we can see, all the data is ordered in comma separated values, and the final argument corresponds to the class that it belongs. All this data is attached to a headers section, which is the one that defines the name which defines each of the data.

```
@ATTRIBUTE sepallength NUMERIC
@ATTRIBUTE sepalwidth NUMERIC
@ATTRIBUTE petallength NUMERIC
@ATTRIBUTE petalwidth NUMERIC
@ATTRIBUTE class{Iris-setosa,Iris-versicolor,Iris-virg.}
```

So, we can take this format as values separated by columns, which will result into something like:

```
sepallength   sepalwidth   petallength   petalwidth   class
5.1           3.5          1.4           0.2          Iris-setosa
```

```
Figure 8.- Example of an ARFF file format
Figure 10.- Example of the headers section
Figure 11.- Example of the relation between the data and the attributes, written in a more comprehensive format
```
Luckily for us, ARFF file is very similar to a conventional CSV file, which turns the conversion of formats quite easy. The problem will be to concatenate all our files and save them in the correct format for Weka to accept them.

4.3.3.1. CONVERTING JSON TO CSV

JSON is an open standard format that uses human-readable text to transmit data objects consisting of value pairs.

Here, an extract of a JSON file with the results calculated using Essentia:

```
"lowlevel": {
    "average_loudness": 0.340264558792,
    "barkbands_kurtosis": {
        "dmean": 2.4716629982,
        "dmean2": 4.18855237961,
        "dvar": 59.422564697,
        "dvar2": 178.47853516,
        "max": 167.013320923,
        "mean": 3.00291180611,
        "median": 1.1680688858,
        "min": -1.97235131264,
        "var": 59.8568077087
    }
},
```

*Figure 12.* Example of a JSON file format

Technically, JSON is a subset of the YAML format, so luckily for us, we can use a YAML parser in order to convert the data into a comma separated file, which is a format that can be interpreted by Weka. Knowing that, we can use any YAML converter to pass from a YAML file to a CSV file. In our case, we have used YAML2CSV [14], a parser created by Arnau Sanchez.

```
for files in *.json;
    do yaml2csv "$files" > "$files.csv";
    sed -i '1,16d;316,366d;385,394d;411,425d"' "$files.csv";
done
```

In order to agilize the process and to save time, we have used the `sed` libraries [15] to delete those lines that were unnecessary, like the metadata parts, or other attributes that are not numerical. Again, we have created a simple script that takes all the JSON files in a folder, and invokes the yaml2csv converter.
The results in this case will be the following,

```
lowlevel,average_loudness,0.340264558792
lowlevel/barkbands_kurtosis,dmean,2.47166299982
lowlevel/barkbands_kurtosis,dmean2,4.18855237961
lowlevel/barkbands_kurtosis,dvar,59.4222564697
lowlevel/barkbands_kurtosis,dvar2,178.474853516
lowlevel/barkbands_kurtosis,max,167.013320923
lowlevel/barkbands_kurtosis,mean,3.00291180611
lowlevel/barkbands_kurtosis,median,1.1680688858
lowlevel/barkbands_kurtosis,min,-1.97235131264
lowlevel/barkbands_kurtosis,var,59.8568077087
```

**Figure 13.** Example of the same file, converted to CSV using Yaml2Csv's library

As we can see, we have the two first columns with the names of the headers, and the third column is corresponding to the values of each descriptor.

The correct format in which Weka interprets the information is by having the first row with all the headers, and the second row with all the descriptors corresponding to these descriptors. Finally, the last column of the file must have the class in which the track belongs. In our case, it will be the genre of the movie: Adventure, Animation, Romance or Action.

In order to adapt all this information and parse it into the correct file, we have created two different Python files. The first, takes the two first columns, corresponding to the attributes, and concatenates them into a single row separated by commas. Then, first value is added at the beginning of the row with the “filename” attribute, and a last value is added at the end of the row with the attribute “Class”, which will later correspond to the class name in which each song belongs. All this is saved into a new CSV file.

The second Python script takes the third column, the one with the data, and concatenates all the values of the column into a single row, separated by commas. Then, at the beginning of the row we have added the filename, and at the end of the row, the class where it belongs. The python scripts can be found at the annex of this project.

The scripts have been invoked following the same structure as before:

```
python csvHeader.py find *.csv | head -1
```

In the case of the header, we will only want to have one header for all the files, so we will only compute it for the first file of the folder. Then, the output file will be stored in csvHeader.csv.
This file will contain all the headers in a single row, separated by commas, as we can see in figure 14.

```
filename,lowlevel/barkbands_kurtosis_dmean,lowlevel/barkbands_kurtosis_dmean2,lowlevel/barkbands_kurtosis_dvar,lowlevel/barkbands_kurtosis_dvar2,lowlevel/barkbands_kurtosis_max,lowlevel/barkbands_kurtosis_mean,lowlevel/barkbands_kurtosis_median,lowlevel/barkbands_kurtosis_min,lowlevel/barkbands_kurtosis_var, [...], class
```

*Figure 14.- Extract of the headers file obtained*

As we can see, the first and second column have been concatenated, and the results have been put in a single line. The first element is the filename, and the last element will be the class.

For concatenating all the data into a column, we will use the Python script csvData.py for gathering the data. As the previous case, this script will take the third column of a csv file and will concatenate all the data into one single row. The first element of the data will be the filename, and the last item of the row will be the class where it belongs.

```
for files in *.csv;
  do python csvData.py "$files" "$files.out" Genre;
  cat *.out > resultsData.csv;
done
```

As we can see, we have called the python script for each mp3 file. In this case, this script takes three parameters as inputs. These are the input CSV file, the output file where we will keep the results, and the genre where the file belongs (Action, Adventure, Animation, Romance). In our case, we are saving the output files in .out files. Once all the .out files have been created, we will use the function cat to concatenate all the files into one single CSV file.

Once we have the header CSV file, and the data CSV file, we just need to merge both files into one single big CSV file, containing all the information from each song in our dataset.

```
cat header.csv data.csv >> dataset.csv
```

In order to merge the files, we will use the cat function again, which will concatenate the header file and all the data files into one single CSV file, which will follow the format allowed by Weka.

The results that we will obtain will look like something like what we can see in Figure 15.
As we can see, all the results are ordered in columns, and each column has its header with the corresponding name of the descriptor.

The full results file can be found in the annex of this project, as the Python code to do so.

4.4. Weka

In order to classify new songs, we will need a trained dataset. Once we have the dataset created and ready following the steps defined before, we can import it into Weka in order to create a train set.

Weka (Waikato Environment for Knowledge Analysis) [16] is a machine-learning suite developed by the University of Waikato. It contains a set of tools and algorithms for data analysis and predictive modeling. This tool is very useful in order to analyze and classify a given dataset.

The aim of this is to analyze all the descriptors extracted from the data, and see what’s the relation between the tracks of the same genre, and the tracks from different genres. Depending on how the tracks’ descriptors group, we can extract patterns. Once we have these patterns, we can take the descriptors from a new track, and see which is the distance between the descriptors and the groups created in the dataset.

Once the dataset has been imported in Weka, we can try different classification algorithms in order to try and obtain the greatest accuracy in our set.
4.4.1. Classifiers

Weka implements various classifiers, depending on the needs of the different datasets. In our case, we have tried different classifiers based on knowledge of the different algorithms, as we defined before: ZeroR, Decision Tree, SVM, SMO, K-Nearest Neighbors.

We have tried different classifiers, and evaluated the results taking its accuracy. The better the accuracy of the model, the better the results, as it means that a new instance could be classified with X accuracy of having it correctly classified.

The following results correspond to the use of different classifiers, and its accuracy levels. In order to determine the accuracy levels, we will take the training dataset and filter it in order to obtain the exact same number of instances for each genre. This way, we will not be ‘favoring’ one genre that might have more instances than another, and vice-versa. Then we use 10-fold cross validation, i.e. we create 10 subsets within the dataset. The process is as it follows: with a dataset with more than 700 labeled data, Weka will create 10 equal sized sets. Each set will be divided in two subgroups, with 90% of the data set as the train set, and the 10% resting as the test set, and produces a classifier. Then it does the exact same thing for the other 9 subsets, and averages the performance of this 10 results. This way, we will have a more precise result, as we will run the datasets several times before obtaining a conclusive accuracy.

The reason why a 10-fold cross validation was chosen is because we want to keep an equilibrium between accuracy and processing time. Because of this, when incrementing the number of folds more than 10, the results were barely improved, considering 10 the optimal number of folds for this dataset.
5. CLASSIFICATION RESULTS

In order to see which is the classifier algorithm that works the best, we will try the ones defined before, and we will see which one is the one that gives us the biggest accuracy.

5.1. Accuracy

In order to see the accuracy of our algorithms, we will look for the percentage of accuracy that the different algorithms give us. The ways of seeing the accuracy will be through 3 different values: The confusion matrix, the accuracy percentage, and the F-Measure.

The confusion matrix will provide information about the most similar classes.

The F-Measure also makes reference to the accuracy of the algorithm. It takes into consideration both the precision and the recall of the test to compute the score.

\[
F_1 = \frac{precision \times recall}{precision + recall}
\]

Here, the precision \( p \) corresponds to the number of correct results divided by the number of all returned results. Recall \( r \) makes reference to the number of correct results divided by the number of expected correct results.

\[
precision = \frac{|[relevant\ features] \cap [retrieved\ features]|}{[retrieved\ features]}
\]

\[
recall = \frac{|[relevant\ features] \cup [retrieved\ features]|}{[relevant\ features]}
\]

We will consider good results those which are close to 1, and bad results those which are close to 0.
The results obtained for different algorithms are the ones that we can see in the following sections.

**ZeroR**

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>170</th>
<th>24.01%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>538</td>
<td>75.98%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.231</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 16.* Results obtained using the ZeroR classifying model.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>&lt;-- classified as</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>54</td>
<td>54</td>
<td>18</td>
<td>a = Action</td>
</tr>
<tr>
<td>54</td>
<td>51</td>
<td>54</td>
<td>18</td>
<td>b = Adventure</td>
</tr>
<tr>
<td>54</td>
<td>54</td>
<td>51</td>
<td>18</td>
<td>c = Animation</td>
</tr>
<tr>
<td>52</td>
<td>54</td>
<td>54</td>
<td>17</td>
<td>d = Romance</td>
</tr>
</tbody>
</table>

*Figure 17.* Confusion matrix obtained using ZeroR

ZeroR is the simplest form of classification, and its results are the ‘base’ in which we will try to improve our classification. In this case we can see that we are obtaining an accuracy of a 26% of correctly classified instances. If we look at the confusion matrix, we can see that all of the instances are being classified as ‘Romance’, thus, the ~25% of accuracy.

If we look at the detailed accuracy by class, we can see that the F-Measure for each class is zero except for the Romance class, in which we have an F-Measure of ~0.4.
**DECISION TREE (J48)**

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>353</th>
<th>49.86%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>355</td>
<td>50.14%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.499</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 18.* Results obtained using the J48 classifying model.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>--- classified as</th>
</tr>
</thead>
<tbody>
<tr>
<td>76</td>
<td>40</td>
<td>34</td>
<td>27</td>
<td>a = Action</td>
</tr>
<tr>
<td>40</td>
<td>92</td>
<td>27</td>
<td>18</td>
<td>b = Adventure</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>93</td>
<td>29</td>
<td>c = Animation</td>
</tr>
<tr>
<td>34</td>
<td>18</td>
<td>33</td>
<td>92</td>
<td>d = Romance</td>
</tr>
</tbody>
</table>

*Figure 19.* Confusion Matrix obtained using J48

Using decision trees we can see that the accuracy for the correctly classified instances is around 45%.

If we have a look at the confusion matrix of this classifier, we can see that from the songs that belong to ‘Action’ (181 instances), 74 of them are correctly classified, and 107 are misclassified. In adventure (182), 92 are correctly classified and 90 are not. In animation, 79 are correctly classified and 101 are not. Finally, in romance, 92 are correctly classified and 98 are not.

In this case, we can see that the genres that are better classified are Adventure and Romance, being Action and Animation worse classified. This can mean that both Adventure and Romance have some features that distinguish themselves from the other genres, or that both Action and Animation have some features that make them more confusing than the others.

Finally, we can see that the F-Measure is similar in all the classes, giving an average F-Measure of ~0.45.

This is clearly an improvement of the previous classifier, but it is not enough, as we cannot have an accuracy of 50%, which would mean that for every new song, we would flip a coin in order to see which genre it is.
### Support Vector Machines (LibSVM)

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>170</th>
<th>24.01 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>538</td>
<td>75.99 %</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.231</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 20.* - Results obtained using LibSVM classifying model

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>Classified as</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>54</td>
<td>54</td>
<td>52</td>
<td>a = Action</td>
</tr>
<tr>
<td>18</td>
<td>51</td>
<td>54</td>
<td>54</td>
<td>b = Adventure</td>
</tr>
<tr>
<td>18</td>
<td>54</td>
<td>51</td>
<td>54</td>
<td>c = Animation</td>
</tr>
<tr>
<td>18</td>
<td>54</td>
<td>54</td>
<td>51</td>
<td>d = Romance</td>
</tr>
</tbody>
</table>
```

*Figure 21.* - Confusion Matrix obtained from LibSVM

### Support Vector Machines (SMO)

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>477</th>
<th>67.37%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>231</td>
<td>32.63%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.674</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 22.* - Results obtained using the SMO classifying model
In this algorithm, we can see that the accuracy obtained is around the 70%. In this case, the correctly classified instances for Action are 108, versus 69 which have been incorrectly classified. In the case of Adventure, we can see that 127 instances have been correctly classified, against 54 instances that have been incorrectly classified. In the case of animation, we can see that 119 instances have been correctly classified, versus 61 instances incorrectly classified. Finally, in romance, 144 instances have been correctly classified, and 46 have been incorrectly classified.

With this confusion matrix we can see most of the instances have been correctly classified, being the genre with the best classification Romance, meaning that some of its features are more distinguishable than the ones found on the other genres. This can be seen in the results of the F-Measure, which have a higher value in the Romance genre, and with an average of ~0.7.

The results obtained with this algorithm are much better than the previous one. A result of a 70% is an acceptable result, though it could be improved.

**PART**

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>361</th>
<th>50.99%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>347</td>
<td>49.01%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.508</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 24.* - Results obtained using PART classifying model
With this algorithm, we can see that we have an accuracy of approximately 50%. If we look at the confusion matrix, we can see that for the Action class, we have a total of 74 instances correctly classified, and 103 that have been incorrectly classified. In the case of the Adventure class, 90 instances have been correctly classified, and 91 have been incorrectly classified. In the case of Animation, 88 instances have been correctly classified and 77 have been incorrectly classified. Finally, the Romance class have 108 instances correctly classified and 82 are incorrect. Again, we can see that Romance instances are the ones that have a better accuracy, being Action the class that have the worst accuracy.

If we look at the F-Number, we can see that all the classes have a similar number, except Romance, which have a slightly higher number. In average, the F-Number of the algorithm is around 0.5.

Again, this is not an acceptable result, as it would mean to flip a coin every time that we obtained a result from testing a new song.

**K-Nearest Neighbors**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctly Classified Instances</td>
<td>365</td>
<td>51.55%</td>
</tr>
<tr>
<td>Incorrectly Classified Instances</td>
<td>343</td>
<td>48.44%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.516</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 26.* Results obtained using K-Nearest Neighbors classifying model
In this case, we are obtaining a 55% of accuracy. If we look at the Confusion matrix, we can see that for Action only 82 instances are correctly classified. In Adventure only 108 instances are correctly classified. In Animation only 93 are classified correctly. From these results, we can see that the best classified genre is Romance again, followed very close by Adventure.

Again, looking at the F-Measure of the results, we can see that the best measure is the one obtained by Romance. The rest of them are slightly lower. The average F-measure of the whole dataset is around 0.55.

In conclusion, we can see that the results, summarized, are as it follows:

<table>
<thead>
<tr>
<th></th>
<th>ZEROR</th>
<th>J48</th>
<th>LibSVM</th>
<th>SMO</th>
<th>PART</th>
<th>K-NEAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>51</td>
<td>76</td>
<td>17</td>
<td>109</td>
<td>70</td>
<td>76</td>
</tr>
<tr>
<td>Adventure</td>
<td>51</td>
<td>92</td>
<td>51</td>
<td>123</td>
<td>101</td>
<td>100</td>
</tr>
<tr>
<td>Animation</td>
<td>51</td>
<td>93</td>
<td>51</td>
<td>114</td>
<td>89</td>
<td>94</td>
</tr>
<tr>
<td>Romance</td>
<td>17</td>
<td>92</td>
<td>51</td>
<td>131</td>
<td>101</td>
<td>95</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.231</td>
<td>0.499</td>
<td>0.231</td>
<td>0.674</td>
<td>0.508</td>
<td>0.516</td>
</tr>
<tr>
<td>Correct (%)</td>
<td>24.01</td>
<td>49.86</td>
<td>24.01</td>
<td>67.37</td>
<td>50.98</td>
<td>51.53</td>
</tr>
</tbody>
</table>

*Figure 27.* Confusion Matrix obtained from K-Nearest Neighbors

*Figure 28.* Summarized table with results
5.2. Error Analysis

As we have seen in the previous step, we have analyzed different algorithms and calculated different results. According to them, we can see that our best option here is to use the SMO algorithms, as the global results that we obtain are the highest.

As we have seen before, there are still some genres that its descriptors are easily confused. This may be because the descriptors that we are using are not the best suited for our case scenario. Another option can be that the data contained in the descriptors are not a key feature in the songs from the genre.

When starting this project, there was the assumption that some genres will be more harmonic, others will have a higher beat rate, etc. But when analyzing the results, we can see that there are no ‘special’ features that are present in the songs from one specific genre that define the songs in that genre.

As an example, we can see some of the features in songs from each genre.

![Figure 29](image)

*Figure 29.* Different descriptors arranged by their genres.

This example shows us the distribution of the rate of tonal chords changes. In this case, we can see that all the genres have data along all the line, making it quite impossible to extract information from this descriptor, as the classes don’t show any recognizable pattern. The same happens with other descriptors. Although this, we have been able to classify correctly with an accuracy of a 70%.

5.3. Improving the results

There are several ways to try to improve the accuracy of the results. One of them is to increase the number of descriptors of the dataset, trying to use descriptors that are more meaningful to the classes of our dataset.
Another way to improve our results would be to study new algorithms and see if there is another that can work better on our scenario. We could also try to use the same algorithms as before, but with different parameters, choosing ones that could work better on our dataset too.

In our case, we will try to obtain more descriptors, and see if these descriptors work better on our dataset.

In order to increase the number of descriptors, we will try to add more of them into the existing dataset. As we have seen before, most of the descriptors analyze mainly low-level features of the tracks, meaning that we have a lot of information about the signal itself, and not as much information about the semantic meaning of the tracks. In order to add these high level descriptors, we will use MTG's Gaia linked with Essentia.

5.3.1. Gaia
Gaia is a C++ library that applies similarity measures and classifications on the results. As we said before, the usage of Gaia with Essentia can add a collection of high level descriptors to the existing dataset created by Essentia.

The high-level descriptors added by Gaia are the following:

**GENRE** determines which is the genre of the song, determining which is the closest or most similar genre, such as Pop, Rock, Classical, etc.

**BALLROOM MUSIC CLASSIFICATION**. This descriptor tries to determine to which of the “dancing” rhythms our song belongs to, such as ChaChaCha, Rumba, Salsa, Waltz, etc.

**MOOD**. This descriptor determines whether the song analyzed can be classified as 7 different moods: happy, sad, aggressive, relaxed, acoustic, electronic and party.

**WESTERN / NON-WESTERN** determines if the song has a rhythm and sonority more typical from the non-western culture (such as Indian, oriental, etc.)

**TONAL / ATONAL**. This descriptor determines the song tonality, whether it is tonal or atonal.

**VOICE / INSTRUMENTAL**. This descriptor determines if the song has predominant voice parts (meaning that there are vocals in the song), or has more instrumental parts.

**GENDER**. This descriptor determines whether the vocals of the song correspond to a male or a female voice.
4.3.1.1. Installing Gaia
In order to install Gaia, we have downloaded its source code from its Github[ ], and configured and compiled as the documentation states. Following the steps in the documentation results into many compilation errors, and in order to execute it correctly, many things have had to be tested and in some cases, rewritten. Anyway, there have been many problems during the compilation and execution of Gaia.

After failing to build Gaia in a Mac computer, we have attempted to build it in a Linux machine.

Gaia has been configured following the specifications on the Github documentation, as it follows:

```
$ ./waf configure --with-python-bindings --with-asserts --with-cyclops
```

Once the configuration has been done, we can build and install

```
$ ./waf
$ ./waf install
```

Once Gaia has been installed, we can start to run Essentia with Gaia linked, and obtain more high level descriptors.

5.3.2. Results with Gaia
In order to obtain these new descriptors, we have to follow the same steps described in part 4.3 of this project, but with Essentia and Gaia built as we have described in the previous steps. Then, we simply execute Essentia and save the new results in a new dataset.

As we can see, the results will be something as it follows:
As we can see, there are many new high-level descriptors, as we mentioned before. Once we have created the new files, we can apply the exact same steps as before: delete all the metadata parts, all the other parts that are not relevant to our dataset, create the csv file and patch it all together and leave it available for Weka.

In Weka, when applying the same algorithms that we applied before, but with the added descriptors from Weka, we obtain the following:

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>462</th>
<th>64.43%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>255</td>
<td>35.56%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.645</td>
<td></td>
</tr>
</tbody>
</table>

Figure 30.- Results obtained from applying SMO to Essentia with Gaia

Figure 29.- Extract from the results obtained with Essentia and Gaia
As we can see, these new results are slightly worse than our previous results, so adding all these high level descriptors didn’t change much to our dataset. There may be many reasons to this failure. One of them is that these descriptors don’t get good results for our scores, and the descriptors that we are obtaining are not relevant to our dataset.

Another reason might be that the new descriptors are good, but when mixing them with other descriptors that are not as good as the others, we are getting mixed results. For instance, we might have a descriptor like this one:

![Figure 32. - Descriptors from High Level - Mood: Acoustic](image)

Here, we can see that most of the songs from romance (green) are accumulated in one part, and some on the other hand; most of the action scores are accumulated to the other one. This might mean that maybe we need to classify our genres differently, and trying to create a tree of binary genres, such as “Romance / Not Romance”, “Action / Not Action”, etc.

5.3.2. **Binary Datasets.**

Another approach to improving the results of our dataset with the calculated descriptors is to change the definition of the different classes. Instead of having the instances belong to one of the
four genres, we can determine whether the instance belongs to a specific genre or not, creating a binary dataset.

**Binary Action**

<table>
<thead>
<tr>
<th>Classification</th>
<th>Count</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctly Classified Instances</td>
<td>276</td>
<td>77.97%</td>
</tr>
<tr>
<td>Incorrectly Classified Instances</td>
<td>78</td>
<td>22.03%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.78</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 33.* - Results obtained from Binary Action using SMO

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>Number of Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>135</td>
<td>42</td>
<td>a = Action</td>
</tr>
<tr>
<td>36</td>
<td>141</td>
<td>b = No</td>
</tr>
</tbody>
</table>

*Figure 34.* - Confusion Matrix from Binary Action using SMO

**Binary Adventure**

<table>
<thead>
<tr>
<th>Classification</th>
<th>Count</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctly Classified Instances</td>
<td>283</td>
<td>78.18%</td>
</tr>
<tr>
<td>Incorrectly Classified Instances</td>
<td>79</td>
<td>21.82%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.781</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 35.* - Results obtained from Binary Adventure using SMO

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>Number of Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
<td>32</td>
<td>a = Adventure</td>
</tr>
<tr>
<td>47</td>
<td>134</td>
<td>b = No</td>
</tr>
</tbody>
</table>

*Figure 36.* - Confusion Matrix from Binary Adventure using SMO
**Binary Animation**

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>291</th>
<th>80.83%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>69</td>
<td>19.17%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.808</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 37.* Results obtained from Binary Animation using SMO

```
a  b  <-- classified as
152 28 | a = Animation
47 501 | b = No
```

*Figure 38.* Confusion Matrix from Binary Animation using SMO

**Binary Romance**

<table>
<thead>
<tr>
<th>Correctly Classified Instances</th>
<th>310</th>
<th>81.58%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrectly Classified Instances</td>
<td>70</td>
<td>18.42%</td>
</tr>
<tr>
<td>F-Measure (Weighted Avg.)</td>
<td>0.816</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 39.* Results obtained from Binary Romance using SMO

```
a  b  <-- classified as
155 35 | a = No
35 155 | b = Romance
```

*Figure 40.* Confusion Matrix from Binary Romance using SMO
As we can see in the four binary datasets, we are obtaining a much better results that with the original dataset, or with the dataset enhanced with the Gaia descriptors. If we look at the percentages and the F-measures of the four datasets, we obtain an accuracy of around an 80%, more than a 10% better than the original dataset, and more than a 20% better than our enhanced dataset with Gaia.

If we take all the accuracies from the four datasets, and we compute the mean, we find a global accuracy of 80.03%, a 16% more accurate than our dataset computed with Gaia, and a 13% better than with the original dataset calculated with Essentia.

### 5.4. Results Analysis

Given all the results obtained in the previous sections, we can analyze them and try to extract some conclusions.

<table>
<thead>
<tr>
<th>Binary Action</th>
<th>Correct</th>
<th>F-Measure</th>
<th>Correct (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Action</td>
<td>276</td>
<td>0.78</td>
<td>77.97</td>
</tr>
<tr>
<td>Binary Adventure</td>
<td>283</td>
<td>0.781</td>
<td>78.18</td>
</tr>
<tr>
<td>Binary Animation</td>
<td>291</td>
<td>0.808</td>
<td>80.83</td>
</tr>
<tr>
<td>Binary Romance</td>
<td>310</td>
<td>0.816</td>
<td>81.58</td>
</tr>
</tbody>
</table>

*Figure 41.* Summarized table with binary results

From the point of view of the descriptors, we can say that, contrary to the initial belief, the low-level descriptors contain more useful information for our dataset than the high level descriptors. One hypothesis for this is that low-level descriptors only contain information from the signal, and it might be that, given the type of tracks that we are working with, we obtain signals that have common characteristics from the songs of the same genre, but don’t have much related regarding high level information, such as mood, or genre. It is true that songs from the same genre should have a common “motive”, but as we have seen, we can find sad, happy, angry songs in all the different genres. Another thing is that we are analyzing movie genres, but the songs don’t have to be from a specific musical genre. This might be a descriptor that confused the dataset, as each movie genre has different musical genres containing it, so we can find that action movies can be contained by electronic music, rock, heavy metal, etc.

From the point of view of the accuracy, we can consider an ~80% of accuracy a good accuracy, given the fact that, as we said before, movie genres are contained by a variety of different music, so it is very difficult to predict which descriptors are going to be useful or not to our dataset. Having
said this, these results suggest that music from Romance is the most clearly distinguishable from the other genres, and Action is the least distinguishable genre from our dataset. This information, but, might be not as accurate as we think, as we are only labeling the soundtracks by one unique genre. The reality is, though, that we might find tracks from an action movie that correspond to a dramatic scene (classifying it as Drama), or that some of the movies might not be entirely from one only genre (for example, Action-Adventure, or Adventure-Drama)
6. REAL TIME CLASSIFICATION

One of the goals of this project was to create a good dataset, and see if, once trained, it can correctly classify new data. So far, we have created the dataset and obtained an accuracy of around 80%. It is time to see if new tracks can be classified, and which is the accuracy of the results obtained.

In order to do this, we will create a program that automatizes all the processes that we have described in the previous sections, and we will wrap it with an interface that can be used as a simpler as possible way to interact with the classifier, and to show meaningful results for the user. In order to do this, we will define a series of steps.

6.1. Data Preparation

In order to analyze a single song against the dataset, we have two differentiated parts to create: the test set, which will be the song in question, and the dataset, which will be the dataset that we have created and calculated its accuracy before, and that we will use in order to compare the test set against all the possible tracks of the dataset.

As we want to compute all the things in real time, we will have to minimize the amount of processing power in order to reduce the process time as much as possible. In our case, we have decided to save the results of training our dataset, and simply call these results every time we want to test a new song. The only problem here is that we are losing the ability to automatically increase our dataset by incorporating the test results into it.

In order to use the test song, we will have to compute its descriptors on real time, and in order to do this, we will have to automate the same process that we have been using in the previous steps. Once the data is processed, we will be able to use it against the trained dataset, and output the results indicating to which genre it belongs the test song.

6.2. Process Design

As we want to test a track against the given train set, and we have determined before that we will use a stored trained set, we have to design the workflow that we will use for both the train and the test set. As both of the sets are obtained in a similar way, we can use most of the code that we have written before and use it for both the test and the train set.

The process then, once the train set has been created, is as it follows:
As we can see, the first three steps are the same that we have already done when creating the train set. This means that the scripts and the python functions created before can be reused when creating the test set.

The SVM Predict will be used in order to calculate the support vectors from our newly created dataset, and be able to compare them to the train set. This will help us create the model, and determine which is the genre that the track has the smaller distance with.

Once the SVM has found the closest genre of our track, we will obtain the results, in the form of a confusion matrix and an accuracy percentage. This will allow us to determine the genre which the song belongs to. Finally, we can make a visual representation of this percentage, making it easier to understand the results.

In order to make all these steps, we will separate the process into three different steps, as it follows:

All the prototype will be done in Processing, as it is an easy choice for prototyping models and easy to create interfaces. Processing is based on Java, so we will be able to use Weka as a Java application, rewriting its code and adapting it to processing, as we will explain further in the next steps.

6.2.1. **Data Gathering**

In order to process the data, we need to locate where the data is. So, first we will need to create an interface which will show the user a prompt in which he will be able to select the new track that he will want to test.
When designing the process, different approaches were considered. First, we thought about creating a button that simply opens a prompt, where the user can select the song and then click start to trigger the rest of the actions.

The design created in order to perform the drag and drop action is as it follows:

As we can see, the action of dragging an element to the central region of the interface will trigger a red circular zone to appear. If the element is dropped inside the red circle, it will be imported to the system. In order to process the song, its path will be stored in a string, in order to know where the file is when processing it, as we can see in the following screen capture:

As we have said before, the path of the file is indeed saved, and which will be used later when we will want to retrieve the path for the data processing.

In addition to that, the action of dropping the file will also trigger a boolean, which will determine if the file is dropped or not (and thus, the string containing the path is saved). This boolean will
indicate that the location of the file is known, and it will send the signal to the other classes that the process of the data can start, as we will see in the following section.

6.2.2. **DATA PROCESSING**

This part will be processed internally, meaning that there is no necessity of a user interface. It will process all the mentioned before steps: the ffmpeg trimming, the Essentia extractors, and the data preparation for the CSV file.

In order to do all these steps, we will reuse most of the steps done when creating the train dataset, but we will adapt it in order to process one single file instead of a dataset of more than 700 tracks. We will also compute Essentia for this file, and then we will adapt the results to a format that can be accepted to Weka, as we did before.

In order to do this, we have created a shell script that will compute all these things all in once for the given file in the first step.

6.2.3. **WEKA CORE**

A Weka Core will be implemented in order to process the test data and compare it to the pre-trained dataset.

In order to do this, we will have to create instances for the test set and the trained set, and then compare them through the classifier model using the algorithm that gave us the best results, which in our case was the SMO algorithm.

In order to do this, we will do the following:
As we can see, the process of the saved dataset is already done, so we just have to create an instance of the Dataset. Then, we have to import the new instance that we want to predict, which we will do by looking for the path of the new data, where we saved it in the script.

Once we have loaded the descriptors from both the dataset and the new instance, we just have to compare them by applying the SMO classifier algorithm. When doing this, we just have to import the saved model. In order to do so, we have to previously saved the classification model as a *.model file.

To save a *.model file, we will use Weka Explorer, as it is what we used to calculate the accuracies with the different classification algorithms.

6.2.4. Outputs
Once we have the Weka Core created and running, it will compare a new track to the existing classification model, and it will calculate its accuracy. We will consider that the genre with the better accuracy will be the genre that we will predict.

These predicted results will have the form of some statistics. These statistics can be interpreted, and then translated to a GUI, giving to the user a percentage of which is the closest genre that the system has predicted for our test song.
6.3. Prototype

As we mentioned before, in order to improve our results, we will use binary test sets instead of the whole dataset. This means that in order to obtain a good prediction, we will have to compare our test song with all the four different saved models.

In order to compare them, we will have to implement a decision tree, following a flowchart like this one:

![Diagram](image)

**Figure 47.** Workflow for our prototype, with four different binary models

As we can see, we will calculate the accuracy for every one of the different saved binary models, in order to give the user the percentages of all the four genres. The main disadvantage of using this method is that we are making the process much slower and hard to process, as we need to make four times the same calculation. But this approach gives us a clear advantage, as we are sacrificing the process speed in order to obtain a much better accuracy on our predictions.

The final prototype consists of three different screens: which will perform the actions:

![Screen](image)

**Figure 48.** Loading screen while the execution of Essentia and Weka takes place
**Input:** it will allow the user to drag and drop the audio file which wants to be evaluated.

**Calculating:** it will show a ‘waiting’ prompt, while the system performs the execution of both the data preparation, and the Weka analysis. The first part is the one that will perform all the heavy work, as it will execute the script preparing the file, which means using Essentia and all the previously created python scripts, and adapt the results so they can be evaluated with the saved Weka Models.

The second part will correspond to the Weka Core. Also invoking weka from a script, it will allow us to read the test file, and compare it with the models. In our case, as we have used binary datasets, we will evaluate the test with the four models. The results of this process will involve something like the results that we can see in figure 49.

![Predictions on test data](image)

**Figure 49.-** Predicted class when comparing test with model

In this case, we took a test track from the action dataset, and evaluated it against the saved model corresponding to the Romance set. As we can see, the results are “Romance”, meaning that this instance does belong to the romance genre.

**Prediction Results:** This screen will show the results, showing the prediction accuracies on each classifier, showing which is the genre that is the closest to the song that we have introduced. This will allow us to interpret the results not as a strong classification, but as an approximation telling us which is the genre that better represents this new instance.
7. EVALUATION

7.1. Results and Limitations

After all the work done, the system has been tested with different songs as inputs, from different genres, even from genres that haven’t been implemented in our prototype, such as ‘horror’ or ‘thriller’.

These tests helped to improve the code used, specially focused on trying to reduce the amount of time necessary in order to perform the computations.

One of the main results that can be discussed in this project is the use of the songs that have been used in the dataset. Different songs, from different movies might have improved or worsened our dataset, thus changing the results of the prediction.

Designing the dataset was a major milestone in this project and a complicated point, as we needed to extract all the information from the songs, and then adapt the output files to a format that could be read by Weka. Choosing the extractor was quite easy, as Essentia seemed to offer the best results given the size of our dataset, but it turned out to be quite hard to build and compile, losing a lot of time into building the system correctly. Another major problem was that Essentia did not give the results in the format that Weka expected, and thus, needed a lot of conversion, not only for the file structure, but also for the organization of the data itself.

Once this milestone was reached, most of the hard work was finished, as the dataset was built and it could be tested with different algorithms.

There are also some limitations to this project. The first one is the usage of Processing. This IDE was chosen in order to create a prototype in a fast and simple way, but it is neither specially stable nor optimal. A lot of processing power is required to operate with Processing, and it would be necessary to migrate to other systems in order to improve the performance of the prototype, such as the Web or the Cloud.

Another limitation is the use of Essentia itself. It is calculated to last more than 15 seconds to analyze and extract the descriptors of a 60 second song. The extractor was used in a stream way, meaning that the songs were computed one after another. A solution to this would be the parallelization of the process, splitting it in half or in a fourth.

Our main limitation now is the time taken for a single song to compute both Essentia and compare the results to the Weka core. The descriptors model has already been saved in order to save time, but it is a reality that the system is not a fast one, specially when calculating Essentia’s descriptors.

Although these limitations, most of them do not affect the results of the prototype, and are expected to be resolved in future versions of it.
7.2. Applications

There are many applications that can use this system in a direct way or in an indirect way. This system will allow us to automatically determine the musical genre of a movie by listening to its soundtrack.

This could be useful in the industry where the amount of movies and soundtracks is enormous, and there is a need to correctly classify both new and existing movies in order to organize the huge databases that online video platforms work with.

An example of these existing platforms would be Netflix, iTunes, or imdb, which would see their classification process shortened if they automatized the problem of having to classify huge amounts of movies all in once.

These services could then use this classifier as a recommender, which would help you choose your next movie basing its results in the soundtrack of previously watched movies. This could be done by analyzing different parts of the movie in real time in a background process in order to obtain an even more accurate result. This would also have the benefits of knowing the tastes of the customer, and helping determine what is the impact of a soundtrack in the movie.

Another application that could be created from this classifier would be a tool in order to help composers to create a soundtrack that adapts the best to the genre of the movie, having in account previous soundtracks from other movies. This would allow the composer to create original soundtracks and then analyze if they would be considered in a certain genre, and if not, being able to guide the composer into which aspects would need to be changed in order to belong more in a certain genre. This application would also allow us to determine if a specific song is adequate for a specific genre.

Another application would be a classic ‘genre identifier’ app, in which the user could point their mobile phone into some movie, and the system could determine the genre of it. This is maybe the most direct implementation of the system, and may not be viable, as there is little use in an app that tells you into which genre a movie belongs to.
8. CONCLUSIONS

By evaluating the work done, all the goals set for this project have been accomplished.

In this project, an automatic classification system has been developed. In order to do this, different parts have had to be implemented.

Once we have finished the project, and evaluated the results, we can analyze which have been the strong parts, and which have been the ones that delayed some of the milestones of the project.

In our case, the first big problem that we encountered was the lack of previous work done by other scientists, making it quite hard to compare results or even see what other had done before me. In the beginning, one major problem was also the lack of information from the history of movie soundtracks that helped me understand which are the main features of a soundtrack, and specially the characteristics that could define a soundtrack from a specific genre. Although having found Roy M. Pendergast’s “Film Music: A Neglected Art”, which has been a great source of information, specially when understanding the social background behind every genre and every film era, most of the information in which this project relies on is the one gathered from my own experience, and hours of dedication listening to a lot of soundtracks, and trying to figure out which are the characteristics of each musical genre. These results then has been shared with friends and members of my family, in order to contrast the results, reaffirming some of them and discarding others.

Another of the problems encountered in this project was found during the process of extracting the features of the soundtrack. At first, Sonic Annotator was used, giving quite a nightmare in the output, as one file was created for each one of the descriptors of every song, making the acquisition of data quite impossible using this tool. Then, we decided to change the workflow and use Essentia instead, resulting in a more comprehensive output, and making the extraction more agile.

Changing the process of extraction once the project was already started delayed the deadlines several weeks, as the program needed to be learnt, compiled and built, then, the outputs needed to be studied again in order to understand how the results were written and which were the necessary adaptations that needed to be made in order to adapt the output files to a file accepted by Weka. Although the delays in the initial deadlines, changing from Sonic Annotator to Essentia greatly improved the process of extraction, and made it easier later to process the files for Weka.

Although all these drawbacks, analyzing the final results and the goals being achieved, I feel that I have applied a lot from the four years of my engineering degree, deepen my knowledge in concepts that were defined in some courses, and what’s most important, I feel like I have learned even more new things, such as coding in Shell, learning python, working with Weka, understanding more about Music Information Retrieval, and learning a lot about cinema and specially about soundtracks.

It is quite amazing that although cinema is quite an old media, its soundtracks are something that have stayed so classic in form, specially when there are many film genres, making the variety so big.
8.1. Future Work Development

As we mentioned before, our system has some limitations. This project will not end here, as we will try refine all the things that we think that could be improved, and also add more features to it, improving its accuracy and also expanding its capabilities.

In order to improve its accuracy, we have tried to think on different solutions that could be added to our system and could lead to better results. It has been considered to label different the movies, making them more precise. Given our genres, we could make sub-genres, defining “Romance-Drama”, “Romance-Comedy”, or “Action-Thriller”. This would mean that movies would be more accurately labeled. We could have more results, as we would have many different options to chose from, augmenting the ‘resolution’ of our results. As we now would have more labels, we could wide our range of movies per genre, and add more films to our dataset, augmenting the variety of tracks, and thus, creating a dataset with more averaged descriptors.

Another approach that could be taken in order to improve our results would be to carefully review our complete list of movies and soundtracks, and choose movies that belong in a more ‘purely’ way to the desired genre.

In order to improve the results even more, a thorough analysis should be made in the descriptors of each instance, keeping the ones that are relevant to our dataset, and ignoring the ones that are not relevant at all, or that are even adding confusing information to our dataset. In order to do this, we should use Weka’s Explorer and filter the attributes that are relevant. There are many automatic ways of selecting the attributes that have a better rate, but in our case, and with so many descriptors, we should go one by one and analyze what does each attribute contribute to the global results.

Finally, one future work that will be definitely considered is the fact of not only analyze the soundtrack of a movie, but also analyze its photography, expanding the capabilities of the system to not only sound, but also image.

In order to do this, we would have to analyze the whole image of the movie, or at least a big part of it, so we can have meaningful information about the light and color of the picture. One way to do this could be by analyzing each frame every minute, and composing a global picture with all the frames selected. Then, we could analyze this new picture, and extract its predominant color, the hue, the saturation, the brightness, etc. Following the previous work done by moviebarcode \[17\], we could obtain something like we see in Figures 46 and 47.

In conclusion, this project has helped to understand
the basis of Music Information Retrieval, and has set some of the basis about music in films, allowing to go further on the relation between soundtracks and genres.
9. REFERENCES


ANNEX

Annex I. Glossary

Here, we will describe some of the most used technical and musical words used in this project, and that might be of good use when reading this paper, sorted in an alphabetical order.

AMPLITUDE measures the different measures that are different than zero in a sinusoidal signal.

CEPSTRUM is the inverse of the spectrum, hence the name, which is \textit{spectrum} with the four first letters reversed. It is computed by applying the inverse Fourier Transform of the logarithm of the spectrum of the signal.

DATASET is a collection of data, filled with all the descriptors.

DESCRIPTOR is a structure containing all the information from some data. In our case, we can use descriptors in order to store the information obtained from the signals, such as frequencies, etc.

ENVELOPE makes reference to the changes of the spectral content of the sound over time. We will have four differentiated stages: Attack, Decay, Sustain and Release. Each Instrument has a different ADSR relation, which determines the timbre.

FEATURE refers to an attribute of the dataset.

FOURIER TRANSFORM is a mathematical transformation that allows us to transform signals between time domain and frequency domain.

FREQUENCY can be defined as the number of repetitions of occurrences per unit of time. In our case, frequency makes reference to the sinusoid of the sound. The higher the frequency, the narrower becomes the sinusoid, augmenting its pitch. Human ears can hear mostly from 20Hz to 20KHz.

(j) FREQUENCY DOMAIN refers to the analysis of signals with respect to their frequency rather than their time. It allows us to show how much energy lies within each frequency of the signal, and includes information about the phase shift that must be applied to its sinusoid in order to recover the original file.

INSTANCE is a singular item of the dataset. In our case, an instance is a song, with all its descriptors.

KEY usually refers to the tonic note. It defines the pitch upon which all other pitches of a piece are hierarchically referenced. Scales are named after their tonics.

MEL SCALE is a perceptual scale of pitches judged by listeners to be equally distanced from one another. It showed that in order to equal pitch increments, the frequency intervals must be larger and larger when the frequency augments, giving as a result a logarithmic function.
MELODY refers to a linear succession of musical tones (or pitches).

PEAK refers to the highest amplitude of a signal.

PITCH is a perceptual property that orders sounds by their frequency. It is usually associated with musical melodies, and it is an attribute of musical tones.

PHASE refers to the initial angle of a signal. It can also mean the fraction of the wave cycle which has elapsed respect the origin (PHASE SHIFT).

SPECTRUM A frequency spectrum of an audio signal is the representation of the signal in the frequency domain. It allows us to interpret signals as a combination of frequencies, allowing us to process and modify the signal. The change from a temporal domain to a frequency domain is usually performed by Fourier Transforms.

TIMBRE refers to the quality of a sound tone, allowing us to distinguish different types of sound productions, as can be voice, string instruments, wind instruments, etc. It is the characteristic that allows us to distinguish one instrument from another, even when both have the same pitch and loudness. Timbre can be found with the spectrum and the envelope of the signal.

TEMPORAL DOMAIN refers to the analysis of signal with respect to their time. In time domain, we can see how the signal changes over time.
## Annex II. Movie List

This is the list with all the movies included in the dataset for our project.

<table>
<thead>
<tr>
<th>Romance</th>
<th>Adventure</th>
<th>Action</th>
<th>Animation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(500) Days Of Summer</td>
<td>Prometheus</td>
<td>Resident Evil</td>
<td>Brave</td>
</tr>
<tr>
<td>Moonrise Kingdom</td>
<td>The Hunger Games</td>
<td>The Expendables</td>
<td>Toy Story 1</td>
</tr>
<tr>
<td>Crazy, Stupid, Love</td>
<td>LoTR: The Fellowship of The Ring</td>
<td>Taken</td>
<td>Toy Story 2</td>
</tr>
<tr>
<td>Le fabuleux destin d'Amélie Poulain</td>
<td>LoTR: The Two Towers</td>
<td>xXx</td>
<td>Toy Story 3</td>
</tr>
<tr>
<td>Juno</td>
<td>LoTR: The Return of The King</td>
<td>Safe</td>
<td>The Lion King</td>
</tr>
<tr>
<td>Adventureland</td>
<td>Inception</td>
<td>Battleship</td>
<td>Pocahontas</td>
</tr>
<tr>
<td>Beginners</td>
<td>Thor</td>
<td>The Matrix</td>
<td>Cars</td>
</tr>
<tr>
<td>Love Actually</td>
<td>Captain America: The First Avenger</td>
<td>The Matrix Reloaded</td>
<td>Monsters, Inc.</td>
</tr>
<tr>
<td>The Holiday</td>
<td>Iron Man</td>
<td>Transporter</td>
<td>Ice Age</td>
</tr>
<tr>
<td>Bridget Jones's Diary</td>
<td>The Avengers</td>
<td>The Terminator</td>
<td>How To Train Your Dragon</td>
</tr>
<tr>
<td>Seeking a Friend for the End of the World</td>
<td>X-Men</td>
<td>James Bond Saga</td>
<td>Tangled</td>
</tr>
<tr>
<td>Eternal Sunshine of the Spotless Mind</td>
<td>Batman Begins</td>
<td>Transformers</td>
<td>The Adventures Of Tintin</td>
</tr>
<tr>
<td>La Vita è Bella</td>
<td>The Dark Knight</td>
<td>300</td>
<td>Finding Nemo</td>
</tr>
<tr>
<td>PS. I Love You</td>
<td>The Dark Knight Rises</td>
<td>Mission Impossible</td>
<td>Up</td>
</tr>
<tr>
<td>Up In The Air</td>
<td>Avatar</td>
<td>Underworld</td>
<td>WALL·E</td>
</tr>
<tr>
<td>Forgetting Sarah Marshall</td>
<td>Star Trek</td>
<td>Saving Private Ryan</td>
<td>Wreck-It Ralph</td>
</tr>
<tr>
<td>Remember Me</td>
<td>Sherlock Holmes</td>
<td>The Bourne Identity</td>
<td>Kung Fu Panda</td>
</tr>
<tr>
<td>Romeo + Juliet</td>
<td>Sherlock Holmes: Game of Shadows</td>
<td>The Bourne Supremacy</td>
<td>Madagascar</td>
</tr>
<tr>
<td>Brokeback Mountain</td>
<td>Troy</td>
<td>The Bourne Ultimatum</td>
<td>The Incredibles</td>
</tr>
<tr>
<td>Pretty Woman</td>
<td>Clash of The Titans</td>
<td>Die Hard</td>
<td>Ratatouille</td>
</tr>
<tr>
<td>Vicky Cristina Barcelona</td>
<td>Hugo</td>
<td>Black Hawk Down</td>
<td></td>
</tr>
<tr>
<td>Revolutionary Road</td>
<td>Into The Wild</td>
<td>Colombiana</td>
<td>Aladdin</td>
</tr>
<tr>
<td>The Reader</td>
<td>Gladiator</td>
<td>The A-Team</td>
<td>Shrek</td>
</tr>
<tr>
<td>Moulin Rouge</td>
<td>Riders of The Lost Ark</td>
<td>Fast&amp;Furious</td>
<td>The Little Mermaid</td>
</tr>
<tr>
<td>Eat Pray Love</td>
<td>Pirates of The Caribbean</td>
<td>Independence Day</td>
<td>A Bug's Life</td>
</tr>
<tr>
<td>Crazy Heart</td>
<td>Alice In Wonderland (Tim Burton's)</td>
<td>The Mechanic</td>
<td>The Nightmare Before Christmas</td>
</tr>
<tr>
<td>Romance</td>
<td>Adventure</td>
<td>Action</td>
<td>Animation</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------</td>
<td>-----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>How To Lose a Guy in 10 Days</td>
<td>Spider-Man</td>
<td>The Rock</td>
<td>The Simpsons Movie</td>
</tr>
<tr>
<td>Cold Mountain</td>
<td>Alien</td>
<td>Mr &amp; Mrs Smith</td>
<td>Robots</td>
</tr>
<tr>
<td>New York, I Love You</td>
<td>Tron: Legacy</td>
<td>The Matrix Revolutions</td>
<td>Antz</td>
</tr>
<tr>
<td>Memoirs Of a Geisha</td>
<td>Serenity</td>
<td>Pearl Harbor</td>
<td>Chicken Run</td>
</tr>
<tr>
<td>Shame</td>
<td>Jumper</td>
<td>Crank</td>
<td>Peter Pan</td>
</tr>
<tr>
<td>Pride &amp; Prejudice</td>
<td>Robin Hood</td>
<td>Green Zone</td>
<td>Bambi</td>
</tr>
<tr>
<td>Dracula</td>
<td>Star Wars</td>
<td>Kill Bill</td>
<td>Pinochio</td>
</tr>
<tr>
<td>One Day</td>
<td>True Grit</td>
<td>Inglorious Basterds</td>
<td>The Brave Little Toaster</td>
</tr>
<tr>
<td>My Best Friend's Wedding</td>
<td>The Mummy</td>
<td>Conan</td>
<td>Alice in Wonderland</td>
</tr>
<tr>
<td>In The Mood For Love (Fa yeung nin wa)</td>
<td>Planet of The Apes</td>
<td>Lethal Weapon</td>
<td>Hercules</td>
</tr>
<tr>
<td>Notes on a Scandal</td>
<td>Kingdom of Heaven</td>
<td>Top Gun</td>
<td>The Sword in the Stone</td>
</tr>
<tr>
<td>Atonement</td>
<td>King Arthur</td>
<td>Deep Blue Sea</td>
<td>The Jungle Book</td>
</tr>
<tr>
<td>Dark Shadows</td>
<td>King Kong</td>
<td>Rambo</td>
<td>Sleeping Beauty</td>
</tr>
<tr>
<td></td>
<td>The Chronicles of Narnia</td>
<td>Air Force One</td>
<td>Cinderella</td>
</tr>
<tr>
<td>Harry Potter</td>
<td></td>
<td>The Jackal</td>
<td>Snow White and the Seven Dwarfs</td>
</tr>
<tr>
<td>Superman Returns</td>
<td></td>
<td>The Negotiator</td>
<td>Beauty and the Beast</td>
</tr>
<tr>
<td>Monty Python and The Holy Grail</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cast Away</td>
<td></td>
<td>Wolverine</td>
<td></td>
</tr>
<tr>
<td>Fantastic Four</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alexander</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Day After Tomorrow</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Master and Commander: The Far Side Of The World</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Mask Of Zorro</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Hobbit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lawrence of Arabia</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Annex III. Descriptors

The descriptors used for this project of each one the movies can be found at the annex file of this project, under the folder called ‘Descriptors’. In the folder, we can find all the descriptors generated for each track of the dataset, and the descriptors all together in various datasets, such as the binary ones.
Annex IV. Code

The code used for every part of this project, both calculating the dataset, creating the shell scripts, the python code, and the processing sketches for the prototype can be found at Bitbucket’s repository https://bitbucket.org/pauboix/movie-classifier