Automatic accompaniment for improvised music

Daniel Martín

MASTER THESIS UPF / 2009
Master in Sound and Music Computing

Master thesis supervisor:
Rafael Ramírez
Department of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona

UNIVERSITAT POMPEU FABRA

Abstract
Modern music students practice improvisation playing on top of a software that generates accompaniment, but it does not listen to the musician's input. Automatic accompaniment systems provide appropriate musical accompaniment to a human player in a given musical context by listening to them. We are interested in developing an AAS. Therefore, the purpose of this project is to build a system that generates accompaniment for a given melody, emulating a jazz pianist accompanying a human soloist improvising in jazz style. The method used is based on re-using stored fragments to create the accompaniment. Two musical fragment representation models have been studied: a tree representation model and a statistical model, from which the latter has been implemented.

Contents

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Objectives 2
- 1.3 The system 2
- 1.4 Organization of the thesis 2

2 State of the art in accompaniment systems **5**
- 2.1 Classification distinctions 5
  - 2.1.1 Audio vs. MIDI input 6
  - 2.1.2 Interactive vs. Reactive 6
  - 2.1.3 System's knowledge 6
  - 2.1.4 Music knowledge based vs. machine learning 6
  - 2.1.5 Others 7
- 2.2 Taxonomy 7
  - 2.2.1 Score followers 7
  - 2.2.2 Automatic arrangement systems 8
  - 2.2.3 Automatic Harmonization systems 8
  - 2.2.4 Improvisation systems 9
  - 2.2.5 Accompaniment systems related to jazz 10

3 Background **17**
- 3.1 Machine Learning 17
  - 3.1.1 Instance-based Learning 17
  - 3.1.2 Multilayer Perceptron 18
  - 3.1.3 Sequential Minimal Optimization (SMO) for Support Vector Machines (SVM) 19
  - 3.1.4 M5 Rules 20
- 3.2 Music theory 21
  - 3.2.1 Jazz 21
  - 3.2.2 Piano comping 21

4 Our system **23**
4.1 Overview 23
4.2 How the system works 24
4.2.1 Database 25
4.2.2 Fragmentation 26
4.2.3 Fragment description and retrieval 28
4.2.4 Transformation 32
4.3 Implementation details 34

5 Results **35**
5.1 Learning the model 35
5.2 Creating accompaniment evaluation 36

6 Conclusions **41**
6.1 Further work 42

References **42**

Chapter 1

Introduction

1.1 Motivation

When musicians play music together they listen to each other and interact among them; each musician's performance influences on the others. Musicians can either follow a score or improvise. In the first case, the notes to be played by each performer depend on the score, but other musical aspects like tempo or dynamics depend on the rest of the band's performance. On the other hand, when musicians are improvising, there are no predetermined notes to be played. Notes actually played depend on the performers themselves, although these are following a fixed harmonic structure. Usually, there is a soloist that improvises while other musicians accompany him/her. In that case, the soloist tends to execute his musical discourse while the rest of the band plays according to that discourse; furthermore, the soloist can be influenced by the other musicians.

There is a popular software used by modern music students called *Band in a Box* [Gan91] that receives as input a chord grid, some instruments, tempo and genre specification and generates accompaniment as output in MIDI format. Musicians can practice improvising on top of that accompaniment. Many music teachers agree that the problem of this software is that its output is poor because of two main reasons. First, output is very predictable, it is not affected by the musician's performance; and second it is a MIDI output, so the quality of the sound depends on the sound card, which often results in bad quality output. Our intention is to focus on the solution of the first problem: to develop a system that should be able to *listen* to the human musician and accompany him in an adequate way.

The idea of a computer 'listening' to a human performer and playing together with him is not new. In the music technology field, this has been explored by many researchers and a significant variety of systems have been developed using different approaches in several music genres and emulating many musical instruments. We call these systems 'Automatic Accompaniment Systems' (AAS). AAS are systems that provide a musical response to a human performer.

Our initial motivation was to come up with a system which accompanies a soloist improvising in real time. We have focused on polyphonic accompaniment, in particular the system consists of a piano emulator that plays according to the soloist performance in jazz music following a fixed harmonic structure.

1.2 Objectives

While the long-term goal is to build an agent to play with in real-time, the immediate goal is to develop a method that generates appropriate accompaniment. Focusing on this immediate goal, we will explore different ways to generate musical response to a given melodic input, studying advantages and drawbacks of different approaches.

1.3 The system

We focus on jazz because it is a music genre in which improvisation and interactivity between musicians are very important. The system will emulate a piano player that follows the same harmonic structure as the soloist, the harmonic structure being determined by a chord grid.

In order to do this we will collect a database of performances in which for each of them we have:

- The melody played by the soloist.
- An accompaniment played by a piano.
- A chord grid.

The general idea is to somehow associate the accompaniment lines with the melodic lines and to establish a relationship among them so that the system is able to generate a new accompaniment line for a given new input; this input consists of a melody line and a chord grid. Therefore, two key issues in the system are the description of the melodies and accompaniments, and the similarity measures between those descriptions.

1.4 Organization of the thesis

Chapter 2 describes the state of the art in accompaniment systems and presents a taxonomy for these systems. Chapter 3 describes the musical and technical background to our research. In Chapter 4 we describe our approach to accompaniment systems. Chapter 5 discusses some results, and finally in Chapter 6 we present some conclusions and future work.

Chapter 2

State of the art in accompaniment systems

Automatic accompaniment systems (AAS) are computer systems that provide a solution to the problem of finding an appropriate accompaniment for a soloist.

Although the word *accompaniment* has been often used to refer to a particular subclass of AAS (concretely, score followers, which we will describe later), we refer to accompaniment in its musical meaning: the art of playing along with a soloist or an ensemble in a supporting manner.

Since the early days of computers, many AAS have been developed. In this chapter we present an overview of them.

2.1 Classification distinctions

AAS accompany a melodic line that is usually performed by a human soloist. The accompaniment is performed in real time yet there are some exceptions. AAS are real-time *interactive music systems*, and more concretely, *machine listening* [Jor05] systems. Interactive systems are systems that use information provided by the human to generate a response, and the other way round: information provided by the computer will determine how the human will proceed. This ex-change of information requires two channels through which human and computer will communicate.

Interactive music systems are those interactive systems that produce music as output; whereas *machine listening* systems are those interactive music systems whose channel input from a human performer is a musical channel; that is, audio or MIDI.

AAS may be characterized using different criteria and can be categorized in many different ways (e.g. [Jor05], [Dav07], [Row93]). Here we discuss the most important criteria to classify them.

2.1.1 Audio vs. MIDI input

AAS determine their output by a given input from the human performer. This input, as in all machine listening systems, is a musical input that can be either audio or MIDI. Processing MIDI is simpler and easier as the pitch and onset are already described, whereas for audio input, pitch and onset detection techniques are required.

2.1.2 Interactive vs. Reactive

The difference between an interactive and a reactive system is that while in the former there is a mutual influence between the system and the human performer, in the latter the system just reacts to the human's input, but the system's output has no influence on the human musician's performance. However, it is not a binary classification; there are different levels of interactivity. The distinction between interactive and reactive is correlated with a real-time vs. non real-time classification. Non real-time systems are never interactive, whereas real-time systems can be either interactive or reactive. We consider AAS to be real-time systems. The only non real-time systems we will discuss are automatic arrangement systems, because they are very close to AAS as they provide a harmonic accompaniment to a given melody.

2.1.3 System's knowledge

System's knowledge classification refers to what the system knows about the human's performance. Davies [Dav07] distinguishes between systems that accompany a human improvising and those that follow a human performing a score. However, the point is whether the system knows or not what the performer is going to play. If the computer has no information about the human's performance, the computer cannot distinguish if the human is playing a given score or improvising. Therefore, we will classify AAS according to the system's knowledge regarding to the human's input. Basically we can distinguish between three types of AAS: first, systems that have the complete information about the human's performance (whereby the human has to be reading a score when performing); second, systems that have information about the harmonic structure, i.e. the chord grid; and third, those that have no information. This classification is related to the score-driven vs. performance-driven dimension of Rowe's taxonomy [Row93].

2.1.4 Music knowledge based vs. machine learning

After processing the input from the human, AAS have to generate an appropriate output. To do so, they can make decisions based either on musical knowledge that someone has introduced in the system or on knowledge obtained by training the system with several examples. Also, both approaches can be combined.

2.1.5 Others

Finally, other classifications can be considered. Davies [Dav07] distinguishes between systems that consider time variations and those that use quantized time. Also, the output of the system can be classified into rhythmic, melodic and harmonic. We can say harmonic output is a vertical output, such as chords (in which notes are played simultaneously); whereas melodic output is horizontal, that is, a sequence of consecutively played notes, such as in solo trading between human and computer. The last distinction to be considered is the music style in which the system will focus; it can either be electroacoustic music, classic music, jazz, bossa-nova, etc.

2.2 Taxonomy

In this section we give an overview of the different categories of AAS and the systems developed.

2.2.1 Score followers

Score followers are a kind of AAS that have the information of the score the musician is going to play. They follow a score in real-time, while the performer is playing it.

Regarding interactivity, score followers are reactive [Jor05] rather than interactive because the performer has to follow a given path. It is just inverting the role of the music minus one case where the musician follows a tape. In fact, if we think about an extreme situation in which the performer would play without listening to the computer's response, the result would be the same. The point is that the computer's response can influence the human's output, but this output should not influence anymore the computer. So mutual influence should not be longer than two steps; otherwise a collapsing effect in which for instance both components try to play faster and faster could happen.

One of the interests of developing this kind of systems is to allow electroacoustic musicians to perform their pieces where they mix a traditional instrument and tape in real-time.

There have been many score followers developed since Dannenberg and Vercoe started in the 1984. Initially the input was audio input, although Vercoe's Synthetic Performer [Ver84] used some finger information provided by optical sensors. Later, on 1985 Vercoe used MIDI input. Some other examples are Norbert Schnell's

IRCAM Score follower [SCS] or Christopher Raphael's system [Rap06], in which a human performer playing oboe was followed by an orchestral recording.

2.2.2 Automatic arrangement systems

Automatic arrangement systems are those that concentrate efforts in finding out how to accompany a melody given the chord grid information, and therefore, to fill the gap between the chord grid and the final accompaniment performance. They are not a subclass of AAS, as they are not in real-time. Those systems that determine the accompaniment in real time are improvising AAS, we describe them later. Arrangement is a musical task consisting in rewriting an existing piece with new material. In music, arrangement is never done while performing. The equivalent to arranging in real time would be improvising accompaniment.

The most popular automatic arrangement real-time system is *Band in a Box* [Gan91]. It is a MIDI sequencer used by many jazz students to play and improvise over any standard. Because it is MIDI, the user is able to change many parameters like change tempo, key, style (so that rhythmic patterns will change). It is not interactive neither reactive since it has no 'ears' to listen to the soloist.

Another interesting arrangement system is D'accord guitar [CZL+01], its output to a given chord grid and melody is the actual performance in real-time of a guitarist is showed in a screen. It is useful to learn how to play the guitar but there is neither musical effect nor interaction.

Finally, there is a pretty recent system by Emura et al. that generates jazz-style arrangement for a given set of a melody and its chord name sequence [EMY08]. The approach is based on music knowledge; harmonic theory of jazz and knowledge about piano voicings are used to define some constraint the will be taken into account when generating an accompaniment. Further, the system accepts some user requirements that will also determine the output of the system.

2.2.3 Automatic Harmonization systems

Whereas score followers are systems that know both what to play and what the performer is going to play, on the opposite side, there are the Automatic Harmonization systems. These systems try to find the best harmonic accompaniment to a melody with no information about what the musician is going to play neither about the harmonic structure. It is important to point out that they focus on the harmonization, i.e. in finding an appropriate harmonic context for a given music style. Automatic Harmonization systems can be either real-time or not. Real-time automatic harmonization systems are a type of AAS, whereas non real-time automatic harmonization systems are arrangement systems.

Cabral [Cab08] PhD thesis' topic was automatic harmonization in real-time. He uses machine learning techniques among others like mosaicing in order to generate automatic accompaniment. Since Cabral's work is focused on bossanova music, he tries to obtain tonal harmonic accompaniment with good results.

MySong [SMB08] is a system commercialized by Microsoft, with the same goal, but focused on vocal music. Both, Cabral's work and MySong analyze audio input. The level of interactivity of these systems is low as their goal is just to provide a harmonic context to a given melody. However, as it is real-time accompaniment, the system's output could influence the human's performance.

2.2.4 Improvisation systems

Another subclass of AAS is the one that includes those systems in which there is no information that the computer previously knows about what the musician is going to play, although they can have information about the context. The human musician is expected to improvise, so the system cannot know the exact notes the human is going to play. The difference between improvisation that have no harmonic context information and harmonization systems is that the former are focused on performing together with a human musician rather than finding an appropriate harmonic accompaniment.

As Thom [Tho01] points out, improvisation systems can be based on *author-the-aesthetics* paradigm; that means that the decision of the system to generate the output depends on the programmer or user's configuration. On the other hand, other systems can have their own aesthetic criteria learned by being trained using some Machine Learning techniques.

The most important early example of improvisation system is G. Lewis' *Voyager* [Lew00]. Lewis was a free-jazz trombone player and he wanted to develop a system that emulated a 'partner' to play with himself. In this system, the computer analyzes the trombone's performance (through a pitch-to-MIDI converter) in real-time and plays itself accordingly. The author-the-aesthetics paradigm is more present here as the rules that the system follows to make the decision are defined by the author.

Robert Rowe developed *Cypher* [Row93]. This system processed MIDI input of a human player in order to understand human's play; it was based on Marvin's Minsky Society of mind (1986).

In 2003 Pachet came up with the *Continuator* [Pac03], a virtual improviser that listens to a performance of a MIDI keyboard player, and gives a response, imitating the player's style. To do so, the Continuator focuses on certain features that it extracts from the player's music and creates a learned tree that it will use later to produce a musical output.

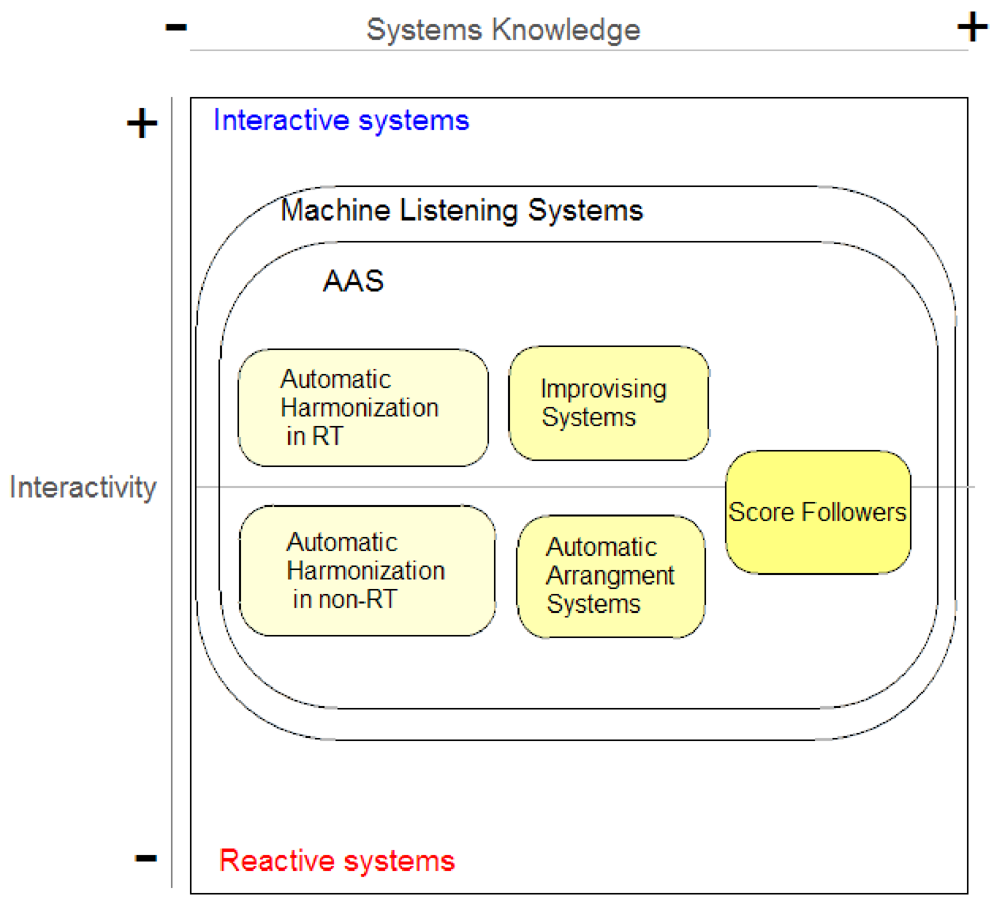

-
Systems Knowledge
+
+
Interactivity
-
Interactive systems
Machine Listening Systems
AAS
Automatic Harmonization in RT
Improvising Systems
Score Followers
Automatic Harmonization in non-RT
Automatic Arrangment Systems
Reactive systems

Figure 2.1: Classification of AAS

2.2.5 Accompaniment systems related to jazz

Since we are talking about improvised music, jazz is one of the music styles in which improvisation has a most important role. AAS related to jazz are based either on autonomous aesthetics obtained by Machine Learning techniques or on musical knowledge.

Band out of a Box

Belinda Thom's BoB [Tho01] (*Band out of a Box*, 1999) is a virtual improvisational partner. The name *Band Out of a box* refers to the popular software *Band in a box* [Gan91], which it is not an interactive system as we discussed in Section 2.2.2. BoB is a system that learns how a musician improvises in a learning stage. Once it has created a model by machine learning techniques, the system trades solos in real-time with the performer. Further, the system can learn a user-specific playing mode, so it can imitate a certain musician. The system is a horizontal AAS because it provides a response to the soloist as a solo dialog rather than accompanying harmonically the soloist.

Detailed description:

Thom's system works in two stages: a learning stage called *Listener*, in which a user-specific playing mode is learned; and a playing stage called *Generator*, when the system trades solos with a human performer.

Listener:

In this stage, the input of the training examples needs to be described. The input melody is fragmented per bar and the fragments' description is done in two steps. The first one is representing the input melody as a Variable Length Tree (VLT). And the second is obtaining a conglomerative set of features CGL from those VLT. In the VLTs, the leaf nodes encode pitch transformation whereas the internal structure encodes duration information. They are variable length because the number of nodes om the tree will depend on the number of notes in each bar.

For the CGL set of features, the sequence of pitches from the leaves in the tree is converted in three fixed-size histograms: a pitch-class histogram (tonality), an interval histogram (continuity), and melodic direction (contour). These histograms will capture deeper levels of structure than the melody surface.

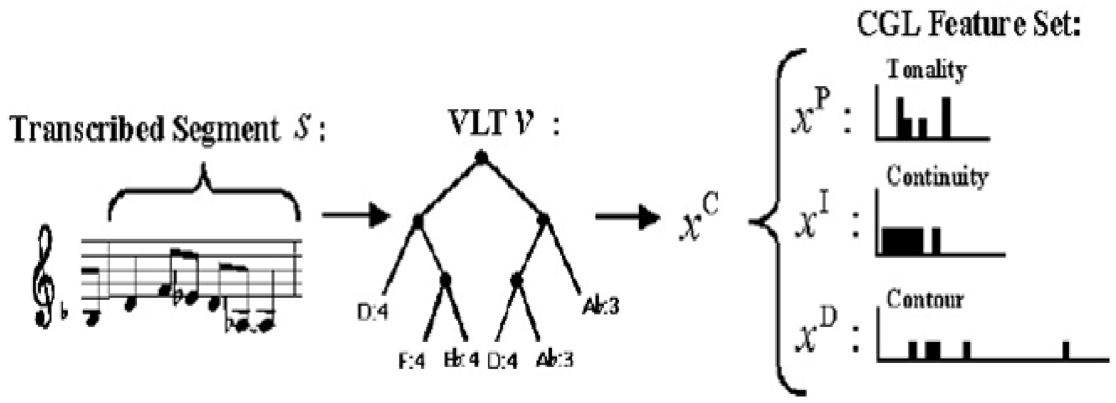


Figure 2.2: BoB's Melody representation

All histograms are classified into clusters called *playing modes*. This clustering is made with a Variable-sized Mixture of Multinomials (vMn) model which is learned from the histograms by using an Expectation-Maximization (EM) strategy. The learned model is used to classify the data into the playing modes.

Generator:

The generator receives the human musician's melody v as input and generates an output v'. To do so, there are three steps:

- Rhythmic transformation; in which the VLT of v is transformed by applying two types of transformation: embellishment (growth) and simplification (collapse). The decision of what concrete transformation to use is stochastic.

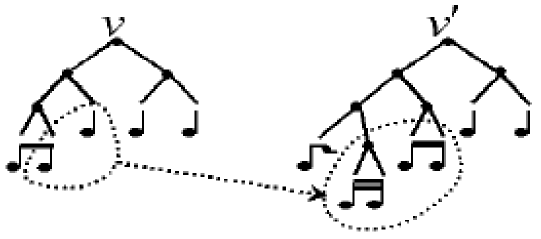


Figure 2.3: Embellishment

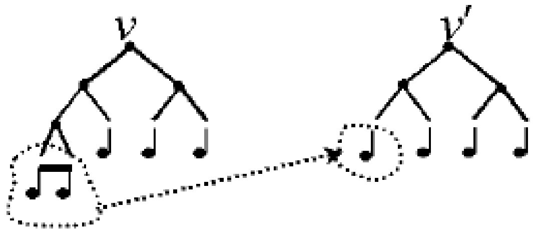


Figure 2.4: Simplification

- Generation of a pitch sequence for the leaves of the transformed VLT. This generation is goal-driven, and is executed under a set of constraints from the learned model in the Listener stage; it also takes into account the most recent context.

- Finally each pitch in the generated sequence is assigned to a leaf of the VLT.

VirJa session

Another AAS for jazz was presented in 1995 in ICMC by I. Hidaka, et al. [HGM]. It is a system that reacts to a Solo in real time. MIDI is the communication channel between human and computer. In this system, the soloist intention is obtained and used to alter the accompaniment.

Detection of primitives

At each beat some musical primitives are detected by analyzing chord and key; these primitives are *tension note*, *chord note*, *scale note* and *other chord note*; also, other primitives are detected directly: *louder note*, *higher note* and *many notes*. Some of the primitives are obtained directly by MIDI information (i.e. loudness). Others are the result of analyzing the relationship between the notes played and the harmonic context using some music theory information. This is possible because the system already knows the harmonic context, which is defined by a chord grid.

Extraction of intention parameters

All these primitives are used to extract some parameters that will define the *intention* of the soloist. This is done each certain period time. The intention parameters are:

- excitement.
- tension: soloist's intention of playing a little different from a standard by playing tension notes
- emphasis on chord: soloist's is trying to emphasize the chord progression by playing notes of the chord.
- chord substitution: soloist's emphasizes on a chord that substitutes the original current chord which has the same functionality.
- theme reprise: soloist's plays a faking theme that can be recognized as a standard.

Alteration of the accompaniment

The extracted soloist's intention parameters are used to control to modules that will alter the system's output:

- *MIDI filter*: adjusts loudness and pitch
- *MIDI phase shifter*: adds notes by shifting timing of the original notes.

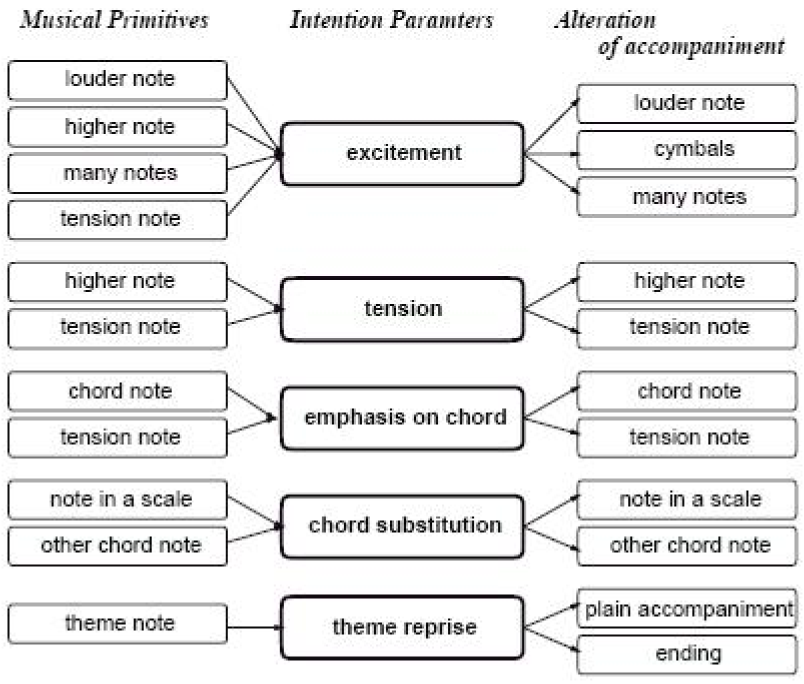


Figure 2.5: Hidaka's system. Soloist's intention

Learning based Jam session system for a Guitar Trio

A few years later, in 2001, M. Goto et al. [HGO] presented a similar system that extended the above one: this one was able to imitate the style of any musician, or acquiring a musician's *reaction model*, as they called it. The system had two modes:

- in the learning mode, the system acquires player personality models in non-real time
- in the session mode, system interacts with a human musician

The reaction model is composed by the impression space and the intention space. The *impression space* represents subjective impression with three dimensions: appealing, energetic and heavy. To obtain an impression vector in this space, canonical correlation analysis (CCA) was used. On the other hand, the intention space represents the intention of the user in a similar way as in *VirJa session*. Finally, the mapping from the impression space to the intention space gives as result the reaction model.

ImPact

ImPact [RRG99] is a system developed by G. Ramalho et al. that emulates a jazz bass player accompanying a musician. It is both real-time and interactive. Its output depends on the *environment*, which consists of the other musicians' performances (pianist, drummer and soloist), the audience and the chord grid. The virtual bass player is the *agent* and it is composed by a listener, a reasoner and an executor.

The *reasoner* is the part of the system that decides what to play. Impact's reasoner is a Case-based Reasoning system: there are some *musical fragments* stored in the system's memory and characterized by some attributes referring to the relation between the system's and the other performer's lines in melodic and rhythmic aspects, and also by the melody and chord grid relation. In front of a new testing instance, the most similar case is retrieved according to the attributes that the instance need. If some attributes are different, the retrieved case is transformed. Finally, the transformed case is stored in the database as a new case. In this way the system learns new cases that could be the target case of future input instances.

The *listener* receives the environment's input (via MIDI) periodically in real-time. The *reasoner* uses that information to retrieve a *musical fragment* among all musical fragments that are stored in its memory and transforms the retrieved fragment in order to adapt it to the actual context. Finally, the executor triggers the notes according to the reasoner's output.

The *listener* of ImPact listens to the evolving context, that is, it not only listens to the soloist but also to the other musicians and the audience. Ramalho was influenced by Hidaka's ideas, so the system defines some events by evaluating some musical properties. An example of these events would be pianist is using dorian mode.

Learning based Jam session system that imitates a player's personality model

In 2003, the research group composed by Goto et al. completed their work by allowing the system to acquire two more models apart from the reaction model [HGAO03]. The *phrase model* was a set of the player's characteristic phrases, as those that students get from their favorite musicians and repeat in all keys in order to take them as part of their musical vocabulary. This set of player's phrases are obtained from the performance's MIDI input, combining machine learning techniques and theoretical knowledge based on the Generative Theory of Tonal Music [LJ83]. On the other hand, the *groove model* is a model based on onset time deviations. The groove model gives information to the system about when does the player improvise playing more slowly or faster than the tempo, with more or less swing, etc. All these models are acquired in the learning mode. In the session mode, the system uses all this information to accompany the performer and imitates the style of the player that has performed in the learning mode.

Cyber-João

Cyber-João (2004) by Ramalho et.al [DST$^+$] is a system that generates guitar rhythmic accompaniment to a given soloist. Because its output depends on the soloist's melody, apart from the given chord grid, we can say it is interactive, or at least reactive. Moreover, it is a Case-based Reasoning (CBR) system. As the authors argue, CBR is a powerful technique as well as a realistic one, as it is the way musicians play.

Their melody fragments are treated as rhythmic patterns, and they are described with parameters of two kinds:

- Environmental parameters: they refer to the context where the pattern is used. These parameters are *harmonic rhythm* (indicates how chords change in a given period of time), and *tempo*.
- Musical parameters: describe fragment's musical properties: *Beginning* describes whether a pattern starts at the down beat; *Fill-in* tells if it is an typical accompaniment that is played when there is no melody. *Usage* determines how frequently is the pattern used, and *density* describes how many musical events are in the pattern.

Chapter 3

Background

In this section we introduce some theoretical concepts that are important in the context of this work.

3.1 Machine Learning

Machine Learning is a scientific discipline which has many applications in the field of music technology as well as in other technology fields as it allows machines to recognize complex patterns and make intelligent decisions based on data. Learning can be supervised or unsupervised. *Supervised learning* learns a function from training data which consists of pairs of input and known desired outputs. The learned function will be useful for predicting the output of any new valid input. *Unsupervised learning* only deals with unlabeled data, so the machine does not know the output of each input. Unsupervised learning problems seek to determine how the data is organized. Another important issue is the type of the data; it can be numerical or nominal (symbolic values, that is, categories). In this work we are using supervised learning methods. Further, the attributes we consider are numeric. In the following we briefly describe the learning methods we are using.

3.1.1 Instance-based Learning

Instance-based Learning [FW05] is a lazy Machine Learning method in which a training data set is stored and a distance function is used to determine the closest training instance to a given new test example. The class of the chosen training instance is assigned to the test example.

There are many distance functions to be used, the most common of them is Euclidean distance:

$$\sqrt{(a_1^1 - a_1^2)^2 + (a_2^1 - a_2^2)^2 + ... + (a_n^1 - a_n^2)^2}$$

Attributes can be either numerical or symbolic. In the symbolic ones, differences between values are 0 if they are the same and 1 if not. For numerical attributes different attribute values may use different scales and this poses a problem; to solve it, when summing up them they have to be normalized.

One problem of Instance-base Learning is that it's time-consuming as each test instance is compared with the whole training set. This can be solved by reducing the training set. There are often redundant examples in a training set, so we can classify each example with the already seen ones and store only the misclassified examples.

Another problem is that the database may be corrupted with noisy examples. Furthermore, the strategy of storing misclassified examples does not work well with noise. There are two possible strategies to deal with this. One is using the k-nearest neighbor (KNN) strategy by which a given number of nearest neighbors (specified by k) are chosen and the class of the test instance is the most repeated one in those k; for numeric data, the final value is the mean of those k-nearest neighbors. Another strategy is to evaluate the performance of the stored exemplars and discard those that are performing under a desired threshold.

Further, there is a drawback in using the Euclidian distance function: it assumes that all attributes are equally important. We can incorporate some class-specific weights for each attribute so that the most important attributes in a class will have higher values.

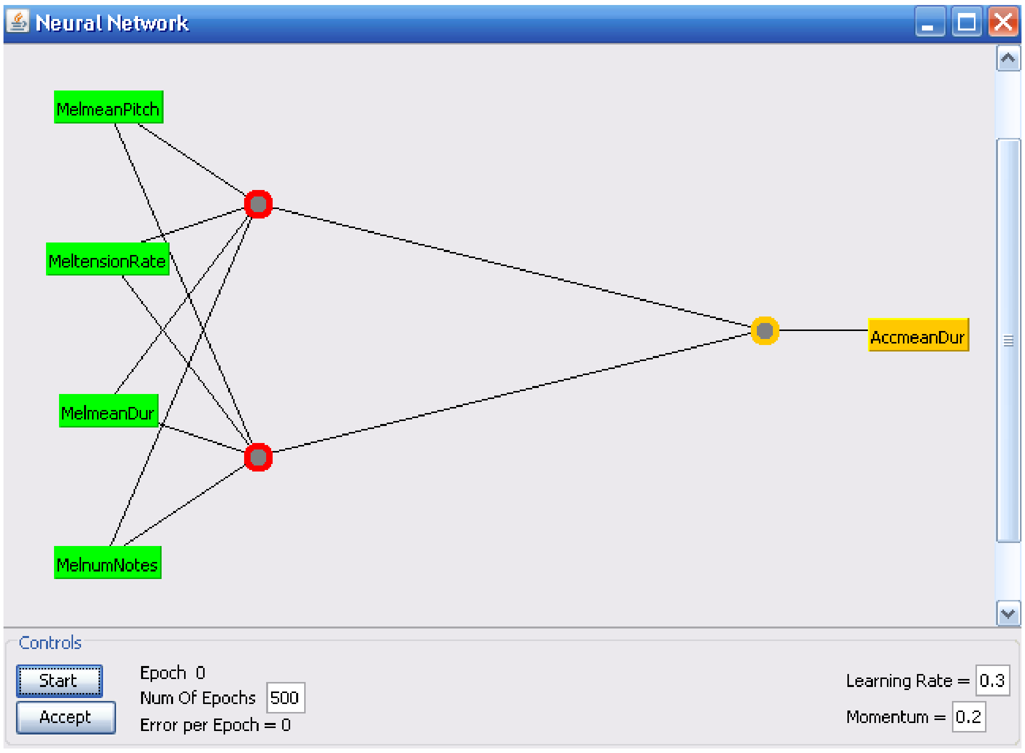
$$\sqrt{w_1^2(x_1 - y_1)^2 + w_2^2(x_2 - y_2)^2 + ... + w_n^2(x_n - y_n)^2}$$

In order to determine the value of each weight, they are set to an equal initial value and are updated after each training instance is classified. If a training instance is x and its most similar exemplar is y, the distance $\|x_i - y_i\|$ is calculated for each attribute i. The weight changes in relation to that difference. If the choice is correct, that is, the chosen exemplar is actually the most similar, the weight of the attribute changes positively, increases, and otherwise decreases.

K* [CT95] is an Instance-based method that uses *entropy* as a distance measure instead of Euclidean or Manhattan distances. Entropy distance is measured as the complexity of transforming one instance into another.

3.1.2 Multilayer Perceptron

An artificial neural network (ANN) is a group of interconnected neurons that process information. Artificial neural networks are mathematical models that try to simulate the structure of biological neural networks. ANN use to be adaptive systems: their structure changes in the learning stage based on the information the flows through the network. Multilayer Perceptron (MLP) is a feedforward artificial neural network (ANN) [MBD+90]. A simple *perceptron* computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights and then possibly putting the output through some nonlinear activation function. MLP is more powerful than a simple perceptron in that it can distinguish data that is not linearly separable. In between the input layer and the output layer are the hidden layers of the network.



Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2

Figure 3.1: Multilayer Perceptron, from WEKA

3.1.3 Sequential Minimal Optimization (SMO) for Support Vector Machines (SVM)

The problem of classifying instances can be solved sometimes by linear models. That is, using linear combination of the attributes to separate classes. Nevertheless, many times a linear classification is not possible. A possible solution is to transform the instance space into a new space where the classes can be classified linearly. A linear model in the new space can represent a nonlinear model in the original space. This linear model in a new dimension is called *hyperplane*. However, the task of finding the best hyperplane can be computationally very complex. Support Vector Machines (SVM) are a set of supervised learning methods used for classification and regression. SVM find the maximum margin hyperplane that separates best the instances into classes. As instances are linearly separable in the new space, there is a polygon that can enclose all of them for each class, the tightest polygon enclosing all instances of each class is called *convex hull*. The maximum margin hyperplane is the hyperplane that is as far as possible from the classes complex hulls as we can see in Figure 3.2. *Support vectors* are those instances that are closer to maximum margin hyperplane. Sequential Minimal Optimization (SMO) is an algorithm for SVM learning using a polynomial kernel [SKBM00].

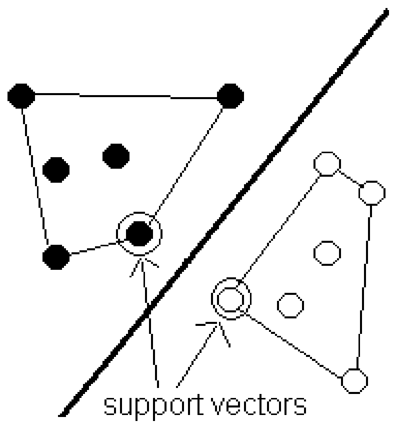


Figure 3.2: A maximum margin hyperplane

3.1.4 M5 Rules

M5 Rules generates a *decision list* for regression problems (problems for predicting continuous variables, rather than discrete categories). For these kind of problems, *regression trees* are a good solution. Regression trees are those trees that have numeric values at the leaves. This value represents the average of all the values of the possible paths to reach the leaf.

A *decision list* is a set of rules that are obtained from a tree, and have to be interpreted in order (check the first rule, if not true, check the second, and so on). The rules are obtained from a *model tree*, which is a regression tree that haves linear equations as leaves. In M5 Rules, in each iteration a model tree is built using M5 algorithm [Qui92] and makes the best leaf into a rule.

3.2 Music theory

3.2.1 Jazz

Jazz is a music genre that was born shortly before the beginning of the 20th century; we could say it is a mixture of various kinds of traditional music; mainly, European music and African music. In the 50s, a style called *Be-Bop* appeared which introduced some new revolutionary ideas referring both to harmony and rhythm. *Jazz Theory*, the contemporary harmonic theory of jazz, is based on BeBop harmonic theory. The *jazz language* refers also to BeBop language, that is, to the specific phrases and scales used in this genre.

From the beginning of jazz, a lot of popular music songs have been written, collected and compiled in a book called *Real Book*. These popular songs are called *jazz standards*.

In a traditional jazz band, let us say a jazz quartet, there are:

- a lead instrument which can be a trumpet or a saxophone (or any instrument capable to play a melody)
- a polyphonic instrument for harmonizing, such as a guitar or a piano
- a bass
- drums

When playing a standard, the melody is usually played at the beginning, then the musicians begin to play their improvised solo successively. The musician that is playing the solo adopts the leader role and the rest of the band accompany him playing in a musically appropriate way according to the soloist's performance. Both the band and the soloist follow a fixed harmonic structure determined by a chord grid, and play according to it. The soloist can improvise with as many iterations of the whole piece as he wants. Each iteration is called *chorus*. The band usually plays again the melody to finish the performance.

3.2.2 Piano comping

Our interest is to focus on the role of a piano while the lead musician is playing the melody or improvising his solo, that is, while he is comping. The pianist has a guide of what he is expected to play, this guide is the chord grid of the standard; in addition to that, he knows that his role is to provide musical support to the soloist. Provided he complies with these two conditions, he can play whatever he wants, there is a high degree of improvisation in his performance. This role is called *piano comping*. The different harmonic lines a piano can draw are called *piano voicings*, that is, voicing means chord assignment including allocation for a given chord name. Also, the piano can play different rhythmic patterns. Both the voicings and the rhythm can be determined by the soloist and by the rest of the band. Furthermore, the piano can interact with the soloist in other ways such as repeating a melodic fragment, filling up spaces the soloist may leave...etc.

Chapter 4

Our system

4.1 Overview

We are building a system that generates polyphonic piano jazz accompaniment to a given melody. This generation is not done in real-time, so the system cannot play together with a human musician. A real-time accompaniment system is a long term goal we would like to achieve, but it is out of the scope of this Master thesis.

According to the taxonomy defined in Section 2.2, our system fits in the category of automatic arrangement systems. In particular, these are the features of the system:

- *MIDI input:* the input is in MIDI format rather than in audio. We decided to focus on MIDI input because it is much easier to process compared to audio input, and this way we can spend more effort on developing the other parts of the system. Nevertheless, most of the components of the system could in the future be adapted to audio input by applying a 'pitch to midi' converter. Thus, the system input is a MIDI melody consisting of a sequence of notes, each note being a structure that contains four values: pitch, onset, duration and velocity.

- *Interaction:* The level of interactivity in our system is null. As we discussed in Section 2.2, non real-time systems are not interactive at all. Anyway, let us assume that in the future we end up building a real-time accompaniment system: the level of interactivity of that system would be exactly the same as the one between a pianist and a soloist. A real-time system that emulates a pianist is interactive because when the system reacts to the human's input, its response can influence human's performance. However, if the system not only reacts but also is able to make decisions from scratch in order to influence the soloist, the level of interactivity is higher because the system assumes at certain points leadership role.

- System's knowledge: as discussed above, this distinction refers to what the system knows about the human's performance. In our system it is very clear. The system has no idea about which notes will the human play but both the accompaniment system and the soloist share the same harmonic structure. This harmonic structure will be determined by the chord grid, from which we will extract the local keys.

- Vertical accompaniment system: the system accompanies the soloist playing along with him and providing a harmonic line rather than trading solos with the human.

- Music knowledge vs. Machine Learning: both of them are taken into account in this system. The melody and the accompaniment are described by extracting from them some features that are musical; for instance, tension rate, that depends on the local key. Also, machine learning methods are used in order to learn the relationship between melody and accompaniment.

In general terms, this is how the system works: there is an initial database of music pieces which are fragmented. Then, given a new melody, the system creates an appropriate accompaniment by *fragmenting* the new melody and *retrieving* for each melody fragment the best accompaniment fragment among all those stored in the database. We train a model using machine learning in order to choose the best melody fragment. Next, retrieved fragments are *transformed* depending on the original and target local keys, Thus, in the end, the accompaniment generated is a list of retrieved fragments. See Figure 4.1

4.2 How the system works

We have already discussed (Section 2.2.5) different approaches for generating accompaniment. The most similar system to ours is Ramalho's intelligent jazz performer [RRG99]. Ramalho's approach is *fragment re-using.* He points that it is a good solution because fragments embody a certain musical knowledge. Further, this *fragment re-using* improves expressiveness as fragments are played by a human musician. And also, it is a natural approach, as it is quite close to the way musicians play: they unconsciously play patterns that they have studied and learned previously. These reasons have convinced us to choose fragment re-using for generating accompaniment.

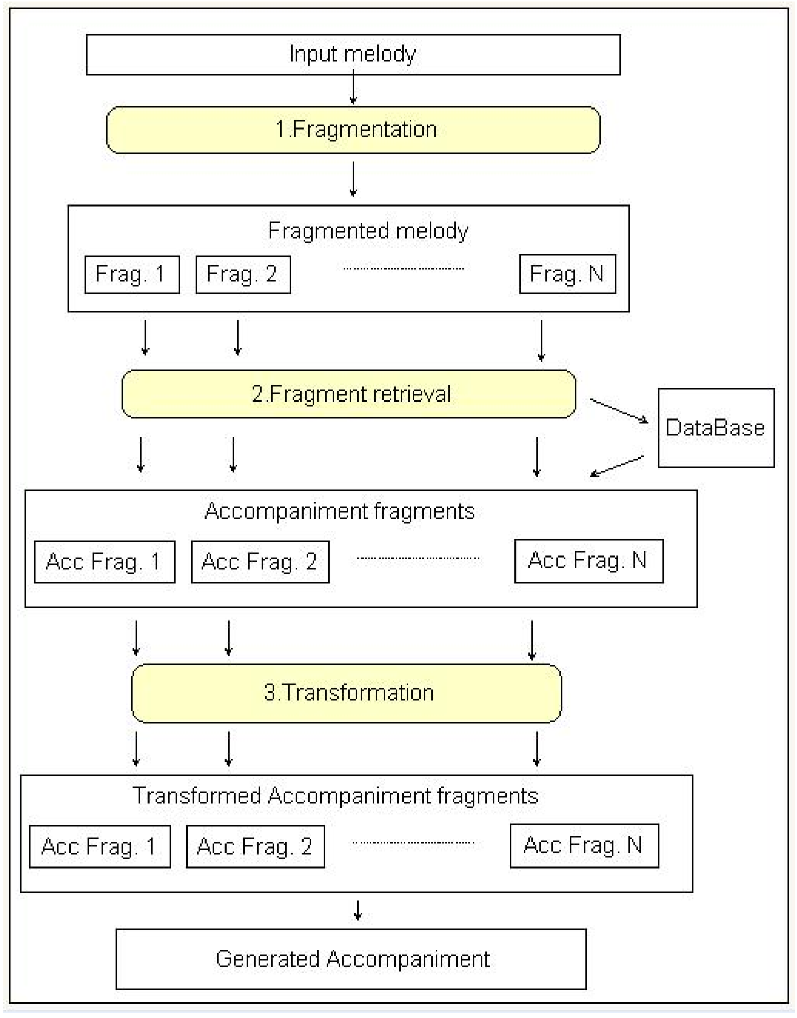


Figure 4.1: System overview

4.2.1 Database

The database consists of melody fragments and accompaniment fragments, both of them in MIDI format. Each melody fragment has its correspondent accompaniment fragment. Also, both of them will have the same chord grid. The total duration summing up all the pieces is about one hour. The recorded jazz standards are: *All of me*, *All the things you are*, *Alone together*, *Days of wine and roses*, *Have you met Miss Jones*, *It could happen to you*, *Just friends*, *Recordame*, *Stella by Starlight*, *There is no greater love* and *Whisper not*. All of them played in mid tempo.

We have two databases for two different fragment description and retrieval approaches (Section 4.2.3). In the first database there is a set of nine standard pieces in which the melody is taken from the *Band in a Box* software whereas the accompaniment is recorded by a pianist with a MIDI piano. The melody is played twice in most of the pieces. On the other hand, in the second database, there are seven standards in which both the melody and the accompaniment are recorded by musicians. The melody is an improvisation recorded by a saxophonist playing a MIDI wind controller, and the accompaniment by a pianist playing a MIDI piano. In this second database pieces are slightly longer, as the solos are between two and three choruses long.

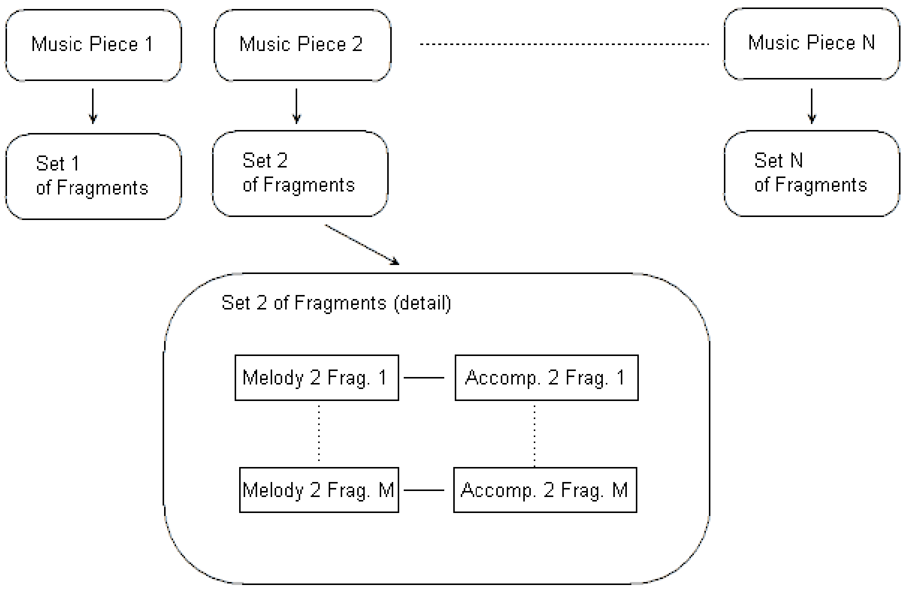


Figure 4.2: Data base diagram

4.2.2 Fragmentation

For fragmenting, we have to decide the size of the fragments and whether all of them will have the same size.

The best decision will be one based on musical criteria. A fragment should represent a musical phrase. Obviously, musical phrases do not have a fixed length. We should use an algorithm for detecting the start and the end of musical phrases. This is a difficult task as musicologists can come to different conclusions when analyzing the same music piece. Actually, melody segmentation is a discussed current research topic.

The problem of having variable-sized fragments is that we will have to compare fragments of different size comparison, so we will have to add or delete tempo beats, and then, to invent the content of that beats when adding, or decide which part to delete otherwise. We could also compare each melody fragment with only those stored fragments that have the same size, but that would reduce the possible target fragments to compare to. Thus, we would need a much larger database.

Therefore we decided that all fragments will have the same length. Although in jazz standards musical phrases use to be longer than one bar, for implementation simplicity each fragment will be one bar long. Further, a fragment will contain notes' information (pitch, onset, duration and velocity) as well as the chord grid. In addition, fragmentation requires to change some MIDI values. Regarding to time, the notes' onset is relative to the start of the piece. When fragmenting we store the onset relative to the fragment's start. Duration is not changed; although the offset of some notes can be out of the time window of the current fragment, that is not really a problem. Velocity is not changed either. Neither is pitch; we will keep absolute pitch as it can give us useful information, although we are aiming at reusing these fragments regardless of their key. Anyway, pitch class relative to the local key can be obtained by a fast operation any time it is needed. We will consider that each fragment has one *local key*; in this way, transformation will give better results as we will see later.

When we are talking about local key, in fact we are really talking about the valid scale to be used over a given chord. Based on musical knowledge, we have considered the following mappings:

- minor 7th chord → dorian mode.

  Example: Cm7 → C D Eb F G A Bb C

- major chord / major 7th chord → major mode.

  Example: Cmaj7 → C D E F G A B C

- 7th chord → mixolydian mode.

  Example: C7 → C D E F G A Bb C

- half diminished chord (minor 7th, flat 5) → locrian mode.

  Example: Cm7b5 → C Db Eb F Gb Ab Bb C

- diminished chord → octatonic scale.

  Example: C dim → C D Eb F Gb Ab A B C

In the standards we are using, there is usually one chord per bar, and often when there are more than one, we can consider that they belong to the same local key. Thus, if a fragment happens to contain more than one chord per bar, we will take just the first chord in order to determine the local key. For instance, let us pretend we have the chords Am7 and D7 in one bar. For Am7 the A dorian scale is: A B C D E F# G A; whereas for D7, the D mixolydian scale is D E F# G A B C D. As we can observe, there are exactly the same notes, so both chords belong to the same local key. This is what usually happens if there is not only one chord per bar. Therefore, we just consider the first one.

4.2.3 Fragment description and retrieval

Fragments stored in the database are described in such a way that they can be compared with other fragments. Fragment description and retrieval can be done in many ways, we have tried two main approaches, which are described as follows.

Approach 1

The database for the first approach contains 9 jazz standards with a melody obtained from the Band in a Box software and an accompaniment recorded with a MIDI piano played by a pianist while listening to the melody.

Fragment description: Melodic Tree Representation. The way we are describing the fragments is by Melodic Tree Representation [RLIn08], so, in fact, we are just changing the format of the melody to another format that will allow us to compare fragments in an easy way.

Melodic Tree Representation is simply representing the melody as a tree in which each leaf is a note with a pitch value. Unlike in the fragments, the pitch values in the leaves are relative to the fragment's local key, because we want to be able to later compare melodies regardless of the key. On the other hand, the internal tree structure implicitly represents time (rhythm): each level of a tree represents the subdivision of the higher level. This subdivision is made in two leaves for binary subdivisions, or three for ternary ones. So in a *two four* time signature (two fourth notes per bar), while the root of the tree would represent the whole bar, in order to represent an eighth note we would need two levels. We can see a representation of the melody in Figure 4.3

In the case of Melodic Tree Representation the problem of comparing two melody descriptions becomes a tree comparison problem. *Edit distance* is a method for comparing two strings in which the distance is based on the number of operations that are required to convert one string into the other. This method can also be used on tree representation. The standard operations that can be done are inserting

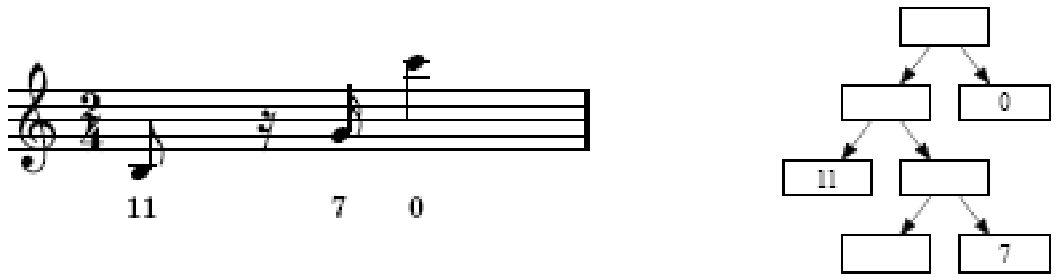


Figure 4.3: Tree representation example

nodes, deleting nodes and transformation, that is, modification of the values of the nodes. Selkow [Sel77] introduced a method based on tree edit distance with one restriction: insertion, deletion and transformation can only be done at the leaf level. Therefore, if an inner node has to be deleted, all nodes belonging to the subtree rooted at it have to be deleted first.

In this approach we have described only melody fragments of the database. Then, given a new melody input, we compare each fragment of the new melody with all the stored melody fragments and, once we have chosen the most similar, we retrieve the related accompaniment fragment (Figure 4.4).

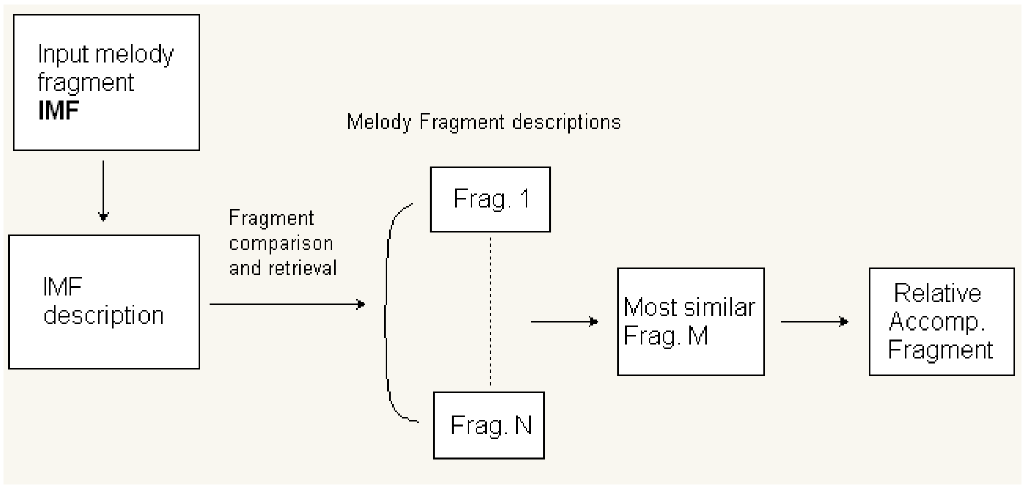

Input melody fragment **IMF**
IMF description
Fragment comparison and retrieval
Melody Fragment descriptions
Frag. 1
Frag. N
Most similar Frag. M
Relative Accomp. Fragment

Figure 4.4: First Fragment retrieval approach

While developing the system, we decided to discard this approach because we have found out that comparing melodies in order to retrieve the related accompaniment is not an optimal solution, because one melody fragment can have many completely different related accompaniment fragments. Furthermore, the decision of which accompaniment fragment to retrieve is based only on melody fragment comparison, without taking into account for what reasons a given melody fragment was accompanied in a particular way. Therefore, we decided to develop a method that focuses on the relation between melody and accompaniment.

Approach 2

The database for this approach contains 7 jazz standards in which both the melody (the solo) and the accompaniment are recorded by humans: a saxophonist playing wind MIDI controller and a pianist playing a MIDI piano. For each standard the performers have played between 2 and 4 choruses.

Fragment description: statistical features. We want to obtain relevant information from the fragments, and also to learn the relationship between melody and accompaniment. That is why we obtain statistical features from the fragments.

Based on musical knowledge, we have defined some parameters in melody and in accompaniment that can influence mutually. The system will be able to learn the relationship between different parameters, which can either refer to pitch, rhythm or dynamics. This relationship will represent some rules that are often applied when musicians perform. Here are some examples of these rules:

- a) If melody plays louder, so does the accompaniment.
- b) If there are tension notes in the melody, the accompaniment plays louder.
- c) When there is more density of notes in a melody there is less density of chords in accompaniment and or viceversa.

By calculating statistical features we are loosing a lot of information and saving just some statistical data, the point is to keep just relevant information. These are the features we have extracted:

For the Melody

Duration mean: this feature tells us if notes are long or short, which can give us a clue of the expressiveness of the fragment, i.e. stacatto or legato.

$$mean(d) = \sum_{n=1}^{N} d_n/N;$$

where *N=number of notes* and *d=duration*

Pitch mean: pitch mean gives us the information of the absolute pitch of the fragment.

$$mean(p) = \sum_{n=1}^{N} p_n/N;$$

where *N=number of notes* and *p=pitch*

Number of notes: which is the same as note density. To get this value, we count the number of notes of each fragment.

Tension rate:

$$tensionRate = \sum_{n=1}^{N}(isTension(n) * d)/N$$

where the output of *isTension* is a boolean [0,1], and d is the duration. In this way, longer notes are more important than shorter notes. Tension notes are those notes that do not belong to the chord of their local key. For example, for C Major, all notes are tension notes except C, E and G.

For the Accompaniment

Duration mean and **Pitch mean** are the same features we calculated for the melody.

Velocity mean:

$$mean(v) = \sum_{v=1}^{N} v_n/N$$

where *N=number of notes* and *v=velocity*

Number of polyphonic onsets: since the accompaniment is polyphonic, in order to obtain the density of notes as we did in the melody, we have to count the onset of the chords; that is, the number of *distinct* note onsets.

Mean notes per onset: the notes per onset is the number of notes that are triggered in each polyphonic onset.

$$mean(notesOnset) = \sum_{n=1}^{N} x_n/Nonset;$$

where n is the current polyphonic onset, N is the number of polyphonic onsets in the fragment, and x is the number of notes in the current onset.

We should have taken into account also velocity mean, but due to some technical problems we lost this information.

This features are similar to Hidaka's primitives [HGM]. Melodic features like *tension rate* are extracted by analyzing the local key.

Both melody and accompaniment fragments are described, and a *model* that defines the relationship between each melody fragment and its respective accompaniment fragment is learned. Hence, thanks to this model the system is able to obtain an accompaniment fragment description from each melody fragment description. So, in front of a new melody, this melody is fragmented and each fragment is described. From this description, which is a statistical description, the *relative accompaniment description* is predicted. Then, the system retrieves the most similar accompaniment fragment among those stored in the database. The similarity measure used is the euclidean distance with a previous normalization of the data (Figure 4.5).

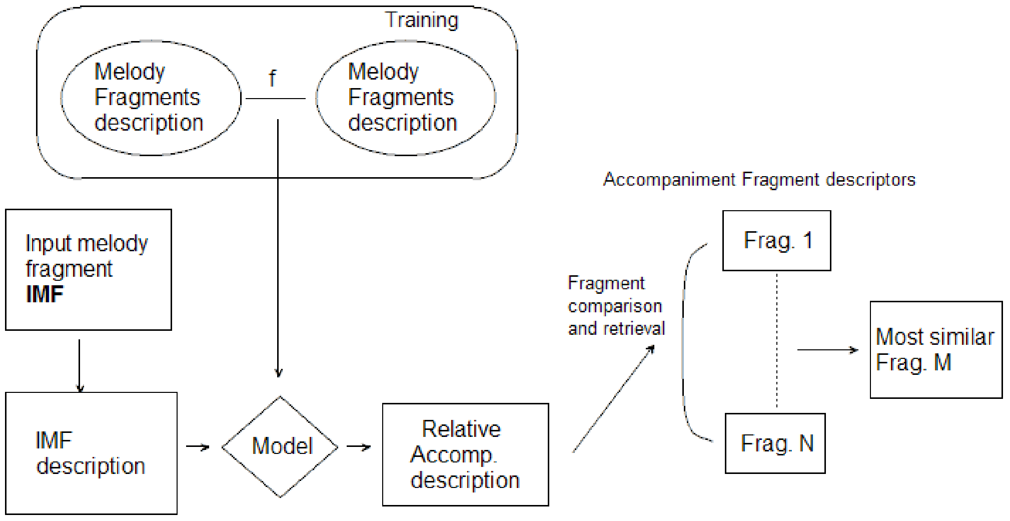

Training
Melody Fragments description
f
Melody Fragments description
Input melody fragment **IMF**
IMF description
Model
Relative Accomp. description
Fragment comparison and retrieval
Accompaniment Fragment descriptors
Frag. 1
Frag. N
Most similar Frag. M

Figure 4.5: Second Fragment retrieval approach

In Section 5.1 we describe how the system learns that model and discuss some results obtained when trying different methods.

4.2.4 Transformation

Once an accompaniment fragment has been retrieved it needs to be transformed in order to fit into the new musical context. This transformation is a pitch transposition that takes into account the harmonic structure. The *local key* (we have introduced this concept in Section 4.2.2) is defined by the chords of the fragments which are further defined by a key and a node. There are twelve keys that correspond to the twelve pitch classes in Western music.

C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#,Ab, A, A#/Bb and B

The chords are seventh chords rather than triads and the available modes in our system are: Major seventh, seventh, minor seventh, half diminished seventh and diminished seventh. Each chord has a scale associated. There are many more chords, but for the moment we can play enough jazz standards with these. Let us see for each mode the notation for key C and the alterations in the degrees of its scale.

- major seventh (Cmaj7): no alterations
- seventh (C7): VII b
- minor seventh (C-7): III b and VII b
- half diminished seventh (C-7b5): III b, V b and VII b

The problem of transformation is traduced to a mapping of notes in the retrieved fragment and its local key (which will be the source key and will be determined by the chord sequence) to the notes in a target key. The process of transforming is done in two steps. First, the recognition of the degrees in the source local key's mode and the mapping of those degrees to the target local key's mode. And second, the transposition from one key to another. Figure 4.6 shows an example of transformation in which a sequence of pitches is transformed from a Cm7 key to a Emaj7 key.

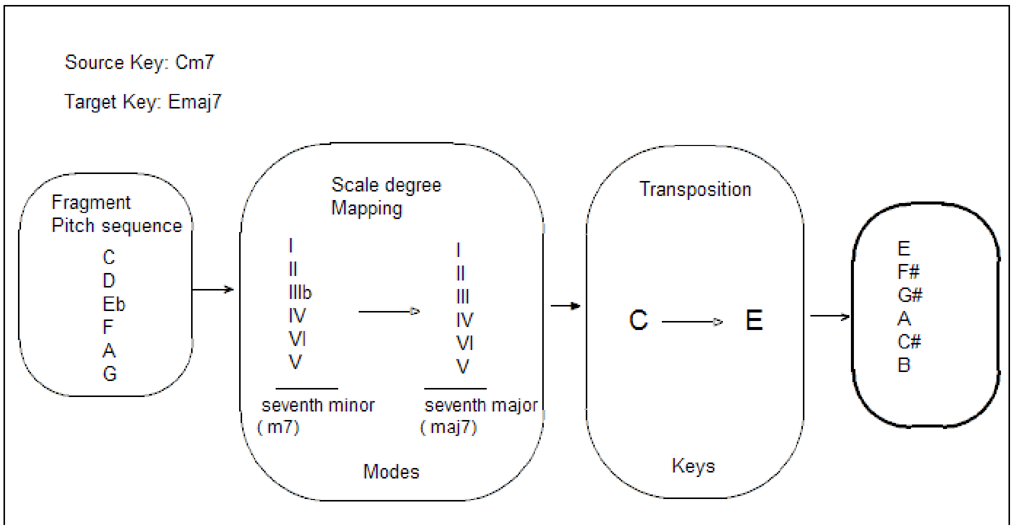


Figure 4.6: Transformation example

There is an issue we did not approach yet: How should we transform chromatic notes that are not in the scale when source and target modes are different? By now we are leaving them as they are. An improvement would be to identify them as a chromatic approximation to a scale note, and map it to the new scale. For instance, Eb in and C Major scale, should be identified as chromatic approximation to E (third degree), and therefore, if the target scale is C minor, the third degree is now Eb and the chromatic approximation becomes D. Anyway, leaving the notes as they are (we just transpose them) does not gives so poor musical results.

4.3 Implementation details

We have the MIDI pieces with the melody and the accompaniment in different tracks. MIDI pieces are transformed to text. This text files will be the input to our Java framework that will fragment the pieces and describe them (either in melodic tree representation or statistical description). To give an overview of the framework these are the main classes:

- AAS: is the main class. It contains an array of Pieces, a ResultPiece and an array of models to learn.
- Piece: it represents a piece that contains two lists of fragments (class FragList), one for the melody and one for the accompaniment.
- FragList: it contains an array of fragments and performs all operations concerning the fragments (add a fragment, compare fragments)
- Fragment: contains the information of a fragment including its description (tree or statistical)
- StatisMus: in this class there are all the statistical values of a fragment.
- Tree representation: for this, we have used a framework from the University of Alicante (UA).

Chapter 5

Results

5.1 Learning the model

In this section we explain how the model in the second approach (Section 4.2.3) to the fragment description and retrieval is learned. As we detailed above, features are extracted for both melody and accompaniment fragments. There are 592 fragments, from which we extract 4 features for the melody and 5 for the accompaniment.

The accompaniment features are:

- Mean Duration
- Mean notes per onset
- Mean pitch
- Mean velocity
- Number of polyphonic onsets.

On the other hand, the melody features are:

- Mean Duration
- Tension rate
- Mean pitch
- Number of notes

The melody features are the known attributes and the accompaniment attributes are the values to predict. For each accompaniment feature there is a model to be found. We have used cross-validation with 10 folds in the training stage. The Machine Learning methods for learning the model are described in Section 3.1. We now go more into detail on the parameters used.

- For the *multilayer perceptron* we have used the 2 hidden layers in all models (which is equal to *number of attributes (4) + number of outputs (1) / 2*), a learning rate of 0.3, which is the amount the weights are updated; and 0.2 of momentum applied to the weights during updating. We have normalized all attributes. And finally we used a 0 seed random number generator for setting the initial weights of the connections between nodes.
- In the *SMO regression* implementation of Support Vectors Machine we have tried to use the method with two different values for the exponent of polynomial kernel: 1 and 2. That is, a linear kernel and a polynomial kernel with exponent 2.
- For the *K-nearest neighbor* method we have used euclidean distance as the distance function to search nearest neighbor and tried with k=1, and k=4, so we took into account 1 and 4 nearest neighbors.
- Finally, for the M5Rules method we have determined a minimum of 4 number of instances to allow at a leaf node.

Correlation coefficients are shown in table 5.1; values are very low. *Mean duration* and *Mean velocity* are the accompaniment attributes that obtain better correlation coefficients. The accompaniment attribute with the worst correlation coefficient is *Mean notes per onset.* However, differences are not big enough to take valid conclusions regarding to attributes. The conclusion we can take is that values are low for all accompaniment attributes. This is because the attributes of our data are not consistent. The recorded data is not good enough neither big enough to extract from it high correlation models. Focusing on the methods, we can see that the lazy methods (KNN and K*) are those that perform worst. The one that gives best correlation coefficients in general is SVM with linear kernel.

5.2 Creating accompaniment evaluation

We have created a model for each accompaniment feature using all melody features as attributes with the Support Vector Machines method (with SMO regression) with linear kernel as it was the one that performed best. When there is a new test melody, for each melody fragment, the system extracts its melody features and uses the models to predict the ideal accompaniment features. Then, it compares the

Table 5.1: Correlation coefficient using all attributes

Dataset	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Mean Duration	0.22	0.25	0.16	0.15	0.12	0.09	0.24
Mean Notes per Onset	0.02	0.03	0.08	0.06	0.01	0.06	-0.04
Mean Pitch	0.08	0.09	0.10	0.07	0.06	0.07	0.01
Mean Velocity	0.24	0.25	0.24	0.07	0.05	0.10	0.24
Number of Polyphonic Onsets	0.15	0.14	0.18	0.03	0.02	0.07	0.14

(1)	Multilayer Perceptron
(2)	SVM with linear kernel
(3)	SVM with in polynomial (exponent=2) kernel
(4)	kNN with k=1
(5)	kNN with k=4
(6)	K*
(7)	M5Rules

predicted features with all those from the accompaniment fragments stored in the database using the euclidean distance and retrieves the most similar.

Before evaluating the results at a perceptual level we have tried the system with a few melodies of the pieces. While retrieving the fragments we have registered which fragment was retrieved each time, and we have observed that there were certain fragments that have been retrieved most of the times whereas there are some fragments that have never been retrieved. This makes us think about the sparsity of the data. Fragment comparison is done by using euclidean distance of the 5 accompaniment features, so in order to visualize the sparsity, we should see how these 5-dimension instances are distributed in space. We cannot visualize 5-dimension data, but we can see some pairs of attributes projected in a 2-dimension plane. For instance we can see Figure 5.1 where X axis represents *meanDur* (duration mean) values and Y axis represents *meanVel* (velocity mean). This figure shows clearly how data is more dense in the area corresponding to high meanDur values and low meanVel values.

We also have listened to some results and we have found out that errors in rhythm are much more noticeable. So we think we should improve the rhythm of the generated accompaniment.

In order to test the results in a perceptual level, we have made a Turing test experiment. We have chosen 5 pieces. For each piece we have both the melody played by a MIDI wind controller, and an accompaniment played by a piano MIDI. Keeping the melody and the chord grid, we have generated a new accompaniment with our system, and also another accompaniment randomly. Whereas our system retrieves the most similar accompaniment fragment stored in the database for each melody fragment, the random approach retrieves any fragment, chosen randomly. Transformation is done in both cases. For each one of the 5 pieces we will have

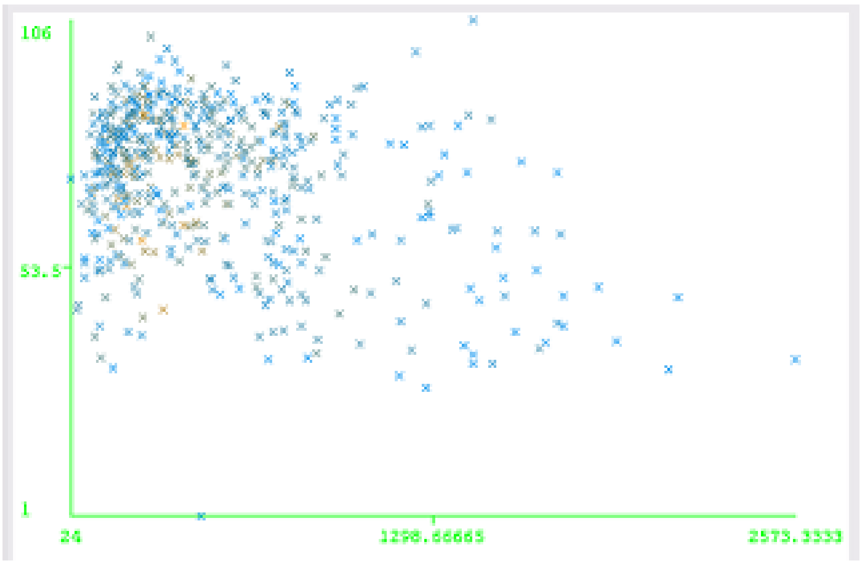


Figure 5.1: meanDur-meanVel

three categories of accompaniment: the *original*, the *generated* one (by our system) and the *random* one. We have chosen excerpts of more or less 10 seconds long for each category. So in the end we have 15 excerpts divided in groups of 3 with the same melody and different accompaniments.

13 subjects with ages between 23 and 36 have listened to them and evaluated how well the accompaniment sounds. There was a brief questionaire asking for general musical skills and for jazz skills which allowed us to divide subjects in 2 main categories: musicians (5 subjects) and non-musicians (8 subjects). Non-musicians are those subjects who listen frequently to music and sometimes to jazz music, some of them play an instrument but they have never studied music seriously. On the other hand, musicians listen to a lot of jazz music and are familiar with jazz language.

Subjects were asked to listen to the excerpts focusing on the piano accompaniment and evaluate them with a mark between 0 and 5 (0 would be ’sounds bad’, and 5 would be ’sounds great’). As it is shown in Figure 5.2, original excerpts got the best marks and the generated excerpts were slightly better than the random ones. Furthermore, we have observed that musicians perceived the difference between generated and random excerpts better than non-musicians, who ranked better random excerpts than the generated ones.

Moreover, we have calculated variance in the marks taking into account all subjects (the mean is represented by the yellow bars in Figure 5.2). Variance tells us how different are the marks between excerpts of the same category. We observed that, the original and the generated excerpts have similar variances, 0.41 and 0.39

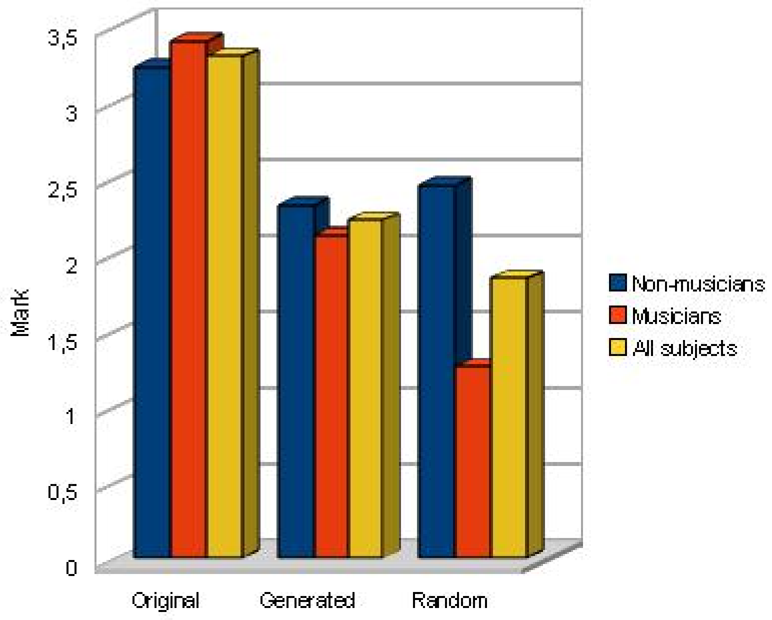


Figure 5.2: Turing Test Results

respectively, whereas random excerpts have a higher variance, 1.03. That means that all generated and original excerpts where evaluated in a similar way whereas random excerpts were more distinctly perceived by the subjects. We could conclude that although our system is far from performing like a human, its performance is better than the random approach's one. This difference is more noticeable for musicians who are used to jazz language. Nevertheless, this is an experiment that has been tested with very few subjects, the conclusions would be more reliable with more subjects.

Some excerpts examples are available at

https://iua-share.upf.es/wikis/smc/Daniel_Martín_Martínez

Chapter 6

Conclusions

We have developed a system that creates an accompaniment for a given melody. This accompaniment is generated by re-using fragments stored in a database. The system learns the relation between melody and accompaniment and uses that knowledge to create the accompaniment for a given melody.

The data used is recorded by musicians playing MIDI controllers. This allows us to get directly MIDI data, but the musician playing the wind MIDI was not used to this instrument, and that has been traduced in a decrease of the quality of the data. In addition to that, the velocity information for the melody has been lost for technical problems, velocity would have given us valuable information for describing the fragments and training the model. For instance, we could have modeled the relation between melody and accompaniment velocity.

Correlation coefficients of the model were very low, the reason of this could be because of the inconsistency of the data; but we should also try to extract more and different features. Then, when we create a new accompaniment by retrieving fragments based on models with a better correlation coefficient, results would be much better.

Sparsity in the accompaniment data is low, this should be fixed recording more data consciously in order to better populate the feature space.

After evaluating the results we can conclude that our system is far from performing as a human, but in ears of jazz musicians our system's results are better than a randomly generated accompaniment. We could say that the reason for that is that our system applies knowledge that has to be learned by getting used to jazz language. However, the test has been evaluated by very few subjects.

6.1 Further work

We have a feeling that we have designed and implemented a good system that could give good results if we improve certain aspects. So next step would be to record more quality data, planning what features to take into account, what relation between melody and accompaniment we want to model, and play consequently.

Further, we should extract more features, mainly those features related to rhythm such as off-beat rate, rhythmic figures, etc. We believe that correctly modeling rhythm aspects would improve a lot results in Turing tests, since rhythm is something very easily perceived. We believe that because of the harmonic complexity of jazz, non-musicians perceive much better bad rhythms than harmonic errors in jazz style.

An important improvement would be to try more than one bar fragmentation; as we already pointed, musical phrases are longer than one bar. Moreover, so far we have just considered one fragment at a time, if we took into account the history of the previous retrieved fragments we would make sure there would be continuity.

When retrieving fragments, we compare them by Euclidian distance being all accompaniment features equally important. We could learn weights of the features by training them; that is, for a given melody fragment, listening to different retrieved fragments, and giving higher marks to those that are more appropriated. This way the system could assign a weight to each accompaniment feature.

Real-time issues. Our long term goal is to come up with a real-time system. But the approach we have chosen has some drawbacks if the system is required to be in real time. First of all, we need to change the way we are doing the fragmentation: we analyze a melody when it is already played and look for its appropriate accompaniment. This is not possible in real-time because the accompaniment has to be played at the same time so there's no time to wait for the melody to end. In real-time, the re-use of fragments would not be based on the representation of the current fragment but on a predicted representation. Predicting can be a difficult task depending on the representation method. Further, the system should learn the relationship between the representation of a fragment and that of the following fragment. Another important issue, like in all real-time systems, is the computational speed of the process and the windowing time, the latter is analog to the fragment size. If we imagine a system that generates response from scratch, the windowing time would be as small as possible as long as there is enough time to extract some relevant parameters to predict next window. In our system, since we re-use fragments, these should be long enough so that they make sense musically by themselves. Otherwise, the method should ensure that the concatenation of fragments does make sense musically.

Bibliography

[Cab08] G. Cabral. *Harmonisation Automatique en temps reel.* PhD thesis, Universite Pierre et Marie Curie, Paris, 2008.

[CT95] J.G. Cleary and L.E. Trigg. K*: An Instance-based Learner Using an Entropic Distance Measure. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 108–114. Citeseer, 1995.

[CZL$^+$01] G. Cabral, I. Zanforlin, R. Lima, H. Santana, and G. Ramalho. Playing along with DAccord Guitar. In *Procs. of the 8 th Brazilian Symposium on Computer Music*, 2001.

[Dav07] M.E.P. Davies. *Towards automatic rhythmic accompaniment.* PhD thesis, University of London, 2007.

[DST$^+$] M. Dahia, H. Santana, E. Trajano, G. Ramalho, C. Sandroni, and G. Cabral. Using patterns to generate rhythmic accompaniment for guitar. *CEP*, 50732:970.

[EMY08] N. Emura, M. Miura, and M. Yanagida. A modular system generating Jazz-style arrangement for a given set of a melody and its chord name sequence. *Acoustical Science and Technology*, 29(1):51–57, 2008.

[FW05] E. Frank and I.H. Witten. *Data Mining: Practical machine learning tools and techniques with Java implementations.* Morgan Kaufmann, 2005.

[Gan91] P. Gannon. Band-in-a-Box. PG Music. *Inc., Hamilton, Ontario*, 1999, 1991.

[HGAO03] M. Hamanaka, M. Goto, H. Asoh, and N. Otsu. A learning-based jam session system that imitates a player's personality model. In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 18, pages 51–58. LAWRENCE ERLBAUM ASSOCIATES LTD, 2003.

[HGM] I. Hidaka, M. Goto, and Y. Muraoka. An automatic jazz accompaniment system reacting to solo. *Proc. of ICMC 1995*, page 167.

[HGO] M. Hamanaka, M. Goto, and N. Otsu. Learning-Based Jam Session System for a Guitar Trio. *system*, 5(6):7.

[Jor05] S. Jorda. *Digital Lutherie: Crafting musical computers for new musics performance and improvisation.* PhD thesis, PhD. dissertation, Universitat Pompeu Fabra, Barcelona, 2005.

[Lew00] G.E. Lewis. Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal*, pages 33–39, 2000.

[LJ83] F. Lerdahl and R. Jackendoff. *A generative theory of tonal music.* MIT press, 1983.

[MBD+90] T. Mitchell, B. Buchanan, G. DeJong, T. Dietterich, P. Rosenbloom, and A. Waibel. Machine learning. *Annual Review of Computer Science*, 4(1):417–433, 1990.

[Pac03] F. Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.

[Qui92] JR Quinlan. Learning with continuous classes. In *In: Proceedings of 5th Australian Joint Conference on Artificial Intelligience*, 1992.

[Rap06] C. Raphael. Aligning music audio with symbolic scores using a hybrid graphical model. *Machine Learning*, 65(2):389–409, 2006.

[RLIn08] D. Rizo, K. Lemström, and J. Iñesta. Tree representation in combined polyphonic music comparison. In *Computer Music Modeling and Retrieval*, 2008.

[Row93] R. Rowe. *Interactive music systems.* MIT Press, 1993.

[RRG99] G.L. Ramalho, P.Y. Rolland, and J.G. Ganascia. An artificially intelligent jazz performer. *Journal of New Music Research*, 28(2):105–129, 1999.

[SCS] D. Schwarz, A. Cont, and N. Schnell. From Boulez to ballads: Training IRCAMs score follower. In *Proceedings of the 2005 International Computer Music Conference*, pages 5–9.

[Sel77] S.M. Selkow. The tree-to-tree editing problem. *Information processing letters*, 6(6):184–186, 1977.

[SKBM00] S.K. Shevade, SS Keerthi, C. Bhattacharyya, and KRK Murthy. Improvements to the SMO algorithm for SVM regression. *IEEE Transactions on Neural Networks*, 11(5):1188–1193, 2000.

[SMB08] I. Simon, D. Morris, and S. Basu. MySong: automatic accompaniment generation for vocal melodies. 2008.

[Tho01] B. Thom. *BoB: An Improvisation Music Companion.* PhD thesis, Carnegie Mellon University, 2001.

[Ver84] B. Vercoe. The synthetic performer in the context of live performance. In *Proceedings of the 1984 International Computer Music Conference*, pages 199–200, 1984.

Acknowledgements I would like to thank Rafael Ramírez for all his support and valuable guidance in this Master Thesis. Also to my friend Amaia Ureta for her contribution to the data recording by playing the piano, and to David Rizo from the University of Alicante for providing me with a framework for melody tree comparison. I am grateful also to my family and friends, some of which have contributed as subjects in the experiments. Finally, I would like to thank Xavier Serra, as director of the MTG, for giving me the opportunity to be part of the MTG as a master student.