pattern recognition + machine learning based on musical information

Patrick Mennen
S422324

Pattern recognition and machine learning based on musical information

Patrick Mennen

HAIT Master Thesis series nr. 11-014

THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ARTS IN COMMUNICATION AND INFORMATION SCIENCES,

MASTER TRACK HUMAN ASPECTS OF INFORMATION TECHNOLOGY,

AT THE FACULTY OF HUMANITIES

OF TILBURG UNIVERSITY

Thesis committee:

Dr. M.M. van Zaanen

Dr. J.J. Paijmans

Tilburg University

Faculty of Humanities

Department of Communication and Information Sciences

Tilburg, The Netherlands

October 2011

Table of contents

1. Introduction

Music is an art-form consisting of expressions of sound and silence through time and consists of a sequence of measures containing chords, notes and rests described by at least duration and in most cases a pitch. The combination of each of these elements determine the characteristics of any given musical score.

Music information retrieval (MIR) aims at retrieving information from musical scores and this information can be used in order to perform a variety of tasks. The most important tasks, finding similarities, music recommendation based on a given query and music-classification, are briefly described in this section, but there are many more uses for music information retrieval (like track separation, instrument recognition and even music generation) .

In 1995 research was done (Ghias, Logan, Chamberlin, & Smith, 1995) which allowed an end-user to query a database with music just by humming a piece of a song. Nowadays popular smartphones like Android-based phones or Apple's iPhone offer a range of free applications (most famously Soundhound and Shazam) that allow an end-user to query an online database by humming, singing or recording a partial track. The success rate may vary per user, but especially for the more popular songs the software achieves a high accuracy and with each request the service improves as the data sent by the user is also stored in the database for future reference. Both applications use similar technology but each application incorporates their own database with audio information. The technology behind these applications comes from research conducted in 2004 by Wang who is actually an employee for Shazam Entertainment Ltd. (Wang, 2006).

MIR research has been conducted in order to counter plagiarism in music. In 2001 a researcher called Yang conducted an experiment which allowed a software application to visualize the resemblance of any given song to other existing musical scores previously stored in a database (Yang, 2001). Newly introduced songs would be compared to this database and a clear identification could be given on whether or not the song was an original new piece or (loosely) based on another song.

Another commonly used practice is using MIR to recommend new music to listeners of music of a specific band or genre (Tzanetakis, Ermolinskyi, & Cook, 2003). It is possible to offer a list of related artists to an end user. There are many more features on which new recommendations can based and returned to the visitor: emotion, mood, year of production and so on (Feng, Zhuang, & Pan, 2003; Kanters, 2009; Li & Ogihara, 2003); The website last.fm ("About Last.fm," 2011) offers users to download and install a plugin (or as they call it the Scrobbler) for their favorite media player, which in turn tracks whatever music the user is playing on his/her computer or mobile device and uploads this information to their website. The uploaded data is then compared to data other users have submitted and based on these data the website can return similar artists or genres. In their turn

users can like (or love in terms of last.fm) the suggestions made which over time specifies whether or not the system associates a certain band or genre with an individual song. Research has been conducted on how the system practically works and which accuracy it attains (Celma & Lamere, 2008).

The last and for this thesis most relevant use for MIR is classification based on genre, country of heritage, artist or composer. Different musicians or composers often either consciously or subconsciously leave a reoccurring pattern of notes, pitch changes, duration or tempo changes in their different scores. This pattern can be seen as the artist's signature and based on this idea we are trying to implement a machine-learning algorithm by using specific computer software in order to detect and extract these signatures from individual musical scores. These extracted patterns (or signatures) can then be used to train a computer to detect these patterns in a different library of musical information allowing it to classify an unknown piece to a specific artist or author. Classification tasks are not strictly limited to an artist or composers, but patterns can be found for different properties of a given song (e.g. demographic information, genre, musical period of composition).

Earlier research (Dewi, 2011; Ogihara & Li, 2008; van Zaanen & Gaustad, 2010, 2011) showed computers trained using a software toolkit can successfully categorize musical scores based on the pitch and duration of the individual notes in the performance. This research allowed to categorize the music based on composer, but also on demographic properties like the pieces original region or a musical period in which said piece was composed. This technique can be particularly useful when one tries to categorize a large library of music files. Instead of doing the categorization process by hand, the system can find patterns in the music that are typical for a specific genre allowing it to automatically assign this genre to the specific score.

Musical scores can be stored on a computer in various formats ranging from a digital representation of a given performance, to an actual representation of the score. Some of the more well-known file-formats are MP3 (Motion pictures expert group layer 3), WAV (Waveform Audio) and MIDI (Musical Interface Digital Interface). These file-formats differ drastically and each of these individual types have some distinguished features and but also have some limitations.
This thesis will go into detail regarding the technical aspects of two file formats and will extend existing research in order to find out whether or not a different file format will yield the same results when used in an experimental setting.

We will compare the well-known and established MIDI-format, to a lesser-known format, namely **kern humdrum, which is specifically designed for research purposes and will try to establish whether or not a computer can extract similar information from a different file format using techniques that already provided excellent results with the ** kern humdrum format.

1.1 Problem statement

Previous research has already established the possibility of using pattern-recognition and machine learning to perform classification tasks on a library of musical information in the **kern humdrum format. The **kern humdrum format was specifically designed for research purposes.

This research is trying to conclude whether or not the possibility exists that these very same techniques can successfully be used on a different file format, which is not originally intended for research purposes but for recording a performance of a musical piece and what modifications to the original setup, if necessary, are required in order to attain these results.

1.2 Hypotheses

We will try and find the answer to the problem statement by testing the following hypotheses.

H0: Converting a library of **kern humdrum files into a library of MIDI-files and running the same experiments on both the original and the converted data should result in a similar outcome.

Even though the two file formats are completely different and serve different purposes, which will be illustrated in later chapters of this thesis, the expectation is that conversion from the **kern humdrum format to the MIDI format has no significant effect or influence on the results generated by the software toolkit used in the experiments and the outcome of the experiment will yield the same results.

H1: While the previous hypothesis predicts that we can get similar information out of both experiments, we also predict that some of the parameters used in the original experimental setup might need adjustment order to gain these results.

The expectation is that converting the source **kern humdrum files to the target MIDI files will not generate a one-to-one representation of the original file format. Therefore we predict that some of the parameters for the feature-extraction program may need some modification in order to circumvent erroneous or biased data generated from slightly different source files.

H2: Quantization of the MIDI timings is necessary because MIDI is known to handle the exact timing of musical events differently compared to **kern humdrum which is a precise one to one representation of a musical score.

Especially with files that are <u>not</u> generated from a **kern humdrum file, we expect that some of the MIDI timings cause errors . In order to prevent these errors to cause biased information we may need to apply some quantization which in essence evens out the value generated by the conversion to the nearest duration.

H3: Given a dataset that solely consists of unconverted MIDI-files the expectation is that the machine-learning algorithm will perform classification of a large categorized dataset significantly better than baseline classification algorithm.

We expect that if a conversion from a **kern humdrum source to a MIDI equivalent causes no real complications in terms of classification accuracy, we can also apply the same techniques to a dataset which consists solely of MIDI files which have no **kern humdrum counterpart. This would indicate that even though the file types are different, applying the same techniques still generates sufficient results.

2. Methodology

In order to test the given hypotheses some background information has to be gathered about the internal workings of both the **kern humdrum and MIDI format and to establish the key differences between the file formats and to find the strengths and weaknesses of each of these formats. This information will be gathered by a literature study which is described in chapter 3.

By utilizing custom-tailored software on two identical datasets of musical information (one set in the **kern humdrum format and the other in the MIDI format) we can verify whether or not training computers to classify music using the different file format is possible. It should be noted that the MIDI files are automatically generated from the ** kern humdrum file and therefore the copy should prove to be identical. As classification on the **kern humdrum files has shown to yield good results (van Zaanen & Gaustad, 2010) we chose to utilize the same **kern humdrum datasets that were used in that research. These datasets are available at the Kernscores [1] -database which conveniently offers the datasets in different file formats like MIDI. The software used in this thesis differs from the software used in the original research as support for multiple file-formats was added by using the Music21 library.

This research consists of a set of three individual experiments. The first experiment compares the results to the original research in order to validate whether or not the new data-extraction module is working properly. The second experiment is used to determine and verify whether or not **kern humdrum and MIDI-files attain similar results and the third and final experiment utilizes a comprehensive dataset which only contains MIDI files and which was previously used in a classification competition.

[1] http://kern.ccarh.org/

3. Literature study

MIDI is an industrial standard established by multiple organizations, the standard and its rules are defined in official standardization documents which are available on the Internet ("The Complete MIDI 1.0 Detailed Specification," 2001). Most of the documents are available free of charge, but some extended documents are available to paying customers only. However these documents tend to be very detailed as the standard is used by manufacturers to implement the MIDI technology in their hard- or software and for the purposes of this thesis these standardization documents are far too detailed. The information found in this chapter is a very brief summary of the relevant information found in the standard-documentation.

As **kern humdrum is a lesser-known format and as it is mainly used for research, not nearly as much information about the format itself and its inner workings is available. The official Humdrum-toolkit provides an online book which explains the purposes, syntax and possibilities of the ** kern humdrum format. As **kern humdrum is solely aimed at researchers, the information available is scarce when compared to the availability of information with regard to the MIDI-standard. The next two sections take an in depth look at the two file formats.

3.1 MIDI

In the early 1980s, Sequential Circuits Inc. (SCI) made a proposal for a Universal Synthesizer Interface. The idea behind this interface was that hardware from different manufacturers could utilize this interface in order to create a standard protocol for synthesizers. The idea was quickly supported and adapted by other manufacturers like Oberheim, Yamaha, E-mu, Roland and Korg.

The first adaptation of this standard primarily supported note triggering, which basically means that it merely specified that a particular note should be played at a given moment during the song. In 1982 several Japanese companies created a counter-proposal to extend the features of the protocol. These features were similar to the Roland parallel DCB (Digital Control Bus/Digital Connection Bus) interface. DCB was a proprietary, meaning owned by a single company in this case Roland and closed source, data interchange interface which allowed sequencers to communicate with programs. At this point the Status and data structure was introduced, which allowed more control than the standard note-triggering protocol. Eventually both proposals, the Universal Synthesizer Interface and the DCB-standard, were combined into the MIDI specification we know today by SCI. In 1987 SCI was acquired by Yamaha.

The standard was released to the public-domain, meaning nobody has ownership over the MIDI standard. This is generally seen as a huge part of the success of the MIDI-interface as nobody licenses or policies the MIDI-standard making it an open and co-operative standard. This ensured that other developers adapted MIDI in their hardware and to this day MIDI is used by sequencers.

MIDI has also been used in many other cases as for example in videogames. One of these videogames is Rock band 3 which allows the player to play along with some of the bigger rock bands in the history of rock and roll (e.g. Deep Purple, The doors and David Bowie). The game has the option to play with a "professional" controller which in essence is a real Fender guitar which uses a MIDI interface to communicate with the game console. On the harder difficulties, the videogame requires the player to play the chords as they are played in the real song which teaches the player to play a real guitar whilst also playing a videogame. (Harmonix, 2010)

Cellular phones used the MIDI standard for their ringtones before the production companies adapted more modern file types like MP3 into a new iteration of their product design.

The MIDI-file format does not store a digital representation of a given musical score, but consists of various commands that are specified in the MIDI-standard. The combination of these commands determine how any given device, from a sequencer to a computer's soundcard, should interpret the file and which "instruments" to use. Using this command set has some advantages and some disadvantages; a typical MIDI-file has a very small file size compared to digitized representations but playback on different devices or soundcards can have noticeably different results as the music instruments need to be emulated by the hardware and the quality of this hardware has direct influence on the quality of the sound output.

MIDI was originally intended to be a protocol between various hardware and thus instructions are formatted in packets that are sent over a serial-interface which allows data to be transferred to hardware that has such a serial interface. These serial bytes are sent every 320 microseconds and have a distinct structure consisting of one start bit, eight data-bits and finally a single stop bit. These commands or MIDI messages can be divided in two categories: the Channel and System messages. Channel messages contain a four bit channel number which addresses the message specifically to one of the sixteen available channels, whereas system messages can be divided into three subcategories namely System Common, System Real Time and System Exclusive. The rate at which commands can be sent is also a limitation, because some notes often need to be triggered simultaneously and the amount of notes that can be triggered at once is limited by the serial package size.

3.2 **kern humdrum

The **kern humdrum format was specifically designed to aid music researchers. It is part of the Humdrum toolkit[2] which is freely available on the internet. The official documentation (Sapp, 2009) states that the **kern humdrum format is intended to provide researchers with a file format that supports a broad variety of tools with regards to data exploration in musical information. The Kern-format was specifically constructed for the toolset and is not meant to transfer the information to other hardware or the computer's soundcard as is the intention of MIDI, rather it describes music in a way that researchers can perform various tests on the data (Huron, 2002). However the toolset comes with some programs that can convert the **kern humdrum format into other formats like MIDI or musicXML.

The **kern humdrum toolkit consists of a set of over 70 different tools that can be used to perform tests on musical information written in the Kern format. The tools available in the toolset can all be started from a command line and no programming skills are required in order to use the tools. Here is a brief overview of some of the available commands in the **kern humdrum toolkit:

- Proof: verifies the syntax of the source **kern humdrum file and it can be used to fix syntactic mistakes in a source score.
- Census: provides extensive information about a given score, it describes the source **kern humdrum file listing some of its features like the number of lines, the number of unique interpretations, the number of comments etc. Basically it provides the end-user with a detailed report of the file in question.
- Assemble: The assemble command allows two or more structurally similar **kern humdrum files to be aligned together, making it possible to merge two or more **kern humdrum files into a new file containing multiple voices.
- Pitch: translates **kern humdrum pitch-related representations into the American standard pitch notation.

The **kern humdrum-format is an ASCII-representation of a musical score with some added meta-information and control-codes. ASCII stands for the American Standard Code for Information Interchange and is a character-encoding scheme which defines 95 visible characters and 33 invisible control characters that can be used to represent textual information.

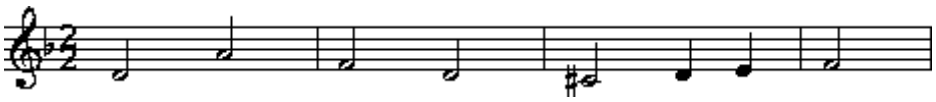
The documentation states that the **kern humdrum format can be used for exploratory research, but strongly advises to use a clear problem statement. Some of the problem statements the official documentation gives as an example;

[2] http://www.musiccog.ohio-state.edu/Humdrum/

- What are the most common fret-board patterns in guitar riffs by Jimi Hendrix?
- How do chord voicings in barbershop quartets differ from chord voicings in other repertoires?
- Which of the Brandenburg Concertos contain the B-A-C-H motif?
- In what harmonic contexts does Händel double the leading-tone?

All of these problems can be analyzed by the various tools that are available in the toolset but the toolset is limited to the **kern humdrum syntax and if there is a need to extract information from a musical score which is not available in this format it needs to be converted manually or by using special software on for example a MIDI equivalent of the score. The **kern humdrum format is an ASCII-representation of a musical score, meaning that it is a human-readable format and it can be opened and modified in any text editor as opposed to MIDI. The inner workings of a **kern humdrum file can best be explained by using an example. We are going to describe the conversion from a measure of notes into a **kern humdrum equivalent. We are converting the short excerpt from Bach's "die Kunst der fuge" displayed in figure 1 into a small **kern humdrum file.

Figure 1: *Musical representation of Bach's composition "Die Kunst der Fuge"*



The **kern humdrum representation for this staff looks like the listing in figure 2. Note that the line numbers are not part of the actual **kern humdrum file but are added in order to describe the inner working of the format in the next paragraph;

Figure 2: *Musical representation of Bach's "Die Kunst der Fuge" in **kern humdrum.*

```
1.  **kern
2.  *clefG2
3.  *k[b-]
4.  *M2/2
5.  =-
6.  2d/
7.  2a/
8.  =
9.  !! This is a comment right between measures
10. 2f/
11. 2d/
12. =
13. 2c#/
14. 4d/
15. 4e/
16. =
17. 2f/
18. 2r
19. *-
```

A **kern humdrum file has a distinct beginning and end-tag as depicted on line 1 and line 19 respectively, everything between these lines should be interpreted as musical-information (except for comments, indicated by !!, as depicted on line 9). Lines 2, 3 and 4 set the clef, the key-signature, which in this case is b-flat and the meter (2/2) respectively. The measures start at line 5 and are indicated by the equal sign (=). The minus sign indicates the first measure is invisible depicting there are no notes prior to this specific measure. Lines 6 and 7 represent the first two notes on the measure and line 8 indicates the next measure. The notes (depicted on lines 6, 7, 10, 11, 13, 14, 15 and 17) are described using a relative duration with regards to the measure. The note 2d/ on line 6 indicates that the note d is half a measure long (1: whole note, 2: half note, 4: quarter note, 8: eighth note etc.) and its stem is pointed upwards which is indicated by the forward slash in the notes definition. The pitch of the note is described by one or more characters which describe the properties of the note please bear in mind that the syntax is case-sensitive meaning that C is not equal to c. The note C can be described in many ways;

c	Middle C (i.e. C4)
cc	C an octave higher than middle C (C5)
C	C an octave lower than middle C (C3)
CC	C two octaves lower than middle C (C2)
c#	C middle sharp (C#4)
cn	C natural, middle c (C4)

Line 18 does not describe a note, but rather a rest. Rests are similar to notes but do not have pitch information as rests are not played. In this case the rest is used to fill up the remainder of the measure. Because the rest is the very last element in the musical score it is hidden in the graphical output. Multiple voices can co-exist divided by a tab and sheet music can be described in its entirety.

Given a syntactically-correct **kern humdrum file each of the different tools included in the toolset can be used to extract information from the file which in turn can be used for research purposes.

3.3 Other file formats

As the previous two sections have already stated, the **kern humdrum and the midi file format were invented for different purposes. The comparison of MIDI with **kern humdrum checks whether or not the techniques used in the original research can be used on a significantly different file format, which happens to have some similarities to the original format. MIDI does not represent sheet music in the same way as **kern humdrum does. Instead of describing notes, the way **kern humdrum does, it triggers specific notes (and even different instruments). Both file formats describe notes which are available in the sheet music in the form of instructions to the machine or hardware it corresponds with. Even though the inner working of MIDI is significantly different, it still allows us to convert the triggered notes into sheet music.

More modern file-formats like MP3 and Flac (Free Lossless Audio Codec) are far more complex than both MIDI and Humdrum, as they store the musical information as compressed digitized sound. Digitized sound is an actual recording of a musical piece and does not describe the meaning of each individual note in the file itself, therefore it is more difficult to extract information from digitized sound and different techniques are required in order to extract information from this type of file.

As sheet music is not represented in digitized file types (there is no command structure as is the case with both MIDI and **kern humdrum) we cannot use the system we plan on using during the course of this thesis on these newer file types, but perhaps techniques similar to the ones used by Wang (Wang, 2006) which measure a score's density can be used to classify songs.

4. Procedure

In order to test the hypotheses defined in chapter 3, there is a need for three individual experiments that are conducted by using custom-written software which is an extension of the software-package used and described in earlier research by van Zaanen (2010). The software was used in multiple theses and experiments which in turn served completely different purposes (Beks, 2010; Dewi, 2011; van Zaanen & Gaustad, 2010). This chapter describes how the software works, but we will first take a look at the three experiments that we will run in order to answer the hypotheses we previously described in chapter three.

The first experiment conducted is nearly identical to van Zaanen's research using the same corpus but using the newly implemented software. This experiment could be seen as the final rehearsal for the new software as the results of this experiment should prove that the new library is doing its job properly and we should basically find the same results as van Zaanen did in his original research.

The second experiment is actually identical to the first experiment, the only difference is the file-format of the corpus. The aim of this experiment is finding out whether or not the same machine-learning techniques can be used on an identical set of data in a different file-format while still receiving correct output. Basically this experiments tests whether or not the parser is able to read and extract information from the MIDI-files directly.

The first two experiments directly complement each other as they are used to check whether or not the software is capable of handling both MIDI and **kern humdrum files correctly and these results can be used to verify the integrity of both the software and the file-types. These experiments basically serve as a final preparation for the third and last experiment which is going to be performed on a third dataset which is only available in the MIDI-format. The initial two experiments are required because the third experiment's corpus is not available in the **kern humdrum format so we cannot test the corresponding **kern humdrum dataset.

For our third and final experiment, we have chosen a comprehensive dataset that purely consists of MIDI files. This dataset was a part of a competition which was held in 2005 at the annual Music Information Retrieval Evaluation eXchange (West, 2011) and consists of a large amount of classes (36) as opposed to the experiments that were used in the original research which only implemented a maximum of four classes. The expectation is that even though there is a difference in the amount of classes, the software will still provide a significant increase of classificationaccuracy when compared to the majority baseline calculation.

The third experiment differs from the second MIDI experiment, because the MIDI files used have not been converted from **kern humdrum to MIDI. However the same dataset was used in the 2005 MIREX competition where other classification systems competed to gain the highest

classification accuracy and it is possible to compare the results of the classification systems that competed in the competition to the accuracy attained in the course of our experiments.

4.1 Data preparation

Preparing the data-files for processing proved to be a challenge even though the Kernscores-database [3] offered multiple versions of each individual score it had no option to download the collection in its entirety. The database is of a considerable size and contains many individual files. Crawling the website manually by using an automated software tool (Wget) proved to be both inefficient and timely mainly because the website's administrator had set up a load balancer which prevented the crawler from downloading too many files in a short time span. This balancer was set up to redirect an overflow of requests to a simple text-file which briefly explained that if power user access was required one could contact the system's administrator.

After personal contact with the system's administrator, Craig Sapp, access to a recursive download was provided which allowed a download for the Essen folksong dataset and the Composers dataset which will be described in the next paragraph. This download only contained the **kern humdrum versions of the files and in order to obtain the MIDI versions manual conversion from the source **kern humdrum files to their MIDI equivalent was required. Sapp advised using the **kern humdrum toolkit's hum2mid (Sapp, 2005) program which is available in the extras package of the toolkit and also provided a shell script that automatically could convert the library into MIDI using the hum2mid application.

The two obtained datasets are the same sets that were used in the research by van Zaanen et al. (2010). This was done intentionally because it gives the option to compare the results generated by each version of the software toolkit to each other. These datasets are the Essen dataset which contains folk songs from both Western and Asian countries. This dataset is a monophonic dataset, meaning there is only a single voice in the song. In the experiments this dataset is indicated as the Countries dataset. The second dataset contains songs composed by famous composers Bach, Corelli, Haydn and Mozart. These songs consist of multiple voices and thus are polyphonic. This dataset is indicated as the Composers dataset.

The dataset used for our third and final experiment was used in a contest which tested different classification systems at MIREX 2005. The Bodhidharma software written in 2004 by McKay achieved the highest classification accuracy in the contest (McKay & Fujinaga, 2005). More information about the internal workings of his software can be found in McKay's thesis (McKay, 2004). The dataset used in the competitions solely contained MIDI files so there was no need to convert the data. This dataset is known in this thesis as the Bodhidharma dataset.

[3] http://kern.ccarh.org/

After converting the **kern humdrum files into MIDI using the hum2mid program I verified the data generated by the software by playing the MIDI files in a media player. The conversion had resulted in a library of broken MIDI files. The problem was to blame on a bug in the at the time current version of the hum2mid application which was not ready for the newer 64-bits architecture newer computers use nowadays. After contact with the toolkit's developers this issue was corrected and the current version of the **kern humdrum toolkit converts **kern humdrum files to their MIDI counterpart successfully on older as well as newer computers.

The software used to conduct the three experiments defined in chapter four makes use of a third-party library called Music21 (Cuthbhert & Ariza, 2010; "music21: a toolkit for computer-aided musicology," 2011) to interpret the musical information contained in the datasets. This interpreter is very strict when it comes to syntax and the slightest syntactic error causes the program to exit as opposed to the hum2mid-tool which is more lenient when it comes to syntactic mistakes.

Testing the generated MIDI dataset with Music21's interpretation software revealed that a large quantity of the files generated by the hum2mid program could not be read by Music21's interpretation software. Music21's interpretation software is an absolute necessity for the three experiments and losing a large amount of files in our datasets would be problematic so we needed to convert the data differently and without using the hum2mid application in order to achieve maximum compatibility with the Music21 parser. Browsing through Music21's API documentation ("Music 21 Documentation," 2011) revealed that Music21 has the option to store its output into various standard audio representation formats like **kern humdrum and MIDI and thus it created the opportunity to create a custom parser based on Music21's own interpretation software and thus ensuring that the files generated would be compatible with our experimental software.

After writing a custom parser in Python (Sanner, 1999), parser.py in the tools directory of the experimental toolset, which tried converting the original **kern files into their MIDI equivalent. This parser is a strict convertor and any syntactic errors in the source **kern humdrum file cause the file to be excluded from both the **kern humdrum and the MIDI dataset.

The amount of files converted successfully determines the size of the dataset for our experiments. A complete overview for the converted data for both the MIDI and **kern humdrum dataset can be found in table 1. The scores in both the **kern humdrum and the MIDI datasets are identical.

Table 1**:** *Description of the Datasets for the First Two Experiments*

Dataset	Amount of files	Converted successfully	Percentage
Countries	**8454**	**8062**	**95.36%**
Asia	2241	**2169**	96.79%
Europe	6212	**5893**	94.86%
Composers	**787**	**469**	**59.59%**
Bach	246	**220**	89.43%
Corelli	247	**47**	19.03%
Haydn	212	**201**	94.81%
Mozart	82	**1**	1.22%
Totals	**9241**	**8531**	**92.32%**

The numbers in table 1 indicate that the parser has some trouble with parsing a percentage of the original source files. It should be noted that the musical scores composed by Wolfgang Amadeus Mozart in the composers dataset gives the new parsing software significant trouble as only one of the files is converted successfully . The expectation is that this will have a positive result on the accuracy the classification software achieves, as it has to only classify three classes instead of four.

The Bodhidharma dataset contains 988 MIDI files which are divided into 38 individual classes, after testing whether or not the files could be read with Music21's converter software it turns out that 728 (73.68 %) of the files were correctly parsed and interpreted. The musical scores were originally evenly divided over each of the classes, putting 26 files in each of the classes however due to the loss of 26.32 percent of the files the categories are not evenly represented which may cause some difficulties whilst performing the baseline calculation in the experimental phase. Most classes still have more than 70 percent of their original contents intact in only four occasions there is a significant loss of information for a specific class. These losses occur in the following datasets: Adult Contemporary (53.85%), Bluegrass (46.15%), Contemporary country (50%) and most notably the Celtic class (30.77%). None of the classes could be converted without the loss of one or more files. The two classes with the best conversion rate were Country blues and Swing with a 92 percent conversion rate. A complete overview of all of the classes in the Bodhidharma set and the successful conversion rate for each of the individual classes can be found in table 2.

Table 2**:** *Classes and Successful Conversion Rate for the Bodhidharma Dataset*

Class	Amount of files	Converted successfully	Percentage
Adult contemporary	26	**14**	53,85%
Alternative Rock	26	**20**	76,92%
Baroque	26	**23**	88,46%
Bebop	26	**21**	80,77%
Bluegrass	26	**12**	46,15%
Blues rock	26	**18**	69,23%
Bossa Nova	26	**21**	80,77%
Celtic	26	**8**	30,77%
Chicago blues	26	**18**	69,23%
Classical	26	**22**	84,62%
Contemporary country	26	**13**	50,00%
Cool	26	**22**	84,62%
Country blues	26	**24**	92,31%
Dance pop	26	**21**	80,77%
Flamenco	26	**22**	84,62%
Funk	26	**19**	73,08%
Hardcore rap	26	**21**	80,77%
Hard rock	26	**20**	76,92%
Jazz soul	26	**22**	84,62%
Medieval	26	**23**	88,46%
Metal	26	**16**	61,54%
Modern classical	26	**20**	76,92%
Pop rap	26	**21**	80,77%
Psychedelic	26	**18**	69,23%
Punk	26	**18**	69,23%
Ragtime	26	**22**	84,62%
Reggae	26	**16**	61,54%
Renaissance	26	**21**	80,77%
Rock and roll	26	**19**	73,08%
Romantic	26	**20**	76,92%
Salsa	26	**15**	57,69%
Smooth jazz	26	**19**	73,08%
Soul	26	**18**	69,23%
Soul blues	26	**19**	73,08%
Swing	26	**24**	92,31%
Tango	26	**23**	88,46%
Techno	26	**19**	73,08%
Traditional country	26	**16**	61,54%
Totals	988	**728**	73,68%

The Bodhidharma dataset was also used in Boudewijn Beks' thesis (Beks, 2010) but he converted the MIDI data to musicXML and then to **kern humdrum before using it for his experiments. The complexity of the original MIDI files also had an impact on his conversion accuracy. The conversion rate for his experiments was 46,53%. Music21, the library used for the new experiments and more thoroughly described in chapter 4.2, internally converts files from a dataset to a Python object but the conversion rate of the Music21 interpreter is higher than the results attained by the mid2hum and mid2xml tools from the **kern humdrum toolkit.

Tests with the Music21 MIDI interpreter revealed a bug which made the interpreter ignore the very last note on any given score. In order to circumvent this bug an additional empty rest was appended to the MIDI-score during conversion from **kern humdrum to MIDI. This additional rest was not appended to the files in de Bodhidharma dataset, as there is no equivalent of this dataset in the **kern humdrum format.

4.2 Software toolkit

The software used in this thesis differs from the software used in the original research by van Zaanen and Gaustad (2010). The original software was only intended to work with the **kern humdrum format and for this thesis the toolkit was expanded to allow support different file formats. This new implementation uses a free and open-source library developed by the Massachusetts Institute of Technology, Music21 [4] to perform the analysis on the extracted data. The software is written with compatibility in mind, meaning that previous experiments should still be able to run properly.

Music21 is a software toolkit with similarities to the Humdrum toolkit, but Music21 is not bound to the specific **kern humdrum syntax as it supports a collection of different formats like for example MusicXML and also MIDI. The toolkit also allows us to create graphical representations of the interpreted data, we can either measure the pitch levels and even regenerate the measures that are available in the source data. Music21 is a highly active project and is receiving constant updates. It can be downloaded from its official subversion repository.

One of the big differences between Music21 and the **kern humdrum toolkit is that basic programming skills are required in order to use the tools that come with the toolkit. Music21 merely provides the developer with an API (Applications Programmer interface) which can be used to extend his/her own programs with the features the Music21 toolkit offers. It is not possible to run experiments from the command line as is the case with the **kern humdrum toolkit. Music21 is written in Python and by writing Python scripts one can use the library in order to gain information about a musical score.

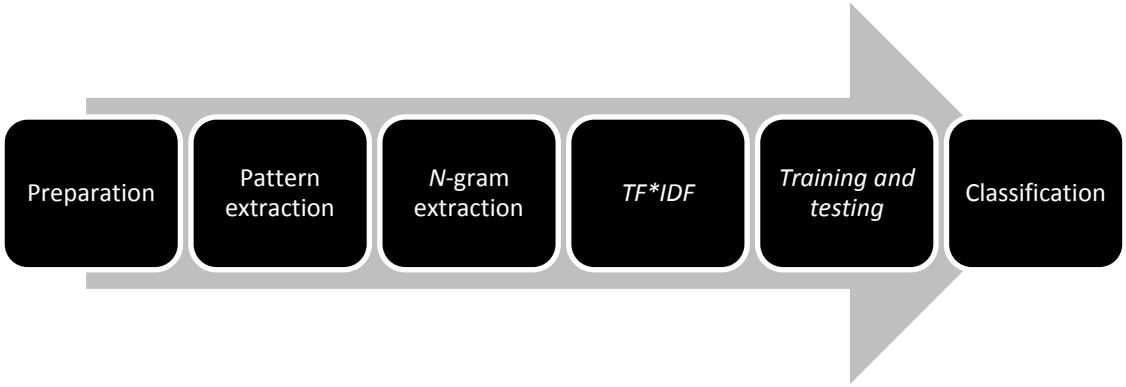
[4] http://mit.edu/music21/

As the original software was specifically written for the **kern humdrum format it invoked methods and commands that were solely applicable for the ASCII-representation that is used by the **kern humdrum files. Music21 uses an entirely different method of extracting information from the various file types. It splits a single score into different accessible objects which can be read and modified from within the Python program. Luckily a large part of the existing codebase used in the original research could be reused without the need for a rewrite.

The parsing program which extracts the various features from the musical scores and prepares them for machine-learning purposes is the only actual part of the software that required a complete rewrite. Even though the internal working of the new interpretation class changed drastically, the new parser's output was aimed to be as close to the output generated by the original version's output as possible. This allows the results generated by the new parser to be compatible with the other tools that were inside the original version's toolkit. This circumvented the need to rewrite the whole toolkit to add support for multiple file formats.

The software application performs a variety of operations on the dataset while conducting the experiment. These operations can be categorized in six stages which are displayed in figure 3.

Figure 3: *Schematic overview of the various tasks the toolkit performs.*



4.2.1 Preparation

The first step the software undertakes is randomly dividing each of the individual songs in the dataset in so called folds. The songs are evenly distributed amongst the folds regardless of their original class. The folds are used for 10-fold cross validation and are used in the training and testing step of the application and described in more detail in section 4.2.5. After the division is complete the software proceeds into the next preparatory step namely the baseline calculation. Calculating the baseline assigns the most common class to each file in the corpus. This process results in the highest accuracy attainable without using any information from the contents of the files. This accuracy can in turn be compared against the results of the new parsing software. Ideally the new parser's accuracy should significantly surpass the accuracy attained by the baseline calculation. As a general rule of

thumb we can assume that the amount of individual classes has a direct influence on the height of the accuracy of the baseline calculation.

4.2.2 Pattern extraction

During the next step the application prepares the files in the different folds for the machine-learning and classification tools. This preparation extracts various features from the source file, generating an output which can be used for machine-learning. Table 4 shows which features were implemented in the Music21 version of the parsing software:

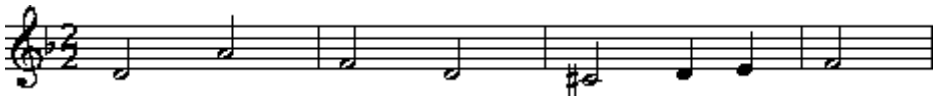
Table 3: *The Individual Encodings Available in the New Parser.*

Encoding	Abs./Rel.	Description	Polyphonic
Pitch absolute	Absolute	Numeric representation of the pitch-space of an individual note or chord (e.g. C4=0, C#4=1 etc.)	No
Duration absolute	Absolute	Numeric representation of the tempo which applies to an individual note, chord or rest taking into account modifiers like dots	No
Multiple pitch absolute	Absolute	Same as pitch absolute but applied to each voice	Yes
Multiple duration absolute	Absolute	Same as duration absolute but applied to each voice	Yes
Pitch contour	Relative	Indicates whether or not the current note's pitch is either higher (+1), lower (-1) or equal (0) to the previous note or chord	No
Duration contour	Relative	Indicates whether or not the duration of the current note or rest is longer (+1), shorter (-1) or equal (0) to the previous duration	No
Duration relative division	Relative	Divides the duration from the current element with the duration of the previous element	No
Duration relative subtraction	Relative	Same as duration relative division only subtracts the previous note pitch space from the current note.	No
Pitch modulo	Absolute	Folds the notes in the first voice to the fourth octave and returns the numeric value (i.e. C1 is transformed to C4 which returns 0)	No
Multiple Pitch Modulo	Absolute	Same as Pitch modulo only applied to all voices	Yes

The harmonics functions, which were available in the original parser were omitted as they were not used in the original research and thus they are not needed for the experiments described in this thesis. If these functions are required for future research they need to be developed at that time. These functions were primarily used by Boudewijn Beks in his 2010 thesis and were used as an extension to the already existing functions that classify polyphonic musical scores.

The system stores the extracted encodings in individual files and these encodings are represented as numerical data. As an example let us recall our earlier excerpt from Bach's "Die kunst der Fuge" and manually extract its pattern for both pitch- and duration absolutes and pitch relative and duration relative features experiments. The system converts the MIDI or **kern humdrum syntax into an object which contains a representation of the elements in a musical score (measures, notes, rests etc.)

Figure 3: *Converting a Musical Score into a Pattern*



Note	d	a	F	d	c#	d	e	f	REST*
Duration	Half	Half	Half	Half	Half	Quarter	Quarter	Half	Half
Converted (Absolute)	2:0.5	9:0.5	5:0.5	2:0.5	1:0.5	2:0.25	4:0.25	5:0.5	0.5
Converted (Relative)		7:0.0	-4:0.0	-3:0.0	-1:0.0	1:-0.25	2:0.0	1:0.25	0.0

** Rests have no pitch as they produce no sound and therefore for rests only the duration is calculated*

Converted (Absolute): In this case the conversion software looks at each element and stores its absolute value as a numeric value. The note D is converted to a numeric value which responds to the number of semi tones with respect to middle C (C4 equals 0) in the case of the note D the numeric value would be two whereas D# would be converted to the number three etc.

In some cases only a partial feature can be extracted because one of the features might not apply to the given element. In the example the last element (a rest) only the duration (0.5) can be calculated because a rest does not have a pitch and therefore this attribute is omitted in the calculation.

Converted (Relative): In this case the conversion software looks at each individual element and compares this with the previous element in the song. Therefore the first note in the song cannot generate any output as there is no predecessor to compare to which is represented in the example with a shaded cell, this first element is not omitted, but used for calculating the value of the second

element. The differences between the previous note and the current note are measured and saved as the value for this feature (e.g. from note D to note A is a difference of seven semitones and the difference between a half note and a half note is zero).

The three experiments implement yet another combination of features which is not illustrated in the example in figure 3 due to its simplicity. Pitch- and duration contour simply looks at the previous element in the song and determines whether or not the pitch or duration is equal (0), higher (1) or lower (-1) than the previous element.

Each of the three experiments are set up to generate three feature files for each individual song in the dataset. These encodings consist of a group of two encodings: 1.) pitch absolute and duration absolute, 2.) pitch relative and duration relative division and 3.) pitch contour and duration contour.

4.2.3 Generate feature vectors

In the next step, the software generates the so-called feature vectors for each of the three experiments. By using different pattern sizes in the form of *n*-grams, we can verify whether or not the size of a pattern has influence on the results of the classification and if so which pattern length is optimal for correct classification.

The toolkit is set up to extract patterns with a sequential size of one to seven consecutive elements in a given song. These elements represent different aspects of the song. In the absolute experiments, the elements describe individual notes, rests etc. whereas in the relative and contour experiments the elements describe relative note information (e.g. the difference between two notes).

These probabilities are computed by looking at a sequence of words or entities located before the entity in question (Jurafsky & Martin, 2009). An *n*-gram is a sequence of words/entities with the length of *n*. An *n*-gram model is a type of probabilistic model used to predict the next entity in a given sequence of words or entities. Jurafsky and Martin (2009) describe an *n*-gram model as a statistical language model that assigns probabilities to any given sequence of words. *N*-gram models are commonly used in statistical natural language processing but are also used for other purposes (e.g. genetic sequence analysis).

In a linguistic context *n*-grams are utilized for a variety of tasks varying from word-boundary prediction to handwriting- and speech recognition. As *n*-grams can be used on a sequence of entities, we can also apply the probabilistic principle to the data we extracted from the three datasets. The numeric representation of the various features (absolute, relative and contour) is used as a sequence. When the *n*-grams have been extracted from the data files, the software assigns weights to the patterns using information retrieval techniques.

4.2.4 TF * IDF

Term Frequency-Inverse Document Frequency (TF*IDF) is used to assign a weight to the extracted patterns (Akiko, 2003). This weight indicates how common a unique pattern is amongst the scores in one of the experiment's datasets. TF*IDF is a common information retrieval technique usually implemented in order to find the importance of a certain word (or sentence and in this case an occurrence of numerical information) in a specific query on a range of documents. TF*IDF is commonly used in the vector space model together with cosine similarity to determine whether or not two individual documents are similar or not (Lochbaum & Streeter, 1989). Variations of this technique are often used in search-engines in order to rank the most relevant documents given a user's query.

Van Zaanen and Gaustad's research (van Zaanen & Gaustad, 2010) has indicated that these very mechanics can also be applied on the data which has been extracted from the music files in our dataset. The individual patterns are weighted and linked to an individual class. In the case of van Zaanen and Gaustad's research the musical data was monophonic (or handled as such by the parsing software), Boudewijn Beks has expanded this research in his thesis and implemented an algorithm that handled more complex polyphonic patterns. (Beks, 2010).

The complete TF-IDF formula is divided into two segments, each describing a variable used in the formula. The first variable, the term frequency value (TF), is the number of times a term appears in any given document divided by the total words that appear in the document. This division acts as a normalization and makes sure there is no bias which favors longer documents over shorter ones. This results in the following formula:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

In this formula the $n_{i,j}$ depicts the number of times the term t_i appears in document d_j. The denominator in turn represents the length of document d_j which is measured by counting the total number of terms in document d_j.

The TF variable is commonly used in natural language processing and if a specific term occurs commonly in a single document, this document is considered more relevant than documents that not or less frequently contain this term.

The second variable, the IDF-value which was originally proposed in 1972 by Karen Spärck Jones (Garcia, 2008; Robertson, 2004) is calculated by searching the entire collection of documents. The measure is obtained by dividing the total number of documents by the number of document in which the word/pattern occurs and then taking the logarithm of the that outcome. The formula for the IDF value for term *t* looks like this;

$$\mathrm{idf}(t) = \log \frac{|D|}{|\{d : t \in d\}|}$$

The formula for the IDF variable might look more complex than it actually is. In the formula $|D|$ stands for the cardinality (or the total number of documents in the corpus), which in this thesis is the total number of classes as a document is a collection of musical scores of a certain class.

The second part of the formula $|\{d : t \in d\}|$ stands for the number of documents in which the term *t* appears. In natural language processing the formula is adjusted slightly in order to prevent an erroneous calculation if the term is not found in the corpus itself (which would result in a division by zero) but this is not required for our setup as each of the patterns is available in at least one of the documents (musical scores).

Both variables are combined in the last step of the formula and the result is a weight for a single term in relation with a given document;

$$\text{tf-idf}(t, d) = \mathrm{tf}(t, d) \times \mathrm{idf}(t)$$

A term in a document will receive a high TF-IDF score if it occurs often in the current document but it is not well represented amongst the rest of the corpus. If a pattern gains a high TF-IDF score, this means that this is important for a specific class or genre. Terms that are commonly represented in many of the documents in the corpus will achieve a low TF-IDF score which in turn results a low importance.

The TF-IDF weightings are used in the classification process in order to determine which class belongs to the document in the corpus. If a single pattern occurs in all of the classes it achieves a TF*IDF score of zero (0) and because this individual pattern occurs in each of the classes it is impossible to use this pattern for classification and thus as a result it is discarded.

4.2.5 Training, testing and classification

When the pattern collection process is complete, the system is ready to be trained on the files that are available in each of the 10 folds and the classification process can commence.

In paragraph 4.2.1 we briefly mentioned the *K*-fold cross validation technique (Mullin & Sukthankar, 2000). This means that the machine learning system is trained on the extracted patterns available in the training data in a methodical way. In *K*-fold cross validation a pool of labeled data is partitioned into *k* equally sized subsets (in our case 10) and each of these subsets is used as testing data whilst the other *K*-1 subsets (9) are used as training data. Afterwards the average accuracy across all *K*-tests is computed which results in the average accuracy for the specific test. The main advantage of this method over the alternative, randomly selecting train and testing data, is that the way the data is divided is not critical. Every song in the corpus gets to be in the testing set exactly

once and is available in the training data *K*-1 times. This means that there is no bias towards a specific file or genre because all patterns are used exactly the same amount of times.

The major downside for using *K*-fold cross validation is that the training algorithm has to run its course *k* times which makes it a more timely process. As an example, the experiments we ran during the course of this thesis ran on a computer which had 11 CPU-units available for calculation and just the **kern humdrum experiments took over a week and a half to complete.

The classification process stores the information for each of the tests in a Tab-separated file which is compatible with most statistical software packages and allows us to extract the results from each of the individual experiments .

5. Results

After each of the individual experiments were executed, the results were stored in a TSF (Tab-separated file) and were imported in the open source statistical package R[5] (Team, 2008).

The results were loaded into a linear model and by using statistical methods like the analysis of variance (ANOVA), Tukey HSD, mean and standard deviation on each of the datasets and for each of the experiments we can check whether or not the various experiments have run their course correctly and measure the accuracy for the individual experiments as such.

5.1 Experiment #1 –Testing the conversion software

The initial intention was to compare the software's output to the research conducted by van Zaanen et al. in 2010 and therefore the datasets used for this experiment (Countries and Composers) were kept identical to the datasets used in the original experiment. However due to the fact that Music21's parser cannot handle each of the individual **kern humdrum files correctly, which in turn causes the datasets to differ significantly, it is impossible to utilize a paired T-test to compare the accuracy of both experiments to each other. The inconsistencies between the two datasets are illustrated in table 1.

Even though the majority of the difference in the dataset is caused by Music21's parsing software there are some other factors that influence the difference between the results generated by the original and the current implementation of the software.

In personal communication van Zaanen indicated that some "hacks" were implemented in the original version of his parsing software which in turn allowed it to interpret syntactically incorrect source-files which would be rendered useless otherwise. These hacks are not usable in the Music21 version of our software.

Another key difference between the two software implementations is that repeats in songs are expanded in van Zaanen's research which means that a repeat is actually visible in the system's output whereas Music21 does not support expansion of repeats as of yet and thus only outputs the notes of an individual repeat once resulting in smaller output files and potentially different patterns.

In conclusion, it is impossible to compare the results generated by both systems to each other, but fortunately we can still determine the new implementations' overall performance which is depicted in table 4.

[5] R or the R project is available at: http://www.r-project.org/ and is also available in the package manager of most popular Linux distributions.

Table 4: *Quantitative Results for the **Kern Humdrum Experiments Mean Accuracy and Standard Deviation.*

Class	Composers						Countries					
	Absolute		Contour		Relative		Absolute		Contour		Relative	
Baseline	46.93%	(5.35)	46.93%	(5.35)	46.93%	(5.35)	73.10%	(0.92)	73.10%	(0.92)	73.10%	(0.92)
1	84.39%	(6.67)	49.00%	(1.98)	60.07%	(19.57)	77.95%	(0.63)	73.33%	(0.58)	80.77%	(1.82)
2	94.94%	(3.88)	38.13%	(15.99)	79.38%	(18.66)	92.14%	(6.82)	55.14%	(24.17)	97.38%	(2.52)
3	91.66%	(3.34)	73.26%	(9.63)	89.28%	(13.92)	99.89%	(0.15)	64.57%	(18.97)	99.93%	(0.09)
4	90.98%	(3.78)	82.29%	(8.23)	93.03%	(11.07)	99.99%	(0.01)	86.91%	(8.02)	100%	(0.00)
5	92.62%	(3.27)	85.20%	(6.86)	94.03%	(8.26)	100%	(0.00)	98.12%	(1.44)	100%	(0.00)
6	93.16%	(3.23)	88.20%	(5.67)	94.03%	(4.12)	99.99%	(0.01)	99.76%	(0.29)	100%	(0.01)
7	93.20%	(5.21)	89.73%	(6.19)	92.94%	(3.99)	99.99%	(0.01)	99.92%	(0.11)	99.98%	(0.02)
1-2	94.48%	(3.60)	37.11%	(14.37)	80.92%	(14.82)	92.14%	(6.92)	55.14%	(24.17)	97.38%	(2.52)
1-3	94.57%	(3.79)	58.50%	(12.34)	85.69%	(15.64)	99.89%	(0.02)	78.03%	(2.29)	99.98%	(0.02)
1-4	94.76%	(3.39)	67.94%	(11.40)	87.81%	(15.04)	99.99%	(0.01)	90.17%	(4.18)	99.99%	(0.01)
1-5	94.83%	(3.33)	71.23%	(11.08)	89.28%	(14.48)	99.99%	(0.01)	98.59%	(0.84)	99.99%	(0.01)
1-6	95.13%	(3.23)	74.69%	(10.41)	90.17%	(14.08)	99.99%	(0.01)	99.90%	(0.12)	99.99%	(0.01)
1-7	95.39%	(3.21)	77.07%	(10.93)	90.97%	(13.80)	99.99%	(0.01)	99.99%	(0.01)	99.99%	(0.01)

Training the system on the data available in the corpus has a positive effect on the accuracy in comparison to the baseline calculation for each of the features. This is reflected in the results of the linear model which states that there is a significantly positive effect on the accuracy for all of the experiments with regards to the baseline (the p value for each of the experiments in comparison to the baseline is 0, indicating a significant difference).

For the **kern humdrum experiments table 4 shows an increase in accuracy in comparison to the baseline for every *n*-gram size peaking at around a pattern-size of five to seven. This illustrates that for this part of the experiment the highest classification accuracy is attained by a relatively high *n*-gram size. In other words a longer sequence of elements in a song will more accurately classify a given score than a shorter sequence.

There is a relationship between the attained accuracy and the pattern size or the combination of pattern sizes. Contrary to the results achieved by van Zaanen in his 2010 experiment, in which the pattern sizes peaked at around an *n*-gram size of 3 or 4, the accuracy achieved by the experiments in this thesis increase whenever the *n*-gram size increases. This means that longer patterns or a combination of a short and longer patterns increase the accuracy of the classification process.

On the other hand this means that smaller patterns have less influence on whether or not a given score can be classified correctly.

As stated before, there is no way to compare these results with van Zaanen's original research as the datasets used in this thesis differ from the datasets used by van Zaanen. Even though the difference in the dataset exists, we can still compare the highest attained accuracy for both systems. These accuracies were generated by the absolute experiments as they are the only ones that are available in van Zaanen's 2010 publication. The comparison is depicted in table 5.

Table 5: *Accuracy For each of the Experiments by both Software Implementations **Kern Humdrum.*

Dataset	Thesis' version	Van Zaanen's version
Countries	100% (0.00)	95.54% (1.72)
Composers	95.39% (3.21)	82.07% (4.25)

The figures in table 5 clearly indicate that the new parser is gaining a marginal increase in accuracy in comparison to the original software. However the difference in accuracy does not confirm that the new parser is working better and this is because the datasets are significantly different.

The increased accuracy attained by the new parsing software can also be explained by comparing the files available in each of the datasets which were used for each of the experiments. Especially with both the Corelli and the Mozart dataset there is a significant loss of files which almost rules the class out in its entirety in the conducted experiment. Realistically speaking, the software only has to take two classes into consideration instead of the four classes that were used in the original research.

This gives the system used in this thesis a huge advantage and this directly explains the increase in accuracy for the composers dataset. The countries dataset also shares this advantage with 710 files which were excluded from the classification process and thus the increase in accuracy is also explainable here.

5.2 Experiment #2 – Applying the conversion software to MIDI

The same procedure was repeated for experiment two. This experiment was conducted at the same time in order to assure that the files are nearly identical for both experiments. The only major difference is that the file type has changed from **kern humdrum to MIDI. The MIDI dataset lacks three files because Music21's parser cannot extract the features from these files. A t-test revealed that the majority baseline did not significantly differ and thus the absence of these three files has no significant influence on the experimental results. The results for the MIDI experiment are found in table 6.

Table 6: *Quantitative Results for the MIDI Experiments Mean Accuracy and Standard Deviation.*

Class	Composers						Countries					
	Absolute		Contour		Relative		Absolute		Contour		Relative	
Baseline	46.57%	(5.52)	46.57%	(5.52)	46.57%	(5.52)	73.11%	(0.92)	73.11%	(0.92)	73.11%	(0.92)
1	83.50%	(6.64)	48.24%	(1.45)	44.23%	(24.95)	78.43%	(0.69)	73.34%	(0.58)	81.78%	(1.89)
2	93.00%	(3.60)	28.23%	(16.47)	66.86%	(16.51)	96.02%	(1.80)	73.55%	(0.61)	98.47%	(0.70)
3	91.13%	(3.53)	62.27%	(9.72)	68.13%	(9.16)	99.97%	(0.03)	76.98%	(2.12)	99.98%	(0.02)
4	93.22%	(3.55)	72.81%	(10.47)	76.89%	(8.23)	100%	(0.00)	89.68%	(4.34)	100%	(0.00)
5	93.22%	(3.01)	80.93%	(7.17)	83.29%	(6.22)	100%	(0.00)	98.47%	(0.87)	100%	(0.00)
6	95.42%	(2.38)	88.19%	(4.78)	91.43%	(5.80)	100%	(0.00)	99.87%	(0.11)	100%	(0.00)
7	97.86%	(1.96)	91.44%	(3.96)	93.87%	(4.77)	100%	(0.00)	100%	(0.00)	100%	(0.00)
1-2	92.73%	(3.62)	27.80%	(15.15)	64.18%	(10.63)	96.02%	(1.80)	73.55%	(0.61)	98.47%	(0.70)
1-3	92.61%	(3.34)	50.64%	(13.29)	69.59%	(10.90)	99.97%	(0.03)	76.98%	(2.12)	99.98%	(0.02)
1-4	93.00%	(3.12)	58.82%	(10.79)	70.70%	(9.76)	100%	(0.00)	89.68%	(4.34)	100%	(0.00)
1-5	93.63%	(3.14)	62.71%	(10.37)	72.78%	(9.63)	100%	(0.00)	98.47%	(0.87)	100%	(0.00)
1-6	94.36%	(2.87)	64.86%	(9.59)	74.45%	(10.13)	100%	(0.00)	99.87%	(0.11)	100%	(0.00)
1-7	94.58%	(3.00)	67.40%	(8.51)	75.25%	(10.24)	100%	(0.00)	100%	(0.00)	100%	(0.00)

As is the case with the first experiment, the experiments conducted have a higher accuracy in comparison to the majority baseline in almost all cases which in turn indicates that the machine learning algorithm is outperforming the baseline calculation significantly for each of the three

features. Once again the p-value for all of the experiments equals to 0 indicating a significant difference.

The linear model allows us to compare the results generated by the **kern humdrum and the MIDI experiment to each other in order to see whether or not there are any significant differences between the accuracy attained by the **kern humdrum and the MIDI experiments.

The accuracies generated by the composers experiments for each file type differ significantly from each other ($P < 0.05$) whereas the accuracies generated by the countries datasets are not significantly different to each other ($P = 0.26$).

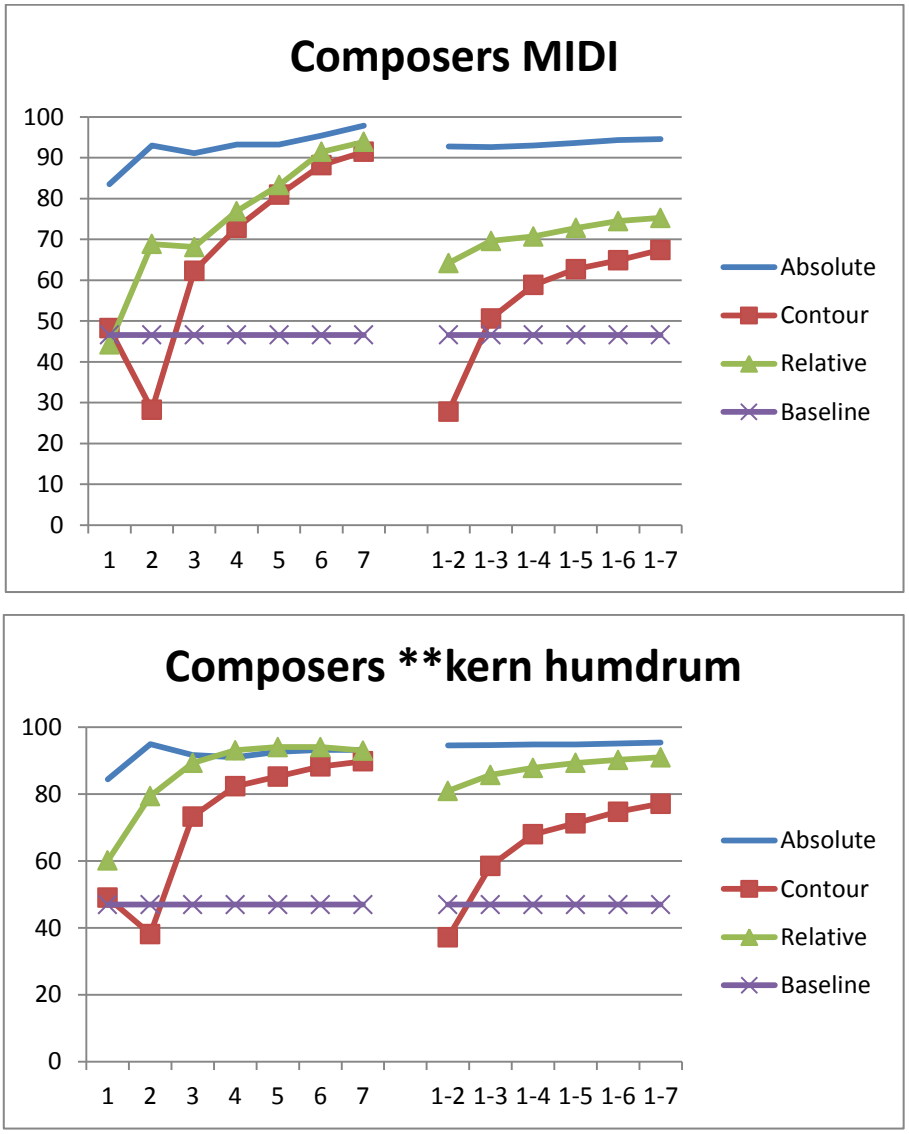
For the Countries experiments this indicates that there is a nearly one-to-one conversion from **kern humdrum to MIDI and that Music21's parsing software is treating the MIDI files the same as their **kern humdrum counterpart. The data files used in the Countries experiment solely consist of scores that just use one single voice and this makes the conversion from **kern humdrum to MIDI relatively easy and this is illustrated by the significant accuracies attained for each of the feature experiments.

The results for the Composer datasets differ significantly, meaning that the accuracies attained by each of the feature experiments differ between the **kern humdrum and the MIDI variant of the dataset. The main differences in accuracy are found in respectively the relative and contour experiments.

Figure 4 clearly indicates the differences between the two versions of the Composers dataset and clearly indicates a drop in accuracy for the MIDI variant especially for the accuracy of the Relative dataset. This drop in accuracy can be explained by the way the patterns are modified during the conversion from the original **kern humdrum files to their MIDI equivalent. Evidently the conversation of the patterns has caused significant differences in comparison to the original version of the dataset. The problem lies within Music21's parser and this is probably to blame on the alpha state of the Music21 toolkit.

The low accuracy for the contour experiments in both versions of the experiment is because the patterns generated by these experiments generally contain too little information in order to properly classify each of the scores.

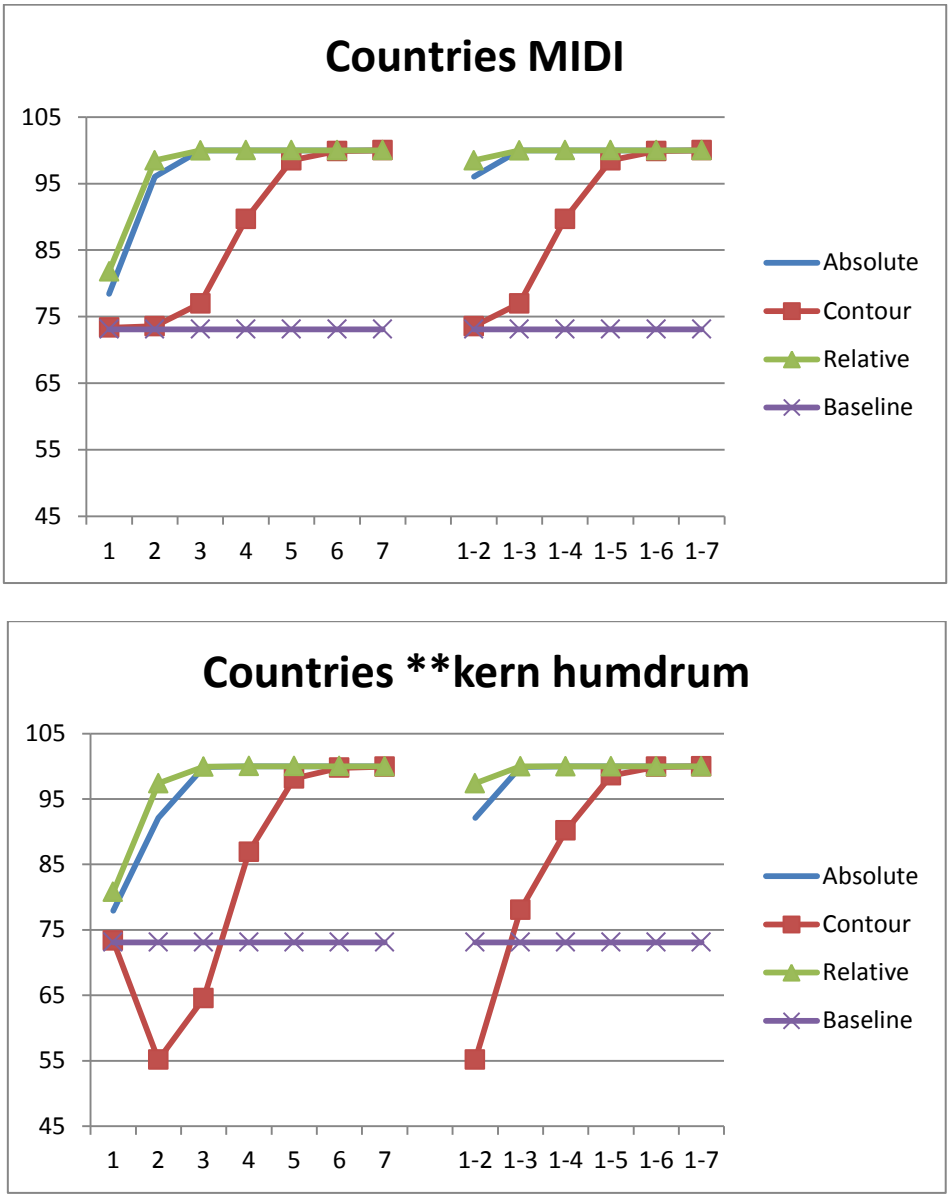
Figure 4: *Accuracies for Respectively the MIDI and **Kern Humdrum Composers Experiments.*



Even though the accuracy differs for the two datasets, the graphs have a similar shape indicating that the pattern size inflicts both versions of the dataset in a similar way. But due to the loss of some information in the conversion from **kern humdrum to MIDI, the **kern humdrum experiment generally outperforms the MIDI experiment.

Because the results for both Countries datasets do not significantly differ from each other the accuracies depicted in figure 5 are closer together.

Figure 5: *Accuracies for Respectively the MIDI and **Kern Humdrum Countries Experiments.*



Notice that in this case, even though the achieved accuracy is not significantly different there are still some differences between the two accuracies attained by the two experiments. In this case, the MIDI experiments seem to outperform their **kern humdrum counterpart as each of the pattern sizes and combinations of pattern sizes seem to perform above the baseline whereas for the **kern

humdrum experiments there are a few patterns-sizes that perform significantly lower in comparison to the majority baseline.

All in all we can conclude that even though the conversion from **kern humdrum to MIDI has its quirks and flaws, mainly due to the conversion software from Music21 handling the file types somewhat differently especially for the files that contain multiple voices. It is possible that Music21 tries grouping simultaneous notes into chords in order to prevent reaching the 320ms limit the MIDI file system has.

5.3 Experiment #3 – Testing the MIDI-only dataset

The Bodhidharma dataset is the most difficult experiment conducted in this thesis. This difference in difficulty is reflected by the majority baseline calculation which has a very low accuracy as is depicted in table 8. The attained accuracy for the majority baseline is this low because the files in the dataset are (almost) evenly represented and scores are divided over a large amount of classes (36). As a comparison, the two other experiments use two and four classes respectively.

Table 7: *Quantitative Results for the Bodhidharma dataset – Accuracy and Standard Deviation.*

	Encoding					
n-gram size	Absolute		Contour		Relative	
Baseline	1.98%	(1.10)	1.98%	(1.10)	1.98%	(1.10)
1	48.08%	(10.00)	4.95%	(2.62)	47.21%	(6.32)
2	76.78%	(3.33)	20.42%	(7.04)	71.72%	(4.31)
3	77.40%	(1.23)	36.73%	(8.10)	75.66%	(1.71)
4	80.84%	(0.65)	52.49%	(7.03)	77.34%	(1.54)
5	80.61%	(1.16)	64.23%	(4.85)	77.29%	(1.78)
6	80.50%	(1.38)	71.27%	(2.86)	77.86%	(1.39)
7	80.35%	(1.41)	73.73%	(2.24)	77.80%	(1.49)
1-2	65.03%	(6.26)	17.37%	(4.85)	65.18%	(3.52)
1-3	71.16%	(4.13)	26.88%	(4.31)	70.56%	(4.07)
1-4	74.22%	(3.04)	38.05%	(6.13)	73.05%	(3.76)
1-5	76.12%	(2.62)	46.76%	(5.52)	74.79%	(3.49)
1-6	77.40%	(2.13)	53.47%	(5.76)	75.88%	(2.87)
1-7	80.35%	(1.93)	58.08%	(4.41)	76.59%	(2.70)

The figures in table 7 illustrate that the performance of the contour functions is far below the performance of the other encodings. Even with this marginal drop in accuracy the software is still doing much better than the baseline calculation. This concludes that the software is doing its job correctly and thus we can conclude that even for this complex setup the tests gain successful results.

The results show that by implementing our system we can identify patterns which work better than the baseline calculation for a complex dataset of 36 individual classes.

By looking at the results of the single size *n*-grams in table 8, we can see that the highest accuracy is attained around an *n*-gram size of four or five whereas longer sequences cause the accuracy to drop.

This shows us that small patterns, which reoccur more frequently than larger patterns are less useful for the classification process in comparison to the larger patterns. However the possibility that one of the larger patterns reoccurs is smaller (versus a small pattern) and this affects the patterns usefulness.

The results attained from the single sized *n*-grams can be combined in order to classify on a combination of short and longer patterns. Using a combination of single sized patterns improves the accuracy attained by the classification process and it remains improving for the remainder of the experiment.

It is interesting to compare the results of our parsing software against the results from the MIREX 2005 competition (McKay & Fujinaga (2005)). During the MIREX competition some of the classes available in the had subclasses as depicted in figure 6;

Figure 6: The complex dataset used in the MIREX2005 competition clearly depicting the parent (in bold) and subcategories (with in turn additional subcategories) used in the competition's dataset. Image taken from the MIREX results paper (McKay & Fujinaga, 2005).

The toolset used in this research does not support subcategories and therefore each category is defined as its own category without the presence of the parent category. In the MIREX competition partial points were granted for classifying an individual piece into the correct parent category and full points were given for classifying both the correct parent- and subcategory. This was done to identify whether or not the parser made large errors (wrong parent category) or small errors (parent category correct, specific subcategory incorrect).

Luckily the MIREX 2005 also conducted a so-called raw experiment which is similar to the method that we implemented in this thesis. The raw experiment only counted classifications into the correct subcategory and did not grant points for estimating the parent category correctly. This lines up perfectly with the method we implemented which lacks support for subcategories and subcategories within subcategories and thus we can compare the results generated by the system's in the competition to the results generated by our own.

As we have stated before the datasets used in the MIREX2005 competition differs significantly from the one used in this thesis because Music21 cannot convert all the musical scores in the original dataset. Unfortunately the software version used to conduct the experiments in this thesis does not allow us to make up for the loss of these files and therefore the classification accuracy is biased in the advantage of the system used in this thesis.

In order to compensate this bias we will take 73.68% of the mean accuracy from the experiment that gained the best results in order to compensate for the loss of roughly 27% of the files in the dataset. This is the best compensation we can make without rerunning the entire experiment. The mean accuracy across all folds for the best performing experiment (absolute encoding) is 68.67%. After compensation this leaves a mean accuracy of 50.60% A comparison of the overall accuracy for all the contestants in the MIREX 2005 competition and our software toolkit is depicted in the table 8.

Table 8: *Comparison between the Mean Accuracy for the Systems in the MIREX2005 Competition and the System Used in this Thesis (Raw).*

Parser	Mean accuracy
Our parser	*50.6%*
Boddhidharma	46.1%
Basili et al. (NB)	45.0%
Basili et al. (J48)	41.0%
Li	39.8%
Ponce de Leon & Inesta	15.3%

The results in the comparison shown in table 8 depict our parser is doing its job better than each of the parsers which competed in the MIREX competition in 2005.

The compensation we added in order to even out the results is not entirely fair towards our experiment, as the files we excluded from the dataset because they were unparsable by Music21's interpreter are classified as incorrect automatically whereas utilizing the majority baseline and probably would have had a somewhat positive impact on the achieved percentage.

The results of this experiment do show that the system can be used to classify MIDI files, as not only do the results outperform the majority baseline calculation, but in turn they also beat the scores attained by each of the systems that competed during the MIREX2005 competition.

6. Conclusion

During the course of this thesis we have set up four hypotheses. Using the results from our experiments we can either confirm or deny these hypothesis.

H0: Converting a library of **kern humdrum files into a library of MIDI-files and running the same experiments on both the original and the converted data should result in a similar outcome.

Both **kern humdrum and MIDI experiments have attained an increase in classification accuracy by using the new implementations of the new software used in this thesis. Unfortunately we could only compare one of the datasets to each other as the polyphonic Composers dataset seems to be converted in a different way. Even though this conversion does not allow us to compare the results of both experiments to each other, each of the experiments significantly outperforms the majority baseline calculation, which means that the system is working as intended.

The monophonic Countries dataset achieved very similar results for each of the experiments and the MIDI dataset actually performed slightly more accurate classification then the **kern humdrum experiment.

The second and third experiments conducted during the course of this thesis prove that even though MIDI files are structurally and technically completely different than **kern humdrum, they can still be used in order to classify musical scores

H1: While the previous hypothesis predicts that we can get similar information out of both experiments, we also predict that some of the parameters used in the original experimental setup might need adjustment order to gain these results.

During the course of these experiments there was no need to change parameters for any of the programs. Even though the output generated for both file types is different, the classification process has similar results and thus there is no need for additional or changed parameters. During the course of this thesis there were some technical issues with the data preparation as we have illustrated in chapter 4. Most of these problems are to blame on the alpha state which the Music21 software currently resides in and are bound to be fixed in a newer release.

H2: Quantization of the MIDI timings is necessary because MIDI is known to handle the exact timing of musical events differently compared to **kern humdrum which is a precise one to one representation of a musical score.

Even though we expected that the MIDI-timing limitations were going to be troublesome, the experiments seemed to have no trouble handling and converting the files or the timings in these MIDI files.

H3: Given a dataset that solely consists of unconverted MIDI-files the expectation is that the machine-learning algorithm will perform classification of a large categorized dataset significantly better than baseline classification algorithm.

Experiment three has shown that we can apply the same techniques previously applied to **kern humdrum and MIDI files to a MIDI-only dataset. Even though the classification task for the MIDI-only experiment was more difficult than the difficulty first two experiments the overall accuracy gained by the experiment still outperformed the majority baseline calculation.

The difference between the majority baseline and the achieved accuracy is visible in each of the experiments we have conducted for both the **kern humdrum and the MIDI files. This leads us to conclude that the techniques introduced by van Zaanen in his 2010 research may be applied to MIDI without any real complications.

The biggest problem encountered during the course of this thesis is caused by the conversion software which the Music21 library implements. Not only does the software have some trouble with some of the scores available in the datasets, but also the output generated by the conversion software tends to differ per file type for the polyphonic scores.

As Music21 is still currently alpha software and the compatibility with these polyphonic MIDI and **kern humdrum files might increase over the course of new releases. Music21 is constantly updated and thus I foresee that most of the bugs that currently reside in the library's codebase will be fixed before the software goes into beta and eventually into release.

7. Future research and follow-up recommendations

During the experimental phase of this thesis we have established that the way Music21 parses the different file formats in a different way. Future research should look into the differences caused by the conversion software Music21 uses and try to determine why this difference occurs.

It might be entirely possible that newer versions of the Music21 parser will cause the output for each of the different file types to change and future research may gain different result as a result of changes made to the Music21 toolkit.

In this thesis the files that could not be read were discarded and not used in the experiments, this creates an unfair advantage whilst comparing the results of the experiments to another other experiments conducted. In order to counter this, files that are unparsable by Music21's internal conversion software and normally would be excluded from the corpus should be classified without looking at the files content by utilizing the majority baseline calculation. By implementing this technique, files that are unparsable are still taken into account in a relatively fair way whilst conducting the series of experiments and hopefully this will average out the big differences that were encountered in the results of the experiments conducted in this thesis.

The three conducted experiments were all tied to a single voice in a given song. In future research one could use the features which are bound to multiple voices to generate more complex patterns to train the system upon.

While writing the software used in this thesis, Music21's developers announced they implemented a toolset for feature-extraction capabilities into the library itself (Cuthbhert, 2011). Native support for feature-extraction simplifies and standardizes the way features are extracted and prevents the necessity of writing support for additional file-formats as Music21 supports many file-formats out of the box. Implementing the internal classification tools into a new version of the classification tools might improve the accuracy gained by each of the experiments.

This thesis has shown that classification on a more complex file-types result in a significant increase in accuracy so the next logical step is using these machine-learning techniques on digitized media (MP3, WAV etc.). The downside to this follow up research recommendation is that it is practically impossible to extract pattern information from digitized audio representation and thus the experiment cannot be conducted using the Music21 library or the software used in this thesis and would require a different experimental setup all together.

References

. About Last.fm. (2011) Retrieved 1-09, 2011, from http://www.last.fm/about

Akiko, A. (2003). An information-theoretic perspective of tf–idf measures. *Information Processing & Management, 39*(1), 45-65. doi: 10.1016/s0306-4573(02)00021-3

Beks, B. (2010). *Classificatie van polyfone muziek met behulp van traditionele IR methoden.* Tilburg University.

Celma, Ò., & Lamere, P. (2008). *If you like the beatles you might like...: a tutorial on music recommendation*. Paper presented at the Proceeding of the 16th ACM international conference on Multimedia, Vancouver, British Columbia, Canada.

. The Complete MIDI 1.0 Detailed Specification. (2001) 96.1. Retrieved 12-01, 2011, from http://www.midi.org/techspecs/midispec.php

Cuthbhert, M. S. (2011). RE: [music21] Getting the duration of a Complex Retrieved 05-31, 2011, from https://groups.google.com/forum/#!topic/music21list/SgucfbiXyQs

Cuthbhert, M. S., & Ariza, C. (2010). *Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data.* Paper presented at the International Society for Music Information Retrieval, Utrecht.

Dewi, L. (2011). *Incorporating Global Information in a Folk Song Classification System.* Tilburg University.

Feng, Y., Zhuang, Y., & Pan, Y. (2003). *Music Information Retrieval by Detecting Mood via Computational Media Aesthetics*. Paper presented at the Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence.

Garcia, E. (2008). Understanding TFIDF Retrieved 05-09, 2011, from http://irthoughts.wordpress.com/2008/07/07/understanding-tfidf/

Ghias, A., Logan, J., Chamberlin, D., & Smith, B. C. (1995). *Query by humming: musical information retrieval in an audio database*. Paper presented at the Proceedings of the third ACM international conference on Multimedia, San Francisco, California, United States.

Harmonix. (2010). Rock band 3 new features: Pro Guitar Retrieved 09-07, 2011, from http://www.rockband.com/blog/rb3-features-pro-guitar

Huron, D. (2002). Music Information Processing Using the Humdrum Toolkit: Concepts, Examples, and Lessons. *Computer Music Journal, 26*(2), 11-26.

Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* New Jersey: Pearson Education, Inc.

Kanters, P. W. M. (2009). *Automatic mood classification for music.* Master of Arts, Tilburg University.

Li, T., & Ogihara, M. (2003). *Detecting emotion in music*.

Lochbaum, K. E., & Streeter, L. A. (1989). Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval. *Information Processing & Management, 25*(6), 665-676. doi: 10.1016/0306-4573(89)90100-3

McKay, C. (2004). *Automatic Genre Classification of MIDI Recordings.* McGill University. Retrieved from http://www.music.mcgill.ca/~cmckay/papers/musictech/MA_Thesis.pdf

McKay, C., & Fujinaga, I. (2005). *The Bodhidharma system and the results of the MIREX2005 symbolic genre classification contest*. Paper presented at the International Conference on Music Information Retrieval.

Mullin, M., & Sukthankar, R. (2000). *Complete Cross-Validation for Nearest Neighbour Classifiers*. Paper presented at the 17th International Conference on Machine Learning (ICML), Stanford, California.

. music21: a toolkit for computer-aided musicology. (2011) Retrieved 28-12, 2010, from http://mit.edu/music21/

. Music 21 Documentation. (2011) Retrieved 09-07, 2011, from http://mit.edu/music21/doc/html/genindex.html

Ogihara, M., & Li, T. (2008). *N-Gram Chord Profiles for Composer Style Representation*. Paper presented at the International Symposium/Conference on Music Information Retrieval.
Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 503-520. doi: citeulike-article-id:8490795
Sanner, M. F. (1999). Python: a programming language for software integration and development. *Journal of Molecular Graphics and Modelling, 17*, 57-61.
Sapp, C. S. (2005). hum2mid Manpage for Humdrum Extras Tools Retrieved 12-01, 2011, from http://extras.humdrum.net/man/hum2mid/
Sapp, C. S. (2009). CCARH Humdrum Toolkit Portal Retrieved 12-01, 2011, from http://humdrum.ccarh.org/
Team, R. D. C. (2008). R: A Language and Environment for Statistical Computing.
Tzanetakis, G., Ermolinskyi, A., & Cook, P. (2003). Pitch Histograms in Audio and Symbolic Music Information Retrieval. *Journal of New Music Research, 32*(2), 143-152. doi: 10.1076/jnmr.32.2.143.16743
van Zaanen, M., & Gaustad, T. (2010). Grammatical Inference as Class Discrimination. In J. Sempere & P. García (Eds.), *Grammatical Inference: Theoretical Results and Applications* (Vol. 6339, pp. 245-257): Springer Berlin / Heidelberg.
van Zaanen, M., & Gaustad, T. (2011). *Influence of Size on Pattern-based Sequence Classification*. Paper presented at the Benelearn 2011, The Hague.
Wang, A. (2006). The Shazam music recognition service. *Commun. ACM, 49*(8), 44-48. doi: 10.1145/1145287.1145312
West, K. (2011). Mirex Home - Mirex WIKI Retrieved 09-07, 2011, from http://www.music-ir.org/mirex/wiki/MIREX_HOME
Yang, C. (2001). *Music Database Retrieval Based on Spectral Similarity*. Paper presented at the International Symposium on Music Information Retrieval (ISMIR 2001), Bloomington, Indiana, United States.