Using Text Mining Techniques for Classical Music Scores Analysis

Alberto Simões, Anália Lourenço, and José João Almeida

Departamento de Informática, Universidade do Minho,
Campus de Gualtar, 4710-057 Braga, PORTUGAL
{ambs,analia,jj}@di.uminho.pt

Abstract. Music Classification is a particular area of Computational Musicology that provides valuable insights about the evolving of composition patterns and assists in catalogue generation. The proposed work detaches from former works by classifying music based on music score information. Text Mining techniques support music score processing while Classification techniques are used in the construction of decision models. Although research is still at its earliest beginnings, the work already provides valuable contributes to symbolic music representation processing and subsequent analysis. Score processing involved the counting of ascending and descending chromatic intervals, note duration and meta-information tagging. Analysis involved feature selection and the evaluation of several data mining algorithms, ensuring extensibility towards larger repositories or more complex problems. Experiments report the analysis of composition epochs on a subset of the Mutopia project open archive of classical LilyPond-annotated music scores.

1 Introduction

The improving ability of score typesetting has made possible many new applications in Music Information Retrieval and Statistical Musicology. Musical catalogues, digital audio editing, real-time performance enhancement and accompaniment systems are now a reality [1].

This work focused on the analysis of music composition, studying classical composition patterns throughout different epochs. Descriptive statistics such as the counting of notes and intervals provide an overall description of each score whereas statistical discriminant, factor, and cluster analyses are often applied to the study of composition evolving and genre. Nevertheless, the analysis of symbolic music representation is, as far as we know, rare and may be considered a breakthrough.

Open source typesets like Lilypond [5,6], abc [7] or MusicXML [3] and commercial typesets like Finale or Sibelius provide machine-readable representations of music scores. Text Mining techniques can easily address the counting of ascending and descending chromatic intervals and each note duration and other kinds of metrics, and the tagging of meta-information. On the other hand, Feature Selection and Data Mining techniques can conveniently address further analysis, account for problem complexity and specific goals.

Research is still at its earliest beginnings and experiments do not yet expose the true potential of analysis. Nevertheless, text processing covers all major aspects of symbolic representations, producing datasets suitable for further mining. Future work devises the improvement (in terms of robustness) of our Lilypond parser, the processing of polyphonic scores and the replacement of semi-tones by musical interval notation. Besides classical music composition, analysis over other music genres (for instance, folk and jazz music) is to be addressed.

2 Symbolic Music Representation

Computational Musicology and Natural Language Processing are very similar areas. Music is a language (usually denoted as the universal language) and needs to be represented in a computer-readable format in order to be processed.

This section discusses the advantages of using symbolic music representation in music analysis, and evaluates the most common formats.

2.1 Audio versus Symbolic Representations

There is a parallelism between the concerns of musicologists and linguists. When analysing a music record from a Bach fugue, we are listening to the organist's interpretation of Bach's own composition. Likewise, when listening to a recorded oral story, one perceives the emotions transmitted by the reader. Even considering that interpretation does not alter the original composition, speech/audio recognition would be an issue. On the other hand, music scores/texts are almost noise free and are better at capturing work's details. Linguists take advantage of sentences boundaries and punctuation marks when analysing texts, and musicologists can use scores meta-information like pitch and time signatures to enrich their analysis.

Computational music analysis is usually based on audio (mp3, wave, aiff, midi), because audio repositories are largely available and symbolic resources are more scarce due to the transcription work involved. Both representations are valid and valuable, but while audio encompasses the performer's emotions and particular interpretation, music scores focus on composition. In this regard, musicologists prefer to work with the latest.

2.2 Music Score Typesetting

The easier way to represent a music score is using an image file. Although this format guarantees portability and human readability, it is impossible to process automatically. Optical Music Recognition and Computational Musicology may be complementary but they are independent research areas.

When looking for music typesetting software one can easily find commercial software like Finale or Sibelius. The problem is that their internal format for music representation is closed and not documented making their use difficult. On the other hand, there are some open formats such as:

– MusicXML [3]
  The most recent is MusicXML, an XML based format. It is being more and more used, although there are still few music scores encoded with it.
– abc [7] abc++ [2]
  These are simple textual formats that generate quality music scores and also MIDI files. There are some repositories with abc encoded music, but the format is not flexible enough for some specific music notation.
– Lilypond [5,6]
  This format was inspired in abc and MusicTEX, and is able to produce high quality music scores and MIDI files. This GNU project keeps a steady development, supporting a wide range of music notation, from ancient notation to modern music. Also, there are tools that support the conversions from abc and MusicXML into the Lilypond notation. Yet, Lilypond has a major drawback: a very complex and instable syntax.

After evaluating overall advantages and disadvantages, Lilypond typesetting was chosen to support the present research. Although syntax is an issue, the richness of this typesetting notation was considered an important breakthrough.

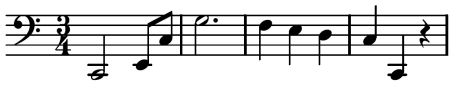


Fig. 1. Music score excerpt.

```
\relative c, {
  \time 3/4
  \clef bass
  c2 e8 c' g'2.
  f4 e d c4 c, r4
}
```

Fig. 2. LilyPond notation example.

Elaborated, real-world scores lead to complex lilypond code. It is not within the scope of this work to expose Lilypond's notation at its full extent. Figure 2 shows the code for a small score excerpt (Figure 1) in order to support further processing and analysis discussion.

3 Music Score Processing

Music score processing may involve different kinds of features depending on the aim of the computational analysis. There are atomic features such as chromatic and diatonic intervals, rhythm and key signature, and compound features like n-grams of intervals and rhythmic patterns. It is also possible to extract meta-information such as instruments, performance and composition style.

3.1 Chromatic Intervals

Lilypond supports two notation styles: absolute and relative positioning. Using absolute positioning force the user to use octavation quotes to specify in which octave the note will be placed. In relative mode, a note without octavation quotes is chosen so that it is closest to the previous one. Since most music has small intervals, pieces can be written almost without octavation quotes in relative mode. Larger intervals are made by adding octavation quotes.

To compute chromatic intervals the notation style needs to be detected (relative positioning needs a `relative` command) so the list of notes is normalized using the relative notation. Then, intervals are calculated two by two: first note with the second, second with the third, and so on. Rests are treated specially, so an interval between the note before the rest and the one after is not calculated. Also, intervals can be ascending or descending. Thus, a positive or negative integer value of semi-tones is calculated for each interval using a built-in table of consequent notes. Table 1 shows the intervals extracted from Figure 1.

Table 1. Interval calculus.

interval	c e	e c'	c g'	g f	f e	e d	d c	c c,
semi-tones	4	8	7	-2	-1	-2	-2	-12

To simplify further discussion on chromatic intervals we will represent them as "an" for ascending intervals, and "am$n$" for descending intervals. The value n is the number of ascending or descending semi-tones. Table 2 summarizes the number of occurrences for each interval.

Table 2. Intervals histogram.

semi-tones	am12	am2	am1	a4	a8	a7
occurrences	1	3	1	1	1	1

3.2 Diatonic Intervals

Diatonic intervals are less informative than chromatic intervals, as different chromatic intervals correspond to the same diatonic interval. Diatonic intervals com-

putation is quite similar to chromatic intervals calculation with a different table of basic intervals.

3.3 Rhythm

For rhythm, Lilypond uses a numeric parameter specifying each note duration. If omitted, the note duration is the same as the previous one. A value of 1 represents the Semi-breve (whole note), 2 represents the Minim (half note), 4 is used for the Semiminim (crotchet), and so on. For dotted notes, one (or more) dots can be added.

Table 3. Rhythm histogram.

rhythmic figure	count
arhythm2dot	1
arhythm2	1
arhythm4	6
arhythm8	2

To compute rhythm information the Lilypond music needs to be normalized so each note contains its duration. Then, they can be easily extracted using regular expressions. To help discussion, rhythmic attributes will be named as "`arhythm`*duration*". Table 3 summarizes the occurrence count for each note duration from the example above.

3.4 Rhythmic Gestures

Some rhythmic gestures are usual in specific epochs. For instance, the use of tuplets or syncopation is relevant for epoch classification. Tuplets detection is simple as Lilypond notation includes specific commands to typeset them. Thus, a quick search for the command and analysis of its parameters suffices for tuplets feature extraction. Syncopation and similar rhythmic gestures need to be analysed taking into account figures relative duration, and their relative positioning on bars.

3.5 Key Signature

As it is known, a specific key signature does not force the tonality for a music. For instance, a key signature without any accidental can be C major or A minor. Meanwhile, Lilypond and some other symbolic notations, include some more information than just the key signature. Lilypond can include the full tonality (pitch and type: major, minor, mixolydian, lydian, and others) in the key signature. Also, there is the problem of key signature change. There are different approaches to solve this problem: counting the number of bars for each tonality found, or just consider the first one.

3.6 Time Signature

Time signature feature extraction is similar to the key signature extraction: there is a specific command to explicitly specify the one being used, but this command can be used more than once.

4 Experiments on Classical Music Epoch Classification

Classical music epoch classification was chosen as case study for this paper. While working with Lilypond typesetting and studying classical music, Mutopia's project [1], was the primary data source. This is a fairly large and diverse open-source repository. There are over than 800 scores richly annotated with meta-information.

Data analysis involved dataset preparation, feature selection, algorithm evaluation and results assessment. Dataset preparation addressed the collection of music scores and their tagging according to composition time periods. Although music composition period boundaries are not consensual, it is fairly acceptable to define rough time limits and there were defined four time intervals (Table 4). Given that music scores for different instruments are not directly comparable, so far only monophonic scores for violin, cello, flute, recorder and clarinet were collected.

Table 4. Classification Epochs.

Name	Year Range	Size	Composers
A	1500–1600	5	J. des Prés, Banister, anonymous
B	1700–1800	11	Bach, Baltzar
C	1800–1900	4	Minkus, Giuliani
D	1900–2000	4	Grieg, Brown

Mining experiments used Weka open-source data mining toolkit [2] which allowed not only the actual mining but also feature analysis and algorithm evaluation. These experiments did not aim at the full construction of a classification model but instead to analyse its feasibility.

Dataset attributes were evaluated in order to assess their predictive ability using three different statistical approaches: Support Vector Machines (SVMs), gain ratio and Chi-squared test (Table 5). As a result, two different feature sets (given the similarity between gain ratio and Chi-squared outputs) were used in data mining experiments.

Data mining aimed at testing different analysis approaches, while determining the set of predictive attributes more suitable for this particular classification problem. ZeroR majority voting provided a reference model, while J48,

[1] `http://www.mutopiaproject.org/`
[2] `http://www.cs.waikato.ac.nz/~ml/weka/`

Table 5. Feature selection outputs.

	Most worthy attributes
SVMAttributeEval	am6, arhythm1, am24, a3, arhythm2dot, am9, am7, am2, a12, arhythm4dot, a13, a7
GainRatioAttributeEval	am7, am6, a12, a7, am2, a1, am12, am9, a9
ChiSquaredAttributeEval	am7, a12, am6, a7, am2, a1, am9, am12, am4

Naïve Bayes and SimpleCart applied information theory, Bayes theorem and cost-complexity analysis to the problem. The 10-fold stratified cross-validation technique, which is the evaluation method of choice in most practical limited-data situations [4], supported the prediction of the error rate of the learning algorithms.

Table 6. Algorithm performance over SVM proposed features.

	ZeroR	**J48**	**NaiveBayes**	**SimpleCart**
Correctly Classified Instances (%)	45.8333	62.5	62.5	58.3333
Kappa statistic	0	0.449	0.4114	0.3352
Mean absolute error (%)	0.3564	0.2024	0.2027	0.2427
Root mean squared error (%)	0.4243	0.4192	0.4263	0.3899
Relative absolute error (%)	100	56.7764	56.8712	68.0993
Root relative squared error (%)	100	98.8108	100	91.9018

	ZeroR			J48			NaiveBayes			SimpleCart		
Class	TP	FP	F1	TP	FP	F1	TP	FP	F1	TP	FP	F1
A	0	0	0	0.4	0.105	0.444	0.2	0	0.333	0.6	0.211	0.5
B	1	1	0.629	0.727	0.308	0.696	0.909	0.462	0.741	0.909	0.462	0.741
C	0	0	0	1	0.15	0.727	0.5	0	0.667	0	0	0
D	0	0	0	0.25	0	0.4	0.5	0.15	0.444	0.25	0	0.4

Figure 3 shows a simplified tree for music classification, result of the J48 model. Also it is interesting to refer that these measures are not just the obtained using the model, but they make sense regarding musicology analysis.

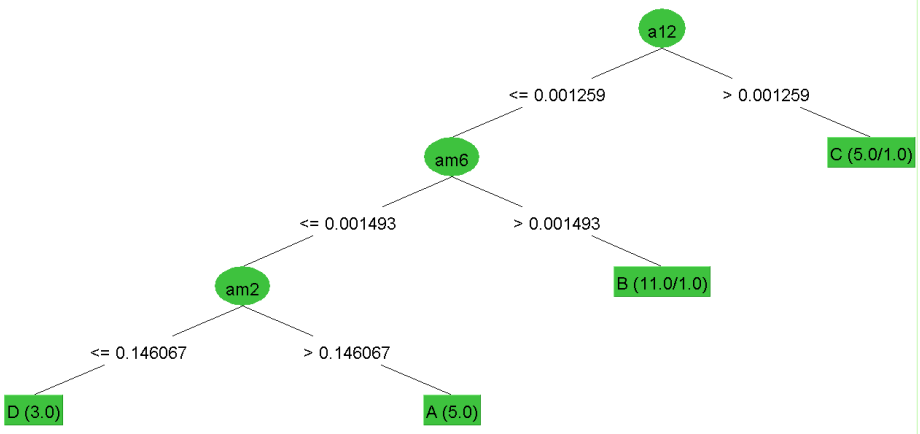
5 Final Remarks

The main contributes of this work can be divided into score processing and music composition analysis. Symbolic music representation was considered a far richer data source than commonly used audio files. In this regard, Lilypond typesetting open-source software provided a convenient representation while Mutopia project delivered a considerable number of already annotated scores. Score automatic processing involved score parsing towards the identification of rhythm

Table 7. Algorithm performance over gain ratio and Chi-squared proposed features.

	ZeroR	**J48**	**NaiveBayes**	**SimpleCart**
Correctly Classified Instances (%)	45.8333	66.6667	54.1667	62.5
Kappa statistic	0	0.5013	0.3449	0.4255
Mean absolute error (%)	0.3564	0.1836	0.2292	0.2151
Root mean squared error (%)	0.4243	0.4036	0.4461	0.3925
Relative absolute error (%)	100	51.5156	64.3156	60.3612
Root relative squared error (%)	100	95.1406	100	92.5063

	ZeroR			J48			NaiveBayes			SimpleCart		
Class	TP	FP	F1	TP	FP	F1	TP	FP	F1	TP	FP	F1
A	0	0	0	0.4	0.105	0.444	0.4	0.263	0.286	0.6	0.158	0.6
B	1	1	0.629	0.818	0.308	0.75	0.727	0.154	0.8	0.909	0.308	0.909
C	0	0	0	0.75	0.1	0.667	0.5	0	1	0.25	0.05	0.25
D	0	0	0	0.5	0	0.667	0.25	0.2	0.2	0.25	0.05	0.25



a12 ascending octavation;
am6 descending augmented fourth;
am2 descending major second;

Fig. 3. Tree representation of the classical music classification model.

figures and the calculation of chromatic intervals and key and time signatures. Music composition analysis included general statistical analysis and data mining, aiming at the construction of classification models. Specifically, models capable of identifying the classical music composition epoch.

Although the Lilypond parser needs to be improved (in terms of robustness), future work is driven by new analysis horizons. Processing is to be extended to polyphonic scores and musical interval notation should replace semi-tones. Besides classical music composition, analysis over other music genres (for instance, folk and jazz music) is to be addressed.

Acknowledgments

Alberto Simões has a scholarship from Fundação para a Computação Científica Nacional and the work reported here has been partially funded by Fundação para a Ciência e Tecnologia through project POSI/PLP/43931/2001, co-financed by POSI, and by POSC project POSC/339/1.3/C/NAC. The work of Anália Lourenço is funded by a scholarship from the *Fundação para a Ciência e Tecnologia* (Ref. POCI/BIO/60139/2004).

References

1. Roger B. Dannenberg and Christopher Raphael. Music score alignment and computer accompaniment. *Commun. ACM*, 49(8):38–43, 2006.
2. Guido Gonzato. *Making Music with ABC PLUS (v 1.0.4)*, 2005. `http://abcplus.sourceforge.net/abcplus_en-1.0.4.zip`.
3. Michael Good. MusicXML: an internet-friendly format for sheet music. In *XML Conference & Exposition*, Orlando, Florida, 2001.
4. Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1137–1145, 1995.
5. Han-Wen Nienhuys. LilyPond, automated music formatting and the art of shipping. In *Forum Internacional Software Livre (FISL7.0)*, 2006.
6. Han-Wen Nienhuys and Jan Nieuwenhuizen. LilyPond, a system for automated music engraving. In *Proceedings of the XIV Colloquium on Musical Informatics*, Firenze, Italy, May 2003.
7. I. Oppenheim. *The ABC Music standard 2.0 (draft IV, 14/8/2003)*, 2003. `http://abc.sourceforge.net/standard/abc2-draft.html`.