From Pitches to Harmony: Developing a Computational Language for Music Theory

Mathieu Vanoirbeek

Digital and Cognitive Musicology Lab (DCML)

Laidlaw Scholars Research Report

Summer 2024

Responsible
Dr. Martin Rohrmeier
EPFL

Supervisor
Joris Monnet
EPFL

Table of contents

1 Abstract

The study of music theory and analysis has traditionally relied on human interpretation and manual exploration of musical structures, such as pitches, intervals, chords, scales and harmonies. However, as the field of computational musicology grows, there is an increasing need for robust tools that can systematically model, analyze, and generate these complex musical elements. Current tools in musicology often lack the flexibility, performance, and integration needed for advanced computational tasks, limiting their effectiveness for musicologists and developers. This research project presents a specialized Python library created to solve these issues by offering a comprehensive framework for representing and manipulating musical structures computationally. The library provides features that enable detailed modeling of musical concepts, efficient computational processing, and music generation capabilities, making it a useful tool for both theoretical and practical applications. By allowing computers to understand music in a meaningful way, this library facilitates research into algorithmic composition and interactive music applications.

2 Introduction

This summer, I undertook a research project at the Digital and Cognitive Musicology Lab (DCML) at the Ecole Polytechnique Fédérale de Lausanne (EPFL).

2.1 What is Musicology?

Musicology literally means "the study of music", specifically music as a field of research, different from composition or performance. Musicology research combines and intersects with many fields, including psychology, sociology, acoustics, neurology, natural sciences and computer science. Since music is inherently human, it is interesting to study how humans perceive, create and appreciate music, which is a very broad topic.

2.2 What does the DCML do?

The DCML explores music from the cognitive, computational, musicological and psychological perspectives. It combines modern algorithmic methods, corpus research, music-theoretical expertise and experimental approaches to conduct research in four main areas: musical structure-building, musicological corpus research, the cognitive foundations of music and computational modeling. The DCML also draws on a global network of experts in musicology, artificial intelligence and neuroscience to deepen researchers' understanding of musical structures by employing cutting-edge technology. Since the field of artificial intelligence is so active right now, new tools and techniques to study music are being developed all the time.

2.3 Personal Motivation

I wanted to undertake a project in this field and this lab, because I'm a musician myself and have a passion for music, whether it's listening, playing or creating. I also love the field of computer science, and thought it would be great to combine these two passions to pursue a research project that could be both really interesting to me and useful to the DCML.

3 Research Project Description

3.1 What is my research project?

My research project consists of creating a digital library to create and model musical structures, like scales, chords, harmonies, etc. The objective of this project is to make it easier for humans to create musical structures for computers, and for computers to better "understand" those music structures. It is an implementation and elaboration of the research paper by the DCML: Johannes Hentschel et al. "Towards a Unified Model of Chords in Western Harmony", which was published in 2022.

3.2 What is the point of my research?

At the DCML, there is currently a need for this library for various research projects. Every time a project is undertaken at the DCML, a new library has to be created from scratch with limited but very specific functionalities, which costs a lot of time. Therefore the goal is to created a general and unified library that can be used for many research projects, removing the need to create a new one every time.

This library is only a small part of the ultimate objective. The goal is to combine this library which models chord and harmonies with another library which focuses more on the rhythmic aspect of music, to create a complete architecture which could allow for any music to be represented.

3.3 How is it going to be created/implemented?

The library is implemented using the programming language Python, and will eventually become an open-source library available to download on PyPI, Python's repository of software.

The idea is that this library will constantly be improved by others, who can keep implementing new functionalities in the public GitHub repository. On top of that, this library has the potential to grow through community contributions and collaboration with other open-source music projects, and possibly integration with other music software.

4 Analysis and Findings

4.1 Challenges and Difficulties

The purpose of the pitchtypes library is to be a core library that can be used for any computational or creational musicology project. Therefore, it has to be very general and very performant. These were the two main challenges that I faced during this project. In this section, I will discuss how I overcame these challenges.

This library has to be a practical tool for many uses: algorithmic composition, music analysis, computational musicology and many more. This is why the library has to be as general as possible. Therefore, the architecture is built from the ground up, starting from the most simple musical object, pitches, all the way up to very complex musical objects, like chords and harmonies. Every object is derived in some way from pitches and intervals, which can be considered as points and vectors in a tonal space, a conceptual representation of how humans perceive and relate the distances and connections between pitches, harmonies, and keys in music, creating a sense of musical structure and progression. Understanding and implementing this architecture was one of the challenges I faced, as it required finding a balance between ensuring user-friendliness and maintaining the rigor needed to represent music theory accurately. The library's modular approach is crucial in addressing this, allowing users to create and manipulate musical objects at different levels of abstraction, whether they are dealing with basic elements like individual pitches or more intricate structures like harmonic progressions.

Another challenge I had to overcome was the need for optimal performance in this library. Everything this library needs to do, whether it's object creation or computation, has to be as efficient and swift as possible. However, the programming language Python struggles with performance, because it is a high level programming language. High-level programming languages are generally less performant because they introduce abstractions which add computational overhead. These languages also distance the programmer from direct hardware access, preventing low-level optimizations that are possible in languages closer to machine code. I was able to mostly overcome these issues by using performant data types that are optimized in Python, so that the computational overhead is much smaller.

> "There are many solutions to a problem, but only one of them is the best"

This is something that I heard many times during the project, and is something that I will definitely remember. I learned to spend more time thinking about how I would implement a

feature rather than implementing it, because this saved me time and ensured higher quality in my work. I strove to find a simple, but elegant solution for any problems I encountered, which made the code performant and at the same time much more maintainable for future developers.

4.2 Objects and Definitions

1. Pitches
A pitch is the specific frequency of a sound, which determines how "high" or "low" the sound is. Pitches are most commonly represented in computers as MIDI values, a range of values between 0 and 127 that each have an associated frequency. For example, the MIDI value 69, the pitch A4, has a frequency of 440 Hz. The piano has 88 keys, which cover the range 21 to 108 of the MIDI values.

2. Intervals
An interval is the distance between two pitches. In music, this distance is often measured in terms of semitones, a step of one key on the piano. Pitches are seen as points, and intervals can be seen as vectors between two points.

3. Chords
A chord is a set of pitches that are related in a specific way. These pitches are chosen to create a particular sound or develop a harmony. For example, a minor chord consists of three pitches that created a sad or melancholic sound, very common in all kinds of genres in western music.

4. Scales
A scale is an ordered sequence of pitches, typically within one octave. Scales are the building blocks for melodies and harmonies. They are most often defined as a list of intervals between each pitch in the scale. For instance, a major scale is represented as [2, 2, 1, 2, 2, 2, 1], where one can clearly see the the half steps and whole steps in the scale.

5. Harmonies
Harmony refers to the combination of different musical objects played in succession, creating a certain sound. Harmonies are often formed by playing chords or by combining multiple melodic lines.

In summary, pitches can be thought of as points, intervals as vectors, chords as specific sets of pitches, scales as ordered lists of pitches, and harmonies as the result of combining these musical structures to create interesting progressions in music.

4.3 Applications

There are many applications for this library. Some examples are:

1. Tools for Researchers
Researchers in music theory and musicology can use this library to analyze and explore musical structures systematically. The library facilitates the study of pitch relationships, harmony, and scales by providing tools to model and manipulate these musical elements. This helps musicologists test theories, analyze musical compositions, and develop new insights into musical patterns and structures.

2. Music Computational Modeling
The library also enables the creation of detailed computational models of musical structures such as pitches, intervals, chords, and scales. By using these models, people can create music efficiently and easily manipulate the data. Computational modeling can also support the development of algorithms for automatic music analysis and processing.

3. Music Generation
The library can be used to create algorithms that generate music by understanding and applying principles of harmony and melody, or simply create the musical structures for users. The goal is to allow computers to better understand musical structures.

4. Visualization
The library can assist in visualizing musical data and structures, making complex musical concepts more accessible and understandable. Visualization tools can represent pitch classes, harmonic relationships and rhythmic patterns with charts, graphs and trees. This could help in both educational contexts, where visual representation helps in teaching music theory, and practical contexts, where visual tools support music analysis and creation.

This library offers a set of tools for musicology, enabling music analysis, computational modeling, and music generation. Its capabilities in computational modeling facilitate efficient music creation and data processing. Additionally, the library supports music generation by enabling computers to understand and apply principles of harmony and melody.

4.4 Future Improvement

This library can easily be improved in the future. The lab already envisions some future projects that could further develop the library and make it a practical tool for all musicology research. Since it will be open source, anyone can contribute and implement their own features

that fit their needs.

There are many small models for specific features, but the point is to bring everything together and create a common tool with a lot of features for many projects. This solves the issue of having many scattered models and organizes them into one core library.

4.5 Comparison to other tools

In the field of musicology, several tools and software platforms are widely used for music analysis, composition, and research. However, these tools often have limitations that could be addressed by this library. Here are some examples of existing tools and their shortcomings:

1. Music21
Music21 is a Python-based toolkit for computer-aided musicology. It supports music theory analysis, algorithmic composition, and music information retrieval (MIR). Disadvantages:

1. Performance Limitations: Being a general-purpose tool, it can be less efficient for highly specialized tasks or large-scale data processing.
2. Limited Harmony and Melody Generation: Music21 has some tools for music generation, but they are not as advanced or customizable as what could be achieved with this library.
3. Lack of Music Understanding: While it provides some visualization features, this tool does not establish a framework for machines to comprehend and process music theory.

2. MuseScore
MuseScore is a free and open-source music notation software that allows users to compose, arrange, and share music. Disadvantages:

1. Focus on Notation: While excellent for notation, it is not designed for deep musicological analysis or computational modeling.
2. Lack of Algorithmic Composition Tools: MuseScore is not intended for algorithmic or computational music generation.
3. No Advanced Analytical Tools: It does not offer tools for sophisticated analysis of musical structures like harmonic progressions or pitch relations.

This library could offer a more flexible and integrated approach to modeling musical structures, allowing users to define, manipulate, and analyze pitches, intervals, chords, and scales with more ease and precision. This library is also optimized for performance, addressing the limitations of tools like Music21 when dealing with large datasets or complex computational tasks. While existing tools offer some basic functionality, this library could provide a more robust framework for generating music based on an understanding of harmonic and melodic structures.

In summary, while existing tools serve important roles in the field of musicology, this new library could offer a more specialized, efficient, and comprehensive solution for modeling, analyzing, and generating musical structures.

4.6 Further uses and applications

1. Integration with AI and Machine Learning
This library could also be used in conjunction with Artificial Intelligence and machine learning algorithms for tasks such as automatic music composition, style imitation, or music recommendation systems. Machine learning models could be trained on the structures and patterns modeled by the library, leading to more sophisticated music generation or analysis tools. For example, there exists a potential for creating AI composers that can generate music in the style of specific composers or genres.

2. Cross-Cultural Music Analysis
This library also has the potential to model and analyze musical structures from different cultures and traditions. Since this library is such a general library that can be customized for any music style or genre, the library could be adapted to handle non-Western scales, harmonies and rhythmic patterns, providing a tool for musicologists studying world music.

3. Real-Time Applications
This library also has the capabilities for real-time applications such as live music analysis and live performance. The library could be used in environments where music needs to be analyzed or generated in real-time, which it can handle thanks to its optimized performance as a core library.

5 Conclusion

In conclusion, the development of this library provides a powerful and flexible tool for the analysis, modeling and generation of musical structures. By allowing musicologists and developers to easily manipulate pitches, harmonies, and melodies, the library creates a bridge between traditional music theory and computational methods. As the library continues to be developed, it will play an important role in advancing the field of computational musicology.

Over the course of this project I learned a lot and developed a lot of valuable skills. Working in a team taught me a lot about communication, thinking as a group and agreeing on solutions to certain problems. I also greatly developed my coding skills, and learned to use tools from the professional world.

I am grateful for the opportunity to have worked on this project and collaborated with such a dedicated and supportive team at the DCML. I look forward to applying the knowledge and skills gained from this experience to future projects.

6 Acknowledgements

I would like to thank Dr. Rohrmeier and the DCML for hosting me for this internship, and I would like to thank my supervisors Joris Monnet and Johannes Hentschel for guiding me and helping me throughout this research project. I am also grateful to the Laidlaw Foundation team for this great research opportunity.

7 References

Hentschel, Johannes, et al. Towards a Unified Model of Chords in Western Harmony. Universidad de Alicante, 2022. `http://rua.ua.es/dspace/handle/10045/123678`