Random Generation of Folk Music through a Corpus-Based Markov Model and Python/Music21

Overview

The aim of this project is to be able to randomly generate folk melodies that are near-indistinguishable from composed melodies. While similar projects have been undertaken by others before, this methodology addresses some of these projects' shortcomings, and is overall more successful. Via a corpus-based computer model of various parameters of folk music, the melodic content generated through this code nearly always follows normative musical syntax and structure without being derivative of any one work or set of works in the corpus.

Previous Attempts

Though this is certainly not the first attempt at a project like this, only two previous attempts—that represent polar extremes—will be briefly discussed. It is between which this model is situated.

In *Virtual Music: Computer Synthesis of Musical Style*, David Cope presents a "musical game," in which he invites the reader to listen to or play various musical examples and then determine which ones were composed by people, and which were composed by his computer program. Of most interest to the present discussion is the third iteration of this "game," wherein the pieces being presented are either mazurkas composed by Chopin or by a computer program that is "composing" like Chopin. While admittedly it is difficult to tell them apart, the reason for this might not be that the program is really thinking or making choices like Chopin, but rather that it is lifting large blocks of material and repurposing it. In this way, it does not seem a stretch to imagine that Chopin composed the work, but it is simply because one might recognize very distinguishable motives from his other works being reused in this context. Thus, the resulting pieces do not feel so much original as they do derivative, an unintentional plagiarism of sorts. This points out a key goal of this project: to generate music that seems original and composed *without* sounding completely derivative of the corpus from which it comes.

On the opposite end of this spectrum are Dorien Herremans, Kenneth Sorensen, and David Martens, and their paper "Classification and Generation of Composer-Specific Music Using Global Feature Models and Variable Neighborhood Search." Though their methodology is different than Cope's, their goal is very similar in that they attempted to create a program (and ultimately a mobile application) that generates counterpoint in a combination of the styles of Bach, Beethoven, and Haydn. While the issue with Cope's work is that it might be accused of sounding remarkably derivative of the source material, Herremans, Sorensen, and Martens have the exact opposite problem in that the generated counterpoint ultimately sounds nothing like that of the composers from which it was modeled.

Their problem is fundamentally two-pronged. The first of these is that the counterpoint they generate is not stylistically sound to begin with. While they claim that it avoids breaking almost all contrapuntal rules, they do not seem to have an understanding of which rules must absolutely not be broken in order to sound like reasonable counterpoint. A brief examination of the generated counterpoint reveals unallowable dissonances, parallel fifths and octaves, direct octaves, etc., all of which completely undermine the overall stylistic soundness of the work.

Perhaps the more important problem with their methodology is the way in which they implement the composer parameters on the counterpoint. Instead of generating counterpoint based on a set of composer-specific rules, they instead continually generate random counterpoint (via the aforementioned process) and then test it for composer-specific features discerned by a computer through a machine-learning process. If the counterpoint does not pass the parameters given to it by the computer, new counterpoint is generated in its place until it does pass the test. Unfortunately, these parameters are very unmusical. Often they include decision-tree based questions such as "Is the average melodic interval greater than or equal to .318?" While questions like this might be important in discerning whether or not a work as whole was composed by one composer or another (a previous project of theirs), these parameters have very little to do with the way we hear and comprehend music. Thus, while the algorithm undoubtedly believes that the counterpoint it is generating is an accurate representation of whatever composer the user has specified, the end result is underwhelming, to say the least. In fact, adjusting the composer preference sliders on the mobile app causes almost no discernible musical difference.

Approach

It seems that the best approach to this task is to combine a corpus-based analytical model with musical intuitions (when and where appropriate). The corpus used here is the collection of German folk songs (in kern score format) from the Essen Folksong Collection.[1] The current methodology is actually very simple, but produces compelling results.

First, eight measures are generated through a mixture of random and predetermined elements. There are currently six separate lists of possible measures (two per possible time signature), which are stored as "quarter length" values (a parameter that Music21 understands essentially as note duration). Bars 1-3 and 5-7 are randomly selected from the first list of bars,

[1] Taken from: https://kern.humdrum.org/cgi-bin/browse?l=essen/europa/deutschl/boehme

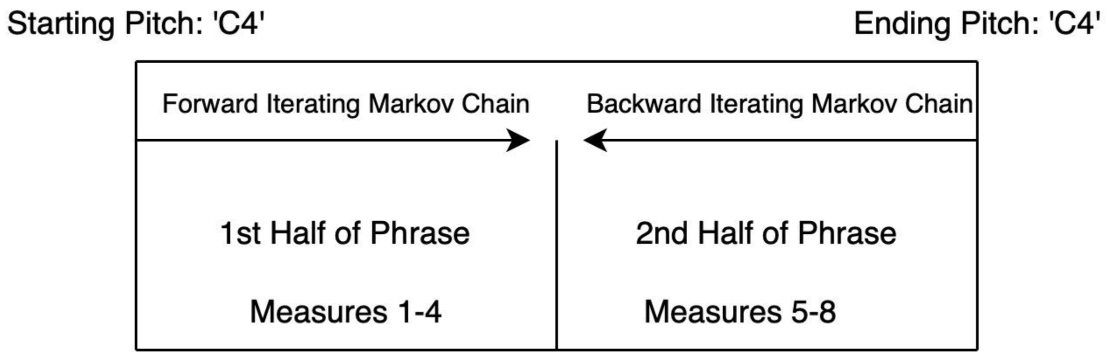
whereas bars 4 and 8 are selected from a separate list of measures that are appropriate for cadential points. Allowing for the possibilities of different time signatures (4/4, 3/4, and 6/8), there are two lists of measures per time signature (interior and cadential measures). The code allows for variability of phrase length (though 8 measures is the default), and has the option to utilize the same rhythms for bars 5-7, (assuming this is an 8-bar phrase) that are randomly selected for bars 1-3, creating a rhythmic parallelism between the first and second half of the phrase. While the possibility of generating rhythm from the corpus itself exists, this method actually produces more favorable and musically interesting results.

The generating model chosen for this project is a Markov chain with memory (of order *n*). That is, future states of the chain are dependent on previous ones. The extent to which previous states are considered corresponds to the selected order, and as will be demonstrated below, varying orders of *n* will deliver drastically different results. The concept of order can be quite delicate when generating musical melodies: on the one hand, too low of an order will fail to yield a satisfying and coherent melody, while too high an order may begin outputting melodies with large recognizable sections from the training corpus.

In order to get this model, each work in the corpus is transposed to a tonic note of 'C' (easily achieved through the metadata present in the kern files), and all accidentals are set to natural. This negation of all accidentals ensures that there are no cross relations or modal mixtures occurring, especially on lower-order Markov processes. Once the piece is transposed, two master Markov models are created for all the works in the corpus – one that generates tuples going from the front to the end, and the other generating from the end forward. Once these models are generated they are stored locally to aid in processing time.

These models are then used to fit pitches to the already-existing rhythms, one model completing measures 1-4 from the front, and the other completing measures 5-8 starting from the back, shown in Figure 1. Because Music21 is able to define a set of starting keys for each Markov chain, and since the chains work inward from either end of the piece, it can be ensured that the work begins and ends on the tonic note. The final step, then, is to make sure that the break between bars four and five (or, the first and second half of each phrase) is as smooth as possible. While it often requires no adjustment at all, the fix is simply to adjust the octave of the second half of the piece up or down if the first note of measure five is more than a perfect fifth away from the last note in measure four.

Figure 1.



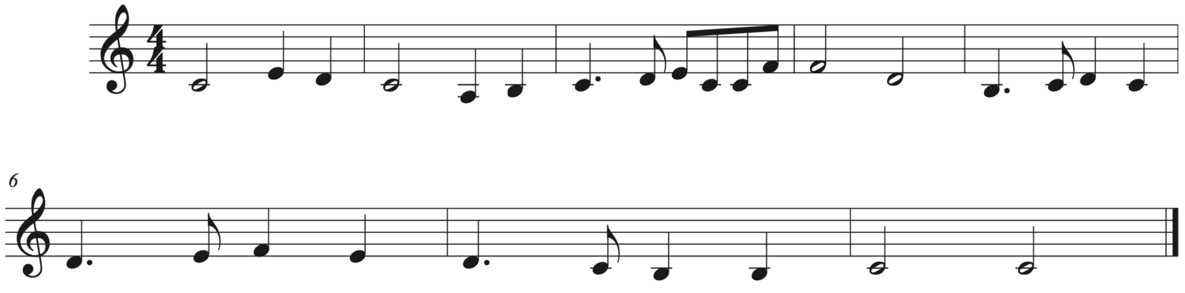
Due to the size of the corpus (approximately 600 folk songs), relatively high Markov orders can be generated without creating something that is completely derivative. Table 1, below, shows current data for each Markov order.

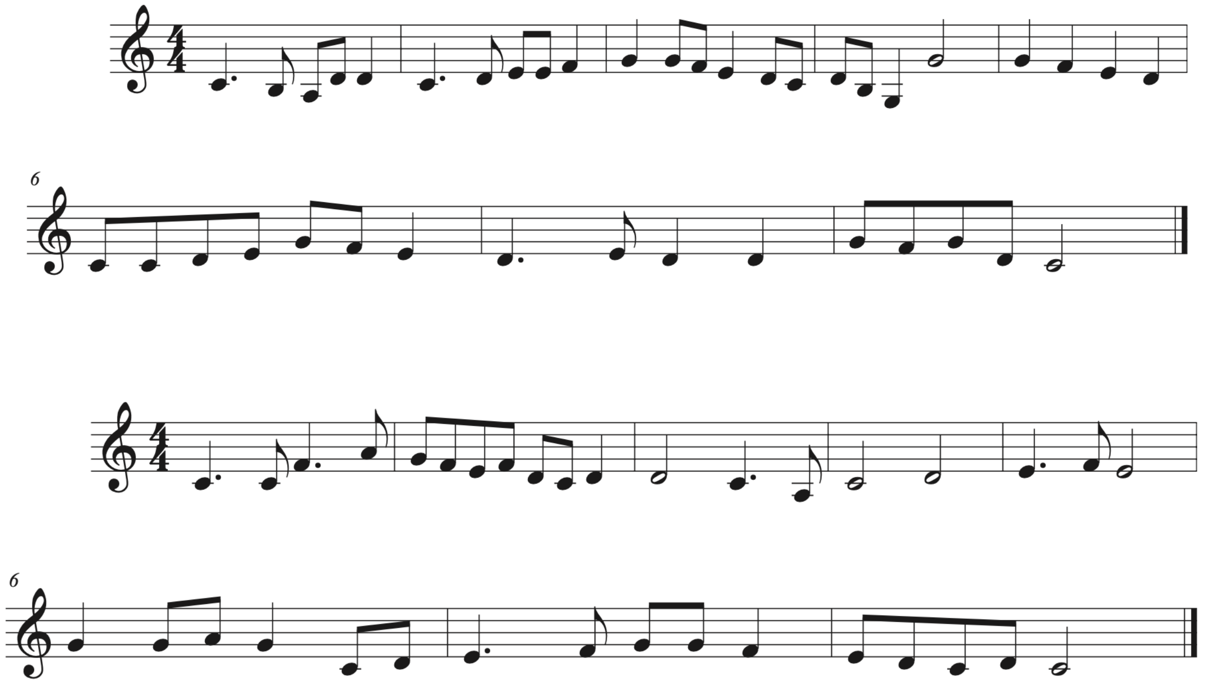
Table 1.

ORDER	AVG. CHOICES	N-GRAMS	CONTINUATIONS
1	2096.529	17	35641
2	244.867	143	35016
3	47.435	725	34391
4	13.621	2479	33766
5	5.474	6054	33141
6	2.963	10974	32516
7	2.013	15839	31891
8	1.601	19522	31266
9	1.406	21793	30641
10	1.302	23058	30016

Markov Pitch Data for Orders 1-10

Even at sixth-order Markov chains, there is still an average of about three continuations for each n-gram. Below are several examples of what might be generated at the sixth order. Anything beyond the 6th order, however, yields less than 1.5 choices per n-gram, and is more likely to lift large melodic chunks directly from specific folk songs.




Examples of Generation at Sixth Order

Continuations/Conclusion

Though results achieved through the early stages of this methodology are successful, this is only the beginning of what could unfold as a much larger project. The first component to improve is the rhythm generation. While the resulting rhythm itself is stylistically sound and convincing, it is currently based solely on random choice and hard coded musically intuitive measures of rhythm, rather than on the works in the corpus. This is for a number of reasons, the foremost of these being a lack of rhythmic diversity in the corpus. Though the tools and the ability to parse metrical and rhythmic information from the corpus are available, the results are underwhelming. There are only 44 pieces in the entire corpus in 4/4, and the rhythms that result from a corpus-based model often turn out to be almost entirely quarter notes. Another reason for this is the current absence of rests within the generated melodies, which can hopefully be integrated into the project.

A second issue regarding rhythm is the superimposition of pitches onto rhythm, rather than a simultaneous generation of both. The issue at hand is that though the melodic content of a melody may be convincing, there are often times when the lack of coordination between melody and rhythm cause an unsatisfactory result. For instance, imagine if the pitches to Happy Birthday were superimposed upon the rhythmic framework, but were displaced by a beat (let's say by omitting the pickup eighth notes), the resulting product would not be musically satisfying. This problem could be mitigated by including additional criteria for fitting the pitches to rhythms. Making these two improvements could pave the way for future additions to this project, including longer melodies, clearer and more varied formal implications, multiple/contrapuntal lines, etc.

Overall, this Markov-based model for generating folk melodies has been largely successful. There are certainly improvements that could be made, though as it currently works, it addresses the perceived shortcomings and improves upon the work of others that have undertaken similar projects. The methodology and execution of this project finds a nice middle ground between the issues of being too musically derivative of the source material, and being too musically insensitive to the given style.

References

Herremans, Dorien, and Kenneth Sörensen. "Composing First Species Counterpoint with a Variable Neighborhood Search Algorithm." *Journal of Mathematics and the Arts* 6, no. 4 (2012): 169-89.

Herremans, Dorien, Kenneth Sörensen, and David Martens. "Classification and Generation of Composer-Specific Music Using Global Feature Models and Variable Neighborhood Search." *Computer Music Journal* 39, no. 3 (2015): 71-91.

Herremans, Dorien, David Martens, and Kenneth Sörensen. "Composer Classification Models for Music-Theory Building." *Computational Music Analysis*, 2015, 369-92.

Pachet, François, and Pierre Roy. "Markov Constraints: Steerable Generation of Markov Sequences." *Constraints* 16, no. 2 (2010): 148-72.

"Review of Michael Cuthbert, Music21: A Toolkit for Computer-aided Musicology (http://web.mit.edu/music21/)." MTO 19.3: Tymoczko, Review of Cuthbert, Music21. http://www.mtosmt.org/issues/mto.13.19.3/mto.13.19.3.tymoczko.html.

Temperley, David. "What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered." *Music Perception: An Interdisciplinary Journal* 17, no. 1 (1999): 65-100.

Temperley, David. *Music and Probability*. Cambridge, MA: MIT Press, 2007.