Special Interest Group Paper: Computer Composition of Counterpoint

Arthur Guo, Nzo Liu, Xinyuan Chao, Xinyue Fan, Haiyi Liu

Graduate Seminar in Music Technology

Prof. Paul Geluso

New York University

Introduction of Algorithm Composition

This part of contribution is created by Xinyue Fan.

Introduction

Algorithmic Composition, also known as automatic Composition. It's a system to simulate the composer's creative thinking, which will greatly improve the automation of the composition. It can make composers work more efficiently and increase the possibility of the cross-development of music and artificial intelligence. Since the 1950s, various AI-based algorithms have been used in composition. It mainly includes Markov chain, neural network, genetic algorithm, as well as hybrid algorithms.

Process of Algorithm Composition.

In the process of algorithm composition, the composer firstly input a creative motivation to the system, These motivations can be manually selected from a large number of small pieces of music randomly generated by the machine or can be selected from a database of existing collecting pieces of different musical styles. Then the system can generate a piece of music with the specified form structure according to the established musical rules and styles trained by various algorithms such as Markov chain, neural network, genetic algorithm. The key of Algorithm Composition is how a composer intervene in this human-computer interaction process to make music more efficient and creative.

Markov Chain

Markov Chain is a simple but interesting technique which can be used to select notes in turn according to a conversion table. The conversion table acts like a function transformation table with which the last note can be predicted by the probability of the previous note. The

transformation table should be manually constructed according to certain rules and get nested with certain musical styles. With Markov Chain, the samples of certain styles (such as certain composers or certain periods) can be collected and a transformation table will be constructed with these samples and ultimately corresponding certain styles can be automatically generated through this transformation.

Artificial Neural Networks

In addition, there are methods based on machine learning and deep learning to train data and compose music, commonly known as Artificial Neural Networks, which mimic the behavioral characteristics of biological Neural Networks. This is a mathematical model of distributed parallel information processing algorithm, in which the recurrent neural network can learn the musical form structure, the use of long and short-term memory (LSTM) neural network can make the music more complete. The advantage of an artificial neural network is that it can learn large numbers of features of music, such as pitch, Timbre, speed and force, but it requires a large number of samples to train. In addition, Genetic Algorithm is often used as an optimization tool to make music more euphonious.

The Pioneer: David Cope

This part of contribution is created by Xinyuan Chao.

Introduction

David Cope is an algorithm composer. He developed the Experiments in Musical Intelligence (EMI or Emmy) system, which is an analysis program that analyzes music Cope has entered into the EMI's database, and this input will be used to guide the creation of new works in the same style. EMI is capable of producing music works in the same style as the works of deceased composers, such as Bach's creations, instrumental concertos and suites, Mozart's sonatas and Chopin's nocturnes.

EMI is based on three basic principles that have underpinned the creation of artificial music to this day:

1. Deconstruction: Analyzing music compositions and separating them into parts.

2. Signatures: Identifying commonality, which signifies and characterizes a style of a genre or composer.

3. Compatibility: Recombine pieces, patterns and styles to create new original works. (Moura, F. T., 2018)

Music composed by EMI

David Cope released lots of albums composed by EMI. In his album *Bach by design*, because the fingering in this album's music is not easy to play, Cope used Disklavier instead. Disklavier is a form of piano that uses sensors and solenoids to recreate the performance of an actual piano playing. Feedback on this album has been negative, not because of the composition but because of the perception that the machine playing sounds stiff.

In another album *Classical Music Composed by Computer*, Cope performed in the styles of Bach, Beethoven, Chopin, Cope, Joplin, Mozart, Rachmaninov and Stravinsky by EMI, and was interpreted by a live orchestra. The album received good reviews and attracted the attention of many people in the classical music and artificial intelligence communities. These shows that Emmy was capable of creating a work of astonishing depth and range. (Chris, G, 2015)

Turing test

The EMI also passed the Turing test. Professor Douglas Hofstadter of the University of Oregon organized a music form of the Turing test. Pianist Winifred Kerner played three songs in Bach style: one written by EMI, one by Dr. Steve Larson, and the last by Bach's actual work. As a result, the audience chose the EMI's as the actual Bach, while believing that Larsen's work was the one created by a computer. (Chris, G, 2015)

Creativity

However, there are some debate about Cope's research. Cope believed that computer has a sense of creativity. He thought that creativity is simple, but consciousness and intelligence are hard. In his book Computer Models of Musical Creativity, Cope says that "Creativity depends on the integration of its various characteristics into a unified whole" and "Creativity depends on aesthetic values...". But critics argue that Cope's research is concerned with mimicking music-style systems rather than studying creative system. Most definitions of creativity contain some notion of value, and a process is not considered creative. Geraint A. Wiggins (2012) thinks that Cope neatly (re)defines his work into creativity studies in particular without a component of self-evaluation by defining creativity himself. (Geraint A. Wiggins, 2012)

"Re-synthesis": Alex Bainter

This part of contribution is created by Haiyi Liu

Introduction

Rule-based machine learning is a term intended to include any machine learning method that identifies, learns, or evolves 'rules' to store, manipulate, or apply. The defining characteristic of a rule-based machine learner is to identify and exploit a set of relational rules that collectively represent the knowledge captured by the system (Bassel et al., 2011). Alex Bainter (2019) uses Markov Chain algorithms from rule-based machine learning to achieve the effect of producing endless performance of the track without repetition.

Analyzing the algorithm of *aisatsana*

The original music he used was *aisatsana* released by Aphex Twin, which is much simpler than most pop music. It is only played on piano and contains 32 phrases, each phrase contains 16 beats, and there is no obvious chorus, verse, etc.

Thus, aisatsana can be described as an algorithm, which can be divided into two parts:

1. Every 16 beats, *do something.*
2. Play a 16 beat phrase.

Bainter considered that if only separate these 32 phrases and play a phrase randomly every 16 beats, though it might work well for a short time, it would eventually repeat itself. He thought it is important that in addition to playing the original phrases, the system could also create and play new ones. Of course, these new phrases have to sound like original phrases. Therefore, Bainter adopted the technique which used Markov Chains.

Building the system with Markov Chain

A Markov chain records a set of possible states and the probabilities of transitioning from one state to another. Therefore, all we need to create a Markov chain are states, and the probabilities of transitioning from each state to the others.

With this Markov Chain, we can generate phrases by beginning at start and following the transition until we reach end. This is called ‘walk chain’. It can be expressed as:

1. Begin at *start*.
2. Record the current state. Select one of the transitions from the current state based on the probability of each transition, and follow that transition to the next state.
3. Repeat step 2 until you’ve reached *end*.

Then, Bainter built the system in JavaScript and parsed the MIDI files using JSON. Since the phrase has some notes on the half beat, he doubled the BPM and uses a half beat as a unit of account. At this point, each phrase consists of 32 half beats. He then converted the phrase to states and calculated the transition probability. He uses a library called "markov-chains" to take phrases as input and used its ‘.walk()’ function to return the list of states created.

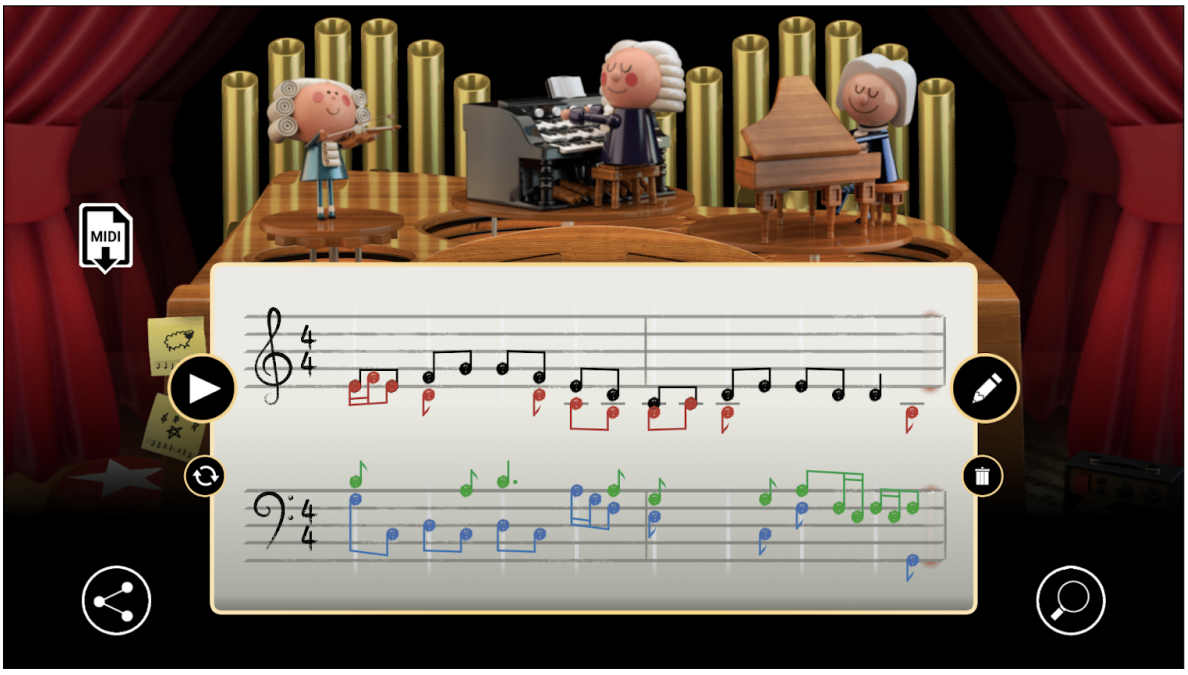
The system Bainter produced is capable of generating more than four million unique phrases that all sound similar to the original. The input source can be replaced by another piece of music, and the same algorithm can be used to generate infinite unrepeated versions of different music. He also provided the code so that we can use it to create our own music.

Coconet by Google Magenta

This part of contribution is created by Enzo Nianze Liu

Introduction

Magenta is an open source project launched by Google, aiming to explore the possibility of training machine learning models as useful tools in creative areas such as music and visual art. On March 21, 2019, in order to celebrate Johann Sebastian Bach's 334th birthday, Google Doodle team collaborated with Magenta team and offered an interactive online counter point generating game, which is backed by a machine learning model trained to mimic the style of Bach. Anyone could write their own melody by clicking on the treble clef, and then by hitting the play button below, the model will harmonize the input melody with rest of three voices.



How Computer Composes

Traditionally, when computer generates music, the most natural way is to do the job in a sequence, under an orderly manner, because when computer generates music, the only thing it understands is just numbers, or in our context, statistics and probability - it does not feel. Given a note, such as C4, and another note, say E4, the computer has to calculate all the possibilities of the next note, and return the most possible notes to fill in the next position. If the model sees millions of times that C4, E4 and G4 always show together during training, the likelihood of generating a G4 given C4 and E4 is then slightly higher -- and we can say that the model 'learns' the 'triad' now, even though the computer knows nothing about the concept of 'triad', it only knows that G4 is most likely to show up in the context of C4 and E4. (After all, C Major triad is formed only because human's perception decides the sound feels good). Thus, before determining the very first note of music, the computer has to have a complete view of the whole piece, since as long as the previous note is determined, there's no way back to refine it anymore -- the note has to be generated in a sequence in one go. This orderly generation requirement is very strict, which leads to the machine learning model behind the Google Doodle online interactive game: 'Coconet'.

Model Introduction: Coconet

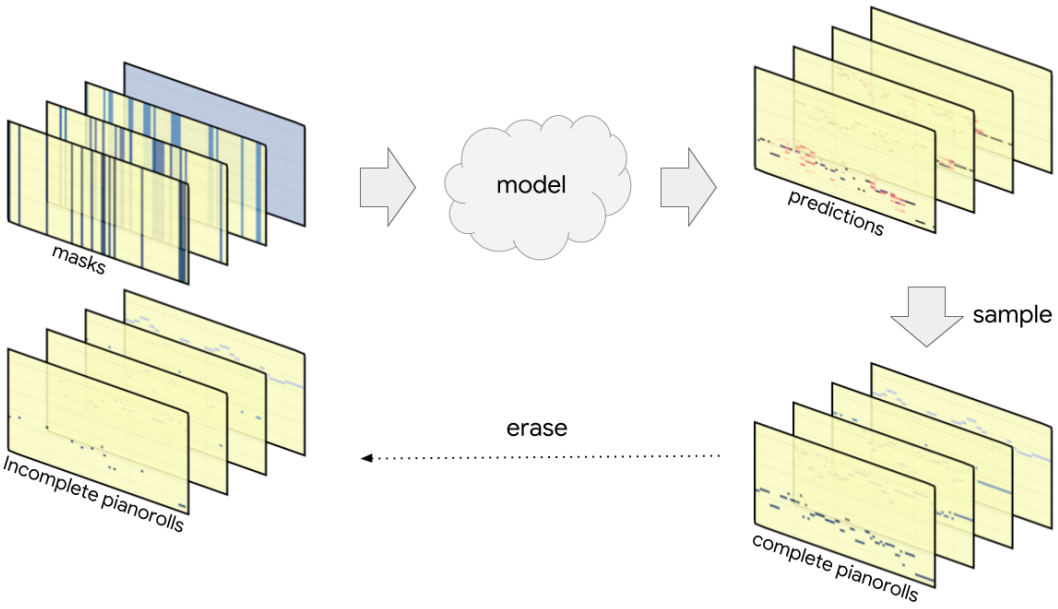
The machine learning model Coconet is basically a convolutional neural network but is integrated with some creative process. The details of the model are described in Huang, C A's paper "Counterpoint by convolution". Here we'll just go over the basics of the model structure and see how it works to generate four voices counter points and harmonize any given input melody.

The biggest idea behind the model is to mimic the process of human composition. The key observation here is that, when human beings compose, the composer will seldom write the music in sequence: composer tends to rewrite the same piece back and forth over and over again in order to get the better refined result that finally meets his or her standard. In other words, the composition is typically done in an orderless manner. To achieve orderless composition by computer, two techniques is created: ***Orderless modeling & Gibbs sampling.***

The basic idea is to use Gibbs sampling to generate music in multiple iterations, and in every iteration part of music scores is erased, so that we can use Orderless modeling to come up with the independent conditional probabilities for missing notes. For the same example of C4 E4 G4 triad, if we mask the E4 in the middle, the model should be able to generate E4 given the context of C4 in the first beat and G4 in the 3rd beat. In math, the model is basically computing:

$$P(x_2|x_1, x_3) \text{ where } x_n \text{ means score at beat n}$$

Gibbs sampling is just a fancy word for phrasing our repeated iterative music generation/modification in multiple passes, and orderless modeling guarantees that no matter which note is randomly erased by a mask in a random iteration, the model is able to generate a conditional distribution for the missing score.



The outcome of Coconet is very interesting: given a partial piece of scores for a fixed length, the model is able to complete the music based on how it gets trained. In the Google Doodle example, the model learns from "a dataset of 306 chorale harmonizations by Bach", according to Magenta Blog post. This interactive game is still available to play today by clicking this link.

Our demo: *Counterpyt*

This part of contribution is created by Arthur Jinyue Guo.

Introduction

Since counterpoint composition has very strict and explicit rules, I had the idea to write a program to automatically generate counterpoint pieces for a given *canctus firmus*. Luckily, I am taking the Tonal Theory course this semester, so I decided to gradually add more complicated rules while we learn more advanced counterpoint techniques. Additionally, we want to figure out if the strict rules can work out by itself, therefore we decide not to integrate any machine learning algorithm.

Starting from the simple case of two-part counterpoint, I created a python program with the library *music21*. According to its inventor Cuthbert and Ariza, " Music21 is an object-oriented toolkit for analyzing, searching, and transforming music in symbolic (score-based) forms" (2010). Using the implementation of musical concepts such as pitches, notes and key signatures in *music21*, it is very convenient to create the structure of a two-part counterpoint piece.

Algorithm structure

The idea of my algorithm is to filter the possible notes by all the rules in counterpoint, and then use a random process to choose a note according to its interval to the previous note. First of all, I created a variable space by limiting the note range by the part of the voice. For example, if I need to generate the note for soprano, the variable space is [C4, G5]. The next step is to filter out invalid intervals, such as parallel perfect intervals and dissonants on downbeats. The rules are extracted from the textbook of Review of Tonal theory, written by Laitz (2010).

The algorithm judges invalid intervals using three existing notes: the previous CF, the current CF, and the previously generated note. After this filtering, all the notes remain in the variable space should be "valid", but clearly some choices are "aesthetically" better than others. Therefore, the algorithm calculates the intervals towards the previously generated note. Smaller intervals have greater probability to be chosen as the current note.

Result

As a result, the algorithm is able to generate some pieces that resemble at least tonal theory students' works.

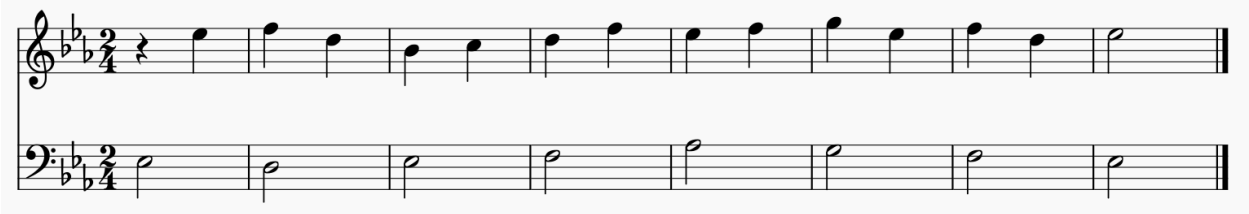


Figure 1: Two-part counterpoint generated by *counterpoint*. Bass is the given *canctus firmus*.

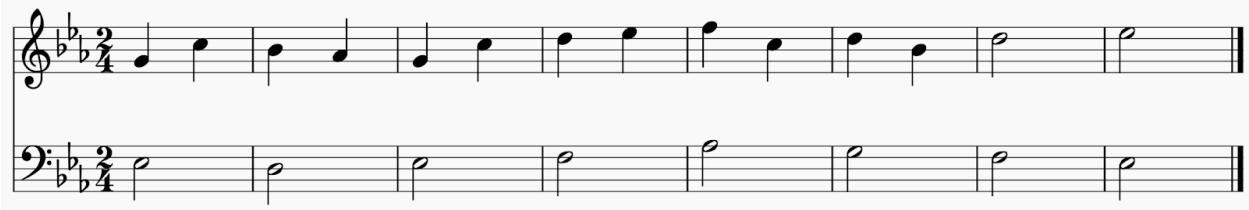


Figure 2: Two-part counterpoint written by myself. Bass is the same CF as *Figure 1*.

However, there are still some flaws in my algorithm. First of all, since the notes are randomly chosen, even though the big leaps have small possibilities, there is still a chance to see some leaps in the result. Secondly, an unobvious yet important flaw is that the beginning two

measures almost remain the same in every result. This is due to the random process tends to yield similar results at the beginning.

Future works

For our algorithm *counterpyt*, the future work is to combine the idea of Gibbs Sampling in our algorithm to further emulate the process of human composing. Also, a lot of work needs to be done if we want to create four-part counterpoint pieces, since the rules and regulations are much more complicated than two-part counterpoint.

There are still some issues worth discussing. Will the system work well with more complex forms other than counterpoint music, such as a phrase that includes a sixteenth note, or a song that contains a theme and chorus, and more instruments? Who owns the copyright, the author, the user, the programmer or the algorithm?

References

Bainter, A. (2019, June 5). Generating More of My Favorite Aphex Twin Track. Retrieved from https://medium.com/@metalex9/generating-more-of-my-favorite-aphex-twin-track-cde9b7ecda3a.

Bassel, G. W., Glaab, E., Marquez, J., Holdsworth, M. J., & Bacardit, J. (2011). Functional network construction in Arabidopsis using rule-based machine learning on large-scale data sets. *The Plant Cell*, *23*(9), 3101-3116.

Chris, G. (2015, April 29). Algorithmic Music - David Cope And EMI. Retrived from https://computerhistory.org/blog/algorithmic-music-david-cope-and-emi/

Cuthbert, M. S., & Ariza, C. (2010). music21: A toolkit for computer-aided musicology and symbolic music data.

Huang, C. Z. A., Cooijmans, T., Roberts, A., Courville, A., & Eck, D. (2019). Counterpoint by convolution. *arXiv preprint arXiv:1903.07227*.

Laitz, S. G., & Bartlette, C. A. (2010). Graduate Review of Tonal Theory: A Recasting of Common-practice Harmony, Form, and Counterpoint. Oxford University Press.

Moura, F. T. (2018, November 19). David Cope: A Lifetime Contribution to Artificial Intelligence and Music. Retrieved from https://musicstats.org/david-cope-a-lifetime-contribution-to-artificial-intelligence-and-music/

Wiggins, G. A. (2007). Computer models of musical creativity: A review of computer models of musical creativity by David Cope. Literary and Linguistic Computing, 23(1), 109-116.