Fall 2018

RNN monophonic sheet music generation with LilyPond

Seth Colbert-Pollack
Kenyon College, colbertpollacks@kenyon.edu

RNN monophonic sheet music generation with LilyPond

Seth Colbert-Pollack
colbertpollacks@kenyon.edu
AI for the Humanities, Fall 2018, Kenyon College

Abstract

Computers are well suited to tasks such as data categorization and labeling. More challenging is data "generation", a problem in which recurrent neural networks (RNNs) and more specifically long short-term memory networks (LSTMs) have made significant progress in the past few years. In this project, I train an RNN on a database of classical sheet music, and use it to generate new sheet music.

Recurrent Neural Networks

Simply **recurrent neural network** network or **RNN** is a system for generating some output based on some input which the network has been trained on. The inputs (in our case, text represented as numbers), shown on the left in the diagram below, are multiplied by some values, represented as arrows. These products are added together, giving us the values in the left hidden layer. The left hidden layer performs some calculations and passes the results to the right hidden layer. The right hidden layer then produces an output (in our case, a character) as well as giving the results of new calculations *back* to the first hidden layer. Then the cycle begins anew.

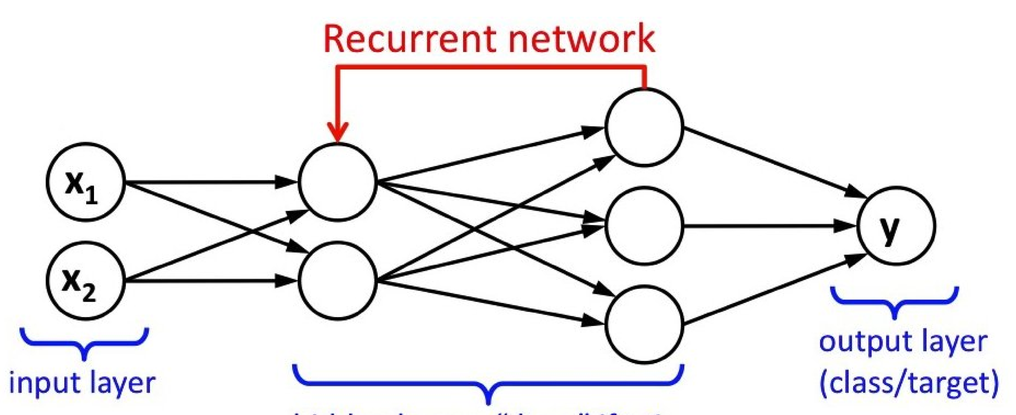

Figure: A recurrent neural network [1].

Data Collection

I trained my model on a series of LilyPond files. I was inspire by Andrej Karpathy's popular blog post [4], where he in part describes a RNN he trained on LaTeX files for a math textbook. **LilyPond** is a LaTeX-like file format for **music engraving**, i.e. the typesetting of sheet music. A LilyPond file is simply a text document, which, when compiled, results in a PDF of sheet music. Unlike an audio file, a LilyPond file gives explicit information on the key, time signature, dynamics, tempo, etc. of a song. For example, LilyPond distinguishes between a dotted quarter note and and eight note, which might sound the same in an audio file.

My training data came from the Mutopia Project [5], an open-source initiative with the goal of making sheet music of public domain musical works. Volunteers typeset old pieces of music in LilyPond. Using `BeautifulSoup`, a Python package, I crawled the Mutopia Project's website for piano solos from the Classical Period (c. 1730–1820), resulting 102 solos for a total of 1.12 MB of data.

```
\header{
  subtitle = "Minuet in F Major"
  composer = "Wolfgang Amadeus Mozart"
}
<<
  \new Staff \relative c' {
        \clef treble
        \key f \major
        \time 3/4
        f'8( a) c,4 c  |
        d8( f) bes,4 bes  |
        a8( c) f,4 e  |
        e2( f4)  |
  }

  \new Staff \relative g, {
        \clef bass
        \key f \major
        \time 3/4
        f2 a4  |
        bes2 bes4  |
        c2 c,4  |
        f c f,  |
  }
>>
```

Figure: The LilyPond file for an excerpt of a Mozart minuet, and the accompanying PDF output.

A complication: Monophony vs. Polyphony

Most, if not all, piano solos in my data set were written for two hands, i.e. there is a staff which shows what the right hand plays, and a staff which shows what the left hand plays. Simply put, these two series of notes are called (melodic) **voices**; music with a single voice is **monophonic**, music with multiple voices is **polyphonic**. The Mozart minuet above is polyphonic.

In Lilypond, melodic voices are written down separately, one after the other. This means all the right hand's music appears before all the left hand's music. Since our model can only look back 40 characters at a time, it cannot write the left and right hands "together". Instead of combining two unrelated melodies that the model produces, I decided to take single voices from the model's output and consider those alone, without accompaniment. Metaphorically, I am training on about 204 single voices instead of 102 two-voice pieces.

Training the model

I trained a model using the Python package `textgenrnn` using the method of Max Woolf [6]. After experimenting with some smaller models, my final model had 3 layers with 128 **long short term memory**, or **LSTM** cells in each layer, and was trained character-by-character, considering the previous 40 characters each time before predicting a new one. After training for 20 epochs, I had some moderately coherent results. The LilyPond code which the network produced was not anywhere near compilable, but it did contain fairly lucid passages which could be considered melodic voices. It would be fair to say the results looked like they came from a human who had just learned how to use LilyPond—that is, couldn't use it very well—but was very enthusiastic. I took these melodies and compiled them into sheet music on their own.

Results

I generated 80KB of data from the model. The resulting LilyPond code was recognizably LilyPond-like, but would require a lot of editing and cleaning up to compile.

```
        s2.*4 | s2.\< s8 | s2\dures bes8 s4. \stemDown r8 <aes aes>32. <bes des>8 <ges ces> <ges c ees> <ges
des g> |
              %
        }
        s2\sustainDown s8. |
                              s,2.\sustainDown |
                        s4 <e, bes'> <ges bes> |
                        s4 <des' f'> <des' ees'> |
                        s4 <des' f'>2 |
                        s4 <bes f'> <ees f'> |
                        s4 <des' ees'> <a c'> } \\ {ges4 des' } >> r4
 <aes c' ees''>4 <aes d' ees'> <aes c' ees'> r <aes des' ees'> <aes c' ees'>
 r4 <aes c' ees'> <aes c' ees'> <aes c' ees'> <aes c' ees'> <aes c' ees'> <aes c' ees'> <aes c' ees'>
 r <g des' e'> <f aes c'> <f aes des'> r r2
 \clef treble << { g'8 f' ees'' d'' c'' bes' a' g' fis'4 g'8 e'16 f' g' a' bes' a'8 g' f'8 e'16 f' e' d'4
c'8 e' d'16 c' b a c' a g f e16 d e f g a g f e d c b a g f e
 c8 c c' c b d' c' d c' b, c' d4 c' d' bes d' e g d' g bes d' g bes d' bes d' bes d' bes d' bes d' bes d'
bes ees' bes ees' bes d' bes d' e' f' a c' e' g' e' g' e' f' g' f' g' f' g' b g' c' g' b g' g' b g' c' g'
c' g' c' g' g' g' f' g' e' g' f' g' g' e' g' f' g' f' g' e' g' e' g' b g' c' g' g' g' b' g' c'' g' c'' g'
c'' g' c'' g' c'' g' c'' g' c'' g' c'' g' c'' g' c'' g' c'' g' c'' c'' g' c'' g' c'' g' c'' g' c'' g' c''
g' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c''
c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c'' c''
```

Figure: A sample of output from the model. Like most of the output, I did not believe this was worth trying to compile.

I chose excerpts from the 80KB output which I thought would require little human intervention to compile, and simplified them until they did compile. Below are three such excerpts.

Observations about the output:

- The passages I selected, which are better than a random selection from the output, don't sound particularly good, but they sound better than random notes played in secession.
- The model is good at writing individual notes, chords and rests.
- The model really hasn't learned what key signatures are. Sometimes it remembers to indicate one, but then doesn't write the rest of the passage in that key.
- It doesn't seem to understand measures or rhythm either. For example, in the second excerpt, it chose a 6/8 time signature but then used a lot of dotted quarters, which do not take advantage of the 6/8 beats. Meanwhile, the third excerpt, it has beamed eights notes strung together across measure bar lines, which is to be avoided.
- It know how to invoke more complicated complicated notations, such as fingering or triplets, but doesn't always do so appropriately. In Excerpt 1, it gives two fingers for a single note, and later writes a "triplet" with only two notes.
- Often the model will "lose steam" and write the name note or notes over and over. We see this in Excerpt 2, where the F natural is repeated over and over.

Criticism of RNN-generation for music

While doing research for the project, I found a number of critiques of RNN-based music generation. The following Hacker News comment is representative:

> *"I don't fully understand the fascination of [failed] music generation with some AI-neural-learning buzzword bingo technique that [is] kick-started by dumb-force-analysing a human made music corpus to achieve it."* —*iammyIP [3]*

What I find compelling about these types of neurals nets is that they are starting completely from nothing and still learning how to emulate the dataset which they are trained on. The model "knows" that songs have keys, time signatures, that there are chords, that it can add sharps or flats; many of the LilyPond commands it uses are ones that I, admittedly a beginner, don't even know myself.

Criticism continued

The commenter goes on:

> *"What... is much more interesting is to generate new music that cannot be composed by a human, and cannot be played by a human. That's playing to the strength of the machines. Sonification of large datasets, sonification of function behaviour. Sonification of the binary world, that's so different to ours. This is much more interesting than the 10th failed emulation of a simple folk song."* —*iammyIP [3]*

I find this critique valid. I envision the tools available, including the one I used in this project, to be not a *replacement for*, but a *supplement to*, human creativity. Frankly, the music my model is generating is not really valuable enough to inspire better human composition. But that does not mean the project was a failure.

Future Work

There are many possibilities to explore which would likely improve the model's music generation.

- **Cleaner data.** The LilyPond files which the model was trained on were written by many different volunteers, who all took different approaches to rendering the music in LilyPond, including using different names of notes. If we trained on a subset of the files which were rendered "similarly", it might improve the model.
- **Additional datasets.** We could compare outputs of the model trained on the Mutopia Project's collections of piano solos from the Baroque Period (143 pieces) and the Romantic Period (181 pieces).
- **More experimentation.** By changing the number of layers in the network, then number of neurons per layer, whether the model is character- or word-based, the length of training in epochs, or other parameters, we are likely to find a better model. This does require more computation time.
- **Skip the sheet-music approach.** There exist programs which can turn a MIDI audio file into an appropriate text-based vector or array of vectors [2]. This would give uniform, clean training data with the added benefit of allowing simultaneous processing of different musical voices, which is impossible in my model's current state.

Conclusion

The biggest takeaways from this project were not regarding the ability of AI to write music, but rather regarding the process of using software tools to gather appropriate training data. As is typical for machine learning projects, the collection and preparation of the data was the most difficult part of the project. This is a reflection not only of my having to learn how to use `BeautifulSoup`, but also the relative ease of use of RNN training programs which are freely available online. Ultimately, I was impressed by the network's ability to meet the challenge of music engraving.

References

[1] Leonardo Araujo dos Santos, *Recurrent neural networks*, from *Artificial Intelligence* Gitbook, `https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/recurrent_neural_networks.html` (2017).

[2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent, *Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription*, Proceedings of the 29th International Conference on Machine Learning (2012).

[3] iammyIP, *As usual the result sounds awfully unstructured...*, Hacker News, `https://news.ycombinator.com/item?id=11406035` (April 1, 2018).

[4] Andrej Karpathy, *The unreasonable effectiveness of recurrent neural networks*, `http://karpathy.github.io/2015/05/21/rnn-effectiveness` (May 21, 2015).

[5] The Mutopia Project, `http://www.mutopiaproject.org/` (2018).

[6] Max Woolf, *How to quickly train a text-generating neural network for free*, `https://minimaxir.com/2018/05/text-neural-networks/` (May 18, 2018).

Acknowledgements

Thanks to Professors Katherine Elkins and Jon Chun for teaching the course and showing us what is possible with today's machine learning. Thanks to Michael Lahanas, Chris Raffa, and Jacob Bozeman for their camaraderie and friendship.