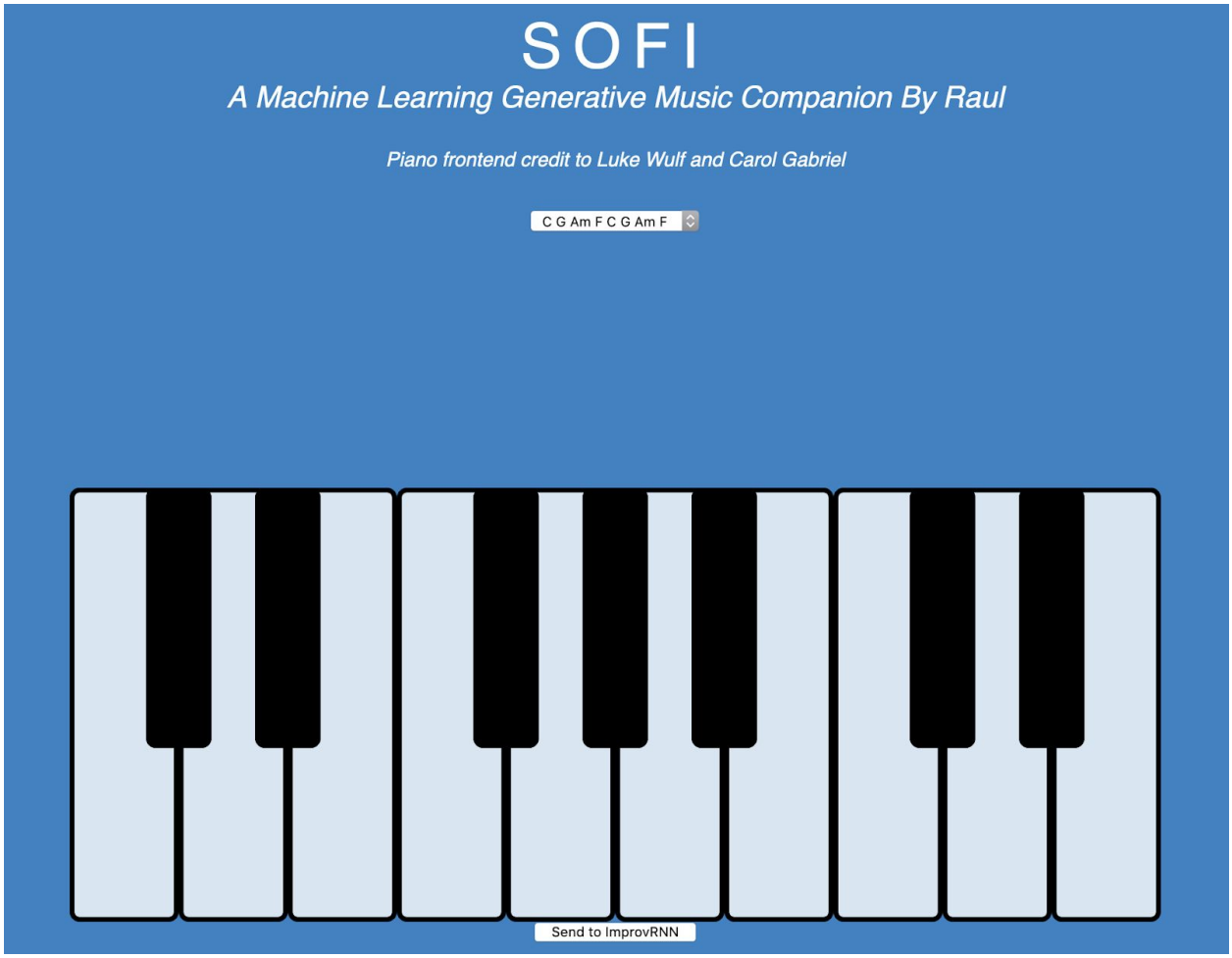
Machine Learning for the Arts
UCSD SPRING 2019
FINAL PROJECT

SOFI

A Generative Music Companion



Raul Pegan

DESCRIPTION

Concept:

For this project I decided to explore the concept I found most interesting throughout this course: is an Artificial Intelligence a creator, co-creator, or tool? As a result, I chose to create some sort of project which would overlap these three points of view, and the best way of testing that is through dynamic art creation. Naturally, music has no static structure, so it is the best fit for this approach. I would like to create an interactive art piece and let the users determine for themselves whether the piece is a creator, tool, or co-creator.

As a fan of product engineering, I also wanted to create a project which was constrained to good product design practices. I wanted my piece to not only explore the landscape I previously described, but also be easy to use for all potential users. Ideally, this would make the project more accessible to anyone and consequently provide a larger amount of interesting results. So I went ahead and "app-ified" the project, creating a simple yet useful interface for the user.

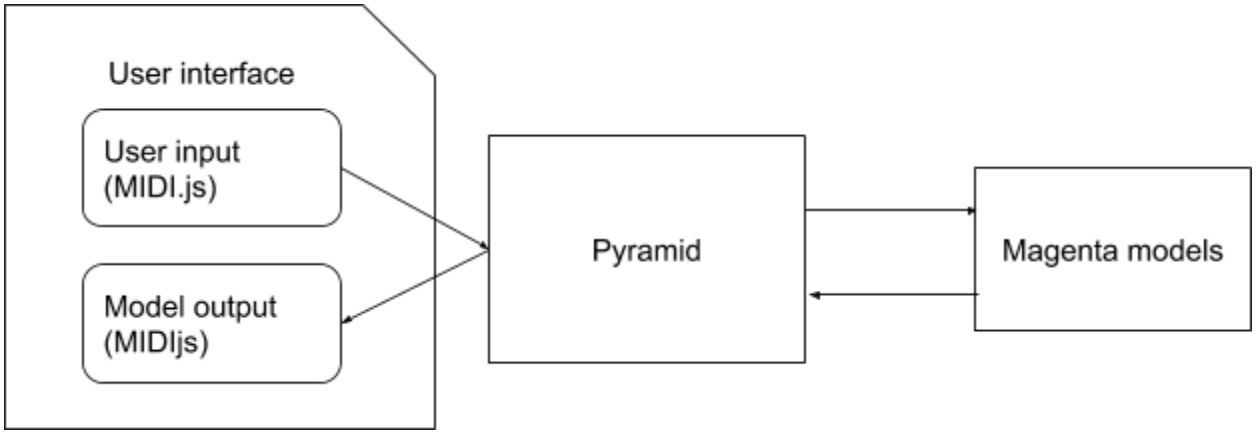
The final ideated version of this project was a simple keyboard interface where the user could input any sort of melody and the machine would automatically read that input and produce a cohesive response back to the user. The user could then take that response and build on top of it, *creating a back and forth conversation between man and machine through the use of music.*

Technique:

This project consisted of two main parts: the interface and the model. For the model I used Google's impressive assortment of music models under the Magenta project [1]. Out of all the magenta models, three fit the vision of the project: melody_rnn [2], performance_rnn [3], and improv_rnn [4]. These models all work in a very similar fashion, they take in the input as a .mid file or sequence of MIDI numbers, and produce an output using that input as a primer. Additionally, improv_rnn also takes in a chord progression as one of the inputs. I used the pretrained weights for these networks.

For the interface, I used Python Pyramid to host the web app, which is mainly a javascript front end that allows the user to play music using either the on-screen keyboard or a MIDI instrument connected via USB. The JS libraries used were MIDI.js [5] to interface with the MIDI controller and playback real time audio, MIDIjs [6] to play the model's MIDI output (**Note:** these two libraries are different, as confusing as it sounds. The first handles the user's input while the second one handles the playback of the model's output), and Tonal.js to give user's real time display of their notes on the app interface.

The user's input would be gathered using the JS libraries, this input would be sent to the magenta models using the python backend, and then the output would be played back through the app front end using JS again. The following diagram illustrates this:



Process:

Developing this product took many different steps. First and foremost, I experimented with the different magenta models in order to determine which one was best and how to optimally use them. In my experience, improv_rnn had the best results, though it is the more “complex” of the three. This is due to the extra layer of control where the user can provide a chord progression and thus customize the output a bit more. Once that was working, I needed to make sure that it could communicate with a web app so that the user had some sort of simple interface. I chose Pyramid for this since it is lightweight and Python based, so it would work well with Magenta. Once the web-app infrastructure was in place, the interface was designed to look good and provide a seamless user experience.

The main challenge for this project was two-fold: technical and artistic. I’ll develop the technical issues first. Due to datahub’s limitations, I was not able to develop a simple solution to host the app on their servers, so I was constrained to running the project locally, on a 2012 low-end MacBook Pro. As a result, the project runs very slowly and there is quite a bit of latency between the input and output of the project. This could easily be overcome with superior hardware, but I lacked those resources. I also tried using models other than Magenta, but they lacked the Artificial Intelligence approach to the project. Markov Chains work well with music generation but it is not really artificial intelligence, and other models out there don’t provide a complex and developed outputs when compared to magenta. So it was clear that magenta was the best call. Lastly, a big technical challenge for me was dealing with MIDI as a whole, since it is a technology I had previously had no experience in, and it is a complex system to work with (especially from within javascript!)

As for the musical challenges, these were the biggest limiting factor for me. Prior to this project, my most involved experience with anything music related was downloading Usher’s “Yeah!” from Limewire approximately 12 years ago. I have absolutely no knowledge of musical theory and I didn’t realize how important it would be for this project. In order to understand the output so I could best improve the input, I needed some musical background and I certainly jumped

into the deep end for that one. I did receive plenty of help from a few musician friends: Habib, an old roommate who is heavily involved in music theory; and Juan Cristobal, my uncle, who is a conductor for a youth orchestra. They certainly helped me develop and understand the musical theory behind the outputs as well as provide me with useful feedback as to how to best design and optimize the interface for non-techy musicians.

Result:

The end result was somewhat close to my vision: the users input their music and the model analyzes that and spits out a somewhat cohesive output. It does however take a very long time, so it is not as dynamic as I would have liked it to be. The visualization of the end design can be found on the cover of this document, and videos of the result can be found in the link for the 'Results' section.

Reflection:

As previously discussed, the chosen final model was improv_rnn as it had (in my opinion) the best results. This could be modified however. I wouldn't say I am particularly satisfied with the end result at all. The product is severely hindered by the latency and the output quality is still not great in my opinion. Due to the steep learning curve for basic music theory, I was not able to tweak the output to my liking, even though I attempted to experiment with that (with chord progressions, etc). Thankfully, Juan and Habib gave me great insight into this world and helped me adapt as quickly as possible. I am very satisfied with the User Interface / Product Design aspect of this project, which makes sense as I am significantly more experienced in this field. If I could keep working on this project, I would choose to spend even more time understanding music theory as well as optimizing the model for a quicker output.

REFERENCE:

[1] Google AI
https://magenta.tensorflow.org/

[2] Melody RNN
https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn

[3] Performance RN
https://github.com/tensorflow/magenta/tree/master/magenta/models/performance_rnn

[4] Improv RNN
https://github.com/tensorflow/magenta/tree/master/magenta/models/improv_rnn

[5] MIDI.js
https://github.com/mudcube/MIDI.js/

[6] MIDIjs
https://www.midijs.net/

[7] Tonal JS
https://github.com/tonaljs/tonal

CODE: (This github project link)

RESULT:
Three files:

- Habib.mov - My friend habib going back and forth with the product
- Juan.mov - My uncle Juan interacting with the product
- UI_closeup.mov - Me interacting with the product, I am not a musician so the generated audio is not good, but this serves as a closer look at the UI

https://drive.google.com/open?id=1xVCapmBD75m5soVq5m3X85zKQ5PnchDl