CS221 Progress Report: Robot Music

Han Lin Aung, Justin Xu, Sicheng Zeng

November 17, 2017

1 Introduction

Our project is a music generator, specifically focusing on hip hop and rap beats. Instead of taking on the task of generating whole songs, our goal is to create an AI that will take in famous beats such as "The Real Slim Shady", parse them into different voices and beat lines, and then use this collection of voices to create new, equally as catchy beats.

We used .midi files as our inputs and outputs, and we restricted our scope to short, repeatable beats rather than lyrical melodies. A successful generator would be one that fits the different beat lines together well in a way that produces a complete, catchy beat with varied voice lines.

2 Progress Report Followup

Instead of using ourselves as an oracle, we will survey random people, having them listen to a set of music, of which half is music we generated, and half is original midi music files. Without knowing whether the music was original or machine generated, the people will be prompted to rate the music from 1 to 10 overall, as well as on several more specific questions, such as whether or not they felt the music was cohesive, or if they could follow the beat. We plan on using Amazon Mechanical Turk to mass survey people, or if that isn't an option, door-to-door surveying or online surveying.

3 Model and Algorithm 1: RNN-Backtracking pipeline

Our first attempted model began with many midi files of drum beats, which were parsed into instrument lines. All of the resulting instrument lines were then passed into a recurrent neural network that would produce similar instrument line outputs. Finally, we would backtrack and search the combinations of instrument lines in order to find the best resulting songs.

Midi files are naturally split into different parts, and can be easily separated into their separate voices using software such as GarageBand, MuseScore, or through coding methods such as methods from the python library Music21. We originally started with a diverse selection of songs, but the results of the training on the RNN were noisy and too diverse in note types,

so we changed to only using drum beats. We also edited the midi files to always run at 120bpm, with the thought process being that beats at the same tempo would mesh better together. To combine the tracks together, we extracted a feature vector from each midi file that contains information about the number of each note type, the musical key and mode, the number of each note length, and the overall range, as a fairly basic list of musical features. The weights on the feature vector would be trained on the dataset created by listing our collection of real drum beats as positive samples, and random combined tracks as negative samples. However, in order to approach/find the combination of tracks with the best weighting, we would have to check each possible combination of tracks, and we concluded that although this was possible by limiting the total number of tracks in each combination to a small number, it would take too long to be worthwhile. The combination of this problem with searching and the chaotic, non-single-line output of the RNN generation made us reconsider our first model, and we eventually arrived at our second model.

4 Model 2: Splice

Our next model utilizes the following beat generator website as influence:
https://splice.com/sounds/beatmaker. Splice has broken a beat down into an 8 instrument 4 measure chunk which is represented by a 8 by 32 grid of boxes that can be turned off or on, where being on represents that instrument being played at that time in the beat and being off represents the absence of that instrument at that time. Using this model, we plan to make an AI that automatically generates a new beat given training samples that we will scrape from other people's creations on the website.

The search space is quite large, as there are 256 boxes that can be turned on or off, making there 2^{256} possible unique beats. We wan to combine the use of music theory, randomness, and a probabilistic model to approximate a new creation.

Since, with a random model, there is always a chance for model to spit out something that sounds decent, we therefore want to take an average evaluation so that we can get a good representation of how well the model is doing that is less affected by random chance. While testing, we will probably individually listen and all give ratings to the beats we get, and then take the average over many iterations, and then tweak models to see if our average case will become better.

Since this is still subjective, its possible that different models will be deemed more successful than other ones, so we will probably build multiple models and present the strengths and weaknesses of each.

5 Algorithm

Our problem will be represented as a factor graph with the assignments to the variables. Each variable will represent one space on the grid, where each point on the grid will have either 0 or 1 depending on if the space on the grid is turned on or not.

Factors for this problem will consider the dependencies of the variables with each other. We can think about also including additional variables which measure certain blocks of information, such as maybe a variable for the assignment of each measure just for overhead reasons.

Some factors we want to consider are how different instruments depend on each other. Given an assignment of variables of one instrument, it usually probably wouldn't make sense, for instance for another instrument to have the exact same beat pattern.

Another factor that we are thinking about is how eccentric a beat in one bar is. We might want a repeated and stable beat for one bar that provides the backbone of the music, but for some other instruments, we might want to sprinkle in some more variety to make it sound interesting as a whole.

An additional idea is to dynamically build up this array with some randomness in the generation, as this allows there to be variety in the beats that we create. There's a balance between variety and expectation of beat patterns when it comes to this, so we think that this model will do justice to that intention.

So, because of this we are also thinking about also including a second factor graph with the probabilities that a certain grid blck is assigned to the on position. Our variables will be the individual boxes that can be assigned a discretized probability from $[0, 1]$ with .05 increments, and the factors will be a set of heuristics we will set up based on the features of the data we've collected.

The first algorithm we are doing is naive bayes, which will simply find the probabilities by assuming all the blocks are independent from each other, by taking the weighted average of all the training examples, and assigning the respective probability to that square. Then, we will simulate a generation based on those probabilities and listen to it.

This algorithm is very reasonable, as using data from the distributions of other samples will give us a reasonable estimation of probabilities if this is the optimal model.

Upon this, we can use the first factor graph to evaluate the weighting of this assignment fares, and make the necessary adjustments to make it an optimal.

Other directions we plan on exploring include more complex algorithms like particle filtering or Gibbs sampling, which will look at adjacent nodes or a randomized search, and also inherently utilizes probability distributions. The problem is that there doesn't exist an "optimal" assignment for the grid, so our evaluation function in that respect doesn't exist.

Some machine learning can also be used to automatically create the weights for the factors, utilizing our training data to learn how rhythms naturally interact with each other.

Another approach we can have is to have build it some manual heuristics based on properties of music theory that we might know. These will include maybe how rhythms interact with

each other.

6 Results

We scraped data from 20 websites, and created a naive bayes distribution. We then ran our algorithm a few times, generating beats for splice based that probaiblity distribution.

Our initial results were promising, but also showed that there was room for improvement. Out of 4 created beats, all sounded musically tolerable, with some irregularities introduced in some of the samples.

From our self evaluations we gave the following ratings to the music:

Sample#	Sicheng	Han	Justin	Average
1	7	7	8	7.7
2	4	6	5	5
3	4	4	4	4
4	5	6	5	5.3

One of the music samples that we created is in the link:
https://splice.com/sounds/beatmaker/0d425a82cad3