Choral Music Generation Using Word2Vec and LSTM Network

Günay Eser
Boğaziçi University
Computational Science and Engineering

Abstract

Music generation has become an area that is researched with a significant pace only recently. Generating music has many challenging points to overcome and it is a long way to obtain a piece that is so close to a human-composed one. Music generally contains several tracks, each of which has different note pattern, style, instrumental characteristics and emotion. When combined, these tracks create a harmonic and tuneful music. So far, papers on music generation were focused on generating instrumental music. In this paper, we propose a new type of music generation, Choral music. Choral music is constituted from different vocal groups like Soprano, Alto, Tenor and Bass. Sometimes these groups get separated into smaller groups like (Soprano 1 and Soprano 2) and each individual group has a particular track in the song to sing. Target of this paper's approach is to compose tracks for 4 singer groups (Soprano, Alto, Tenor, Bass) that is pleasant to hear when combined and transformed into a choir song. We use word2vec word embbedding for vectorize notes and chords, then use LSTM Network for generate new trakcs. Our approach gives better results in generation process when the chords are considered as words instead of notes.

Introduction

A Choir is a musical ensemble of singers of different note ranges and colours. Different note ranges and genders gives different emotions to the music that is performed.Choirs mostly consist of four groups intended to sing four part in harmony. Number of parts can be higher as well. Main focus on this paper is SATB choral music, which is a very common form. (Mikolov et al. 2013)

There has been many process on generating videos, text, music and images recently using GAN's (Mikolov et al. 2013) and LSTM's (Hochreiter and Schmidhuber 1997). However, we experimented choral music generation with a different approach, that combines natural language processing methods with music generation in this paper. Choral music consist of different tracks and each one has a different pattern, emotion and note range. When each singer group performs a note, there occurs a chord.

Chords are the key sources of choral music and resonates to audience's ears which is pleasant to hear when notes are performed with very small frequency errors by singers. For example, the perfect A note should be sounded with 440Hz to resonate with other notes. In every different time tick on a song, a different chords might sound as every singer sings a different note with different volume. Generating choral music can be considered as a multitrack music generation.

However, generating vocal tracks is quite different than it is for instruments and it is much more limited in terms of some physical constraints of a human vocal capacity. Main difference between choir music and instrumental music is that, tracks represent notes for each singer group instead of an instrument like Bass Guitar or Acoustic Piano. Vocal chords of a human is more likely to have a smaller note range than a instrument. Another difference between a vocal track and a instrument track is that a singer cannot sing a chord by herself/himself while instrument can sound chords since they can be playable multiple notes at the same time.

Our approach was at first to consider **each note as a word** and **each chord as a sentence** so that every song can be considered as a paragraph that consist multiple sentences.Then with enough number of songs, there would be a considerably large corpus.

- We transformed multitrack midi formatted songs into piano rolls.
- Then split each second into a certain duration f and build note data instances for every duration f
- For a given sentence lenght s we grouped instances into sentences. And built the corpus.
- We vectorized notes with Word2Vec (Mikolov et al. 2013) approach for the given corpus.
- Trained an LSTM Network with the calculated embeddings.
- Generate the chords one by one with the given parameters.

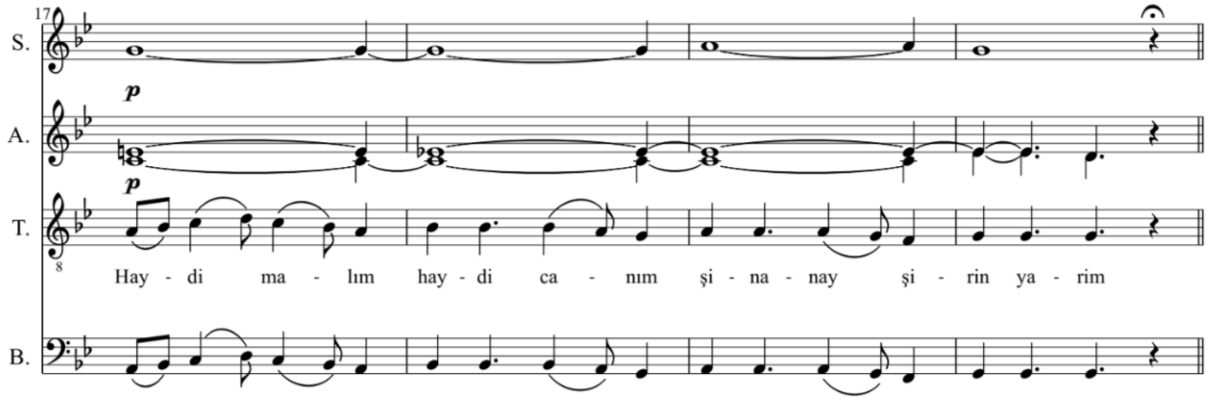

Figure 1: Choir sheet part example

Related Work

First music generation by computer is introduced by Hiller Jr Isaacson (1957) using Markov chains. (Moorer 1972) and (Pressing, 1988) also worked on some rule based music generation. There were also some approaches using GAN's MidiNet (Yang, Chou, and Yang 2017) and MuseGAN (Dong et al. 2018) as well. Music Transformer (Huang et al. 2018) and MuseNet (Payne, 2019) used Transformers to auto-regressively predict MIDI notes.

RNN's are mostly used in this approached. Monophonic music generation like (Sturm et al. 2016) is done. (Hadjeres and Pachet 2016) DeepBach approach was the first to work on choral music generation that aimed to compose Bach style choral tracks. They used 2 RNN's and one 2 regular neural network and merge results into a final softmax function. They only used Bach compositions in order to avoid their time signature problems since Bach used only certain time signatures in his musical sheets and that helped them in the training process. For generation part, they used Psuedo-Gibbs sampling procedure and that helped them define some constraints and put into the model.

Method

Data Representation

We use MIDI files for extracting musical data and build corpus. MIDI files can be built in different ways, for example there can be percussions, extra solo track and, SATB pattern can be different in each MIDI file. After extracting each singer party's tracks, we take piano roll representation of each track.

Implementation

A Piano Roll, $R_{p_x t}$ is a matrix representation of a track where $p \in [0, 127]$ is the number representation of the pitch (C4 = 60 for example), and t is the time instance where it is decided by

$$1(second)/f$$

where f is the split size. For a song that is 1 min 30 seconds, if $f = 6.66$ then, $t \in [0, 600]$. Therefore, $R_{62_x 256} = 80$ represents the note D4 with velocity 80 on the time 256. See Fig 2. One thing to consider here when deciding f, every song has different tempo, therefore notes with same duration signs from different songs will have a different duration in seconds.

After we obtain piano roll of each track we transpose the matrix and merge them. Then we add track numbers to specify the singer party and the velocity information to the data, and get a final dataframe which we call **song-dataframe**. See Fig 3. Adding singer grou information into the data is necessary due to avoid generating impossibly high pitch notes for bass and alto parties to sing. Velocity information is also added to generate more colorful and human-composed-like results.

Each row in these final song-dataframes, represents a chord (or word in terms of language processing) sounded in the specified time instance. Then we group every l rows into one and consider them as a sentence. We tried different l's in order to obtain different corpora. The parameter l should be decided with considering f value. Small l value sometimes corresponds to a very small part of a second for big f values.

At the end of each song, we add extra P more rows for padding, that will separate every song in the corpus. Our main goal here is to teach model to realize different songs in the corpus, so that it wont tend to use notes from multiple song's scale. However, model failed to separate each song's musical scale and generated songs ended up out of scale.

Once all the song-dataframes are at their final shape after padding, we concatenate them and obtain a suitable corpus for vectorizing process.

We use word2vec (Mikolov et al. 2013) embedding to vectorize each chord. We tried different window sizes and embedding sizes, which we will discuss in the experiments section. For vectorizing, we used python's gensim module.

	0	1	2	3	4	5	6	7	8	9	...	14269	14270	14271	14272	14273	14274	14275	14276	14277	14278
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 2: Sparse piano roll matrix sample, song duration is $f/14279$ seconds.

Before putting word embeddings into the training process, we observed the behavior of cosine distance of some chords for different window sizes, embedding sizes and sentence length l. We found out that smaller bigger window size and sentence length results in similarity between chords that is unnaturally too big.

We put our embedded corpus into our Network architecture for training. We preferred Keras for implementing our network model. Our network architecture has:

- An Embedding layer
- An LSTM (Hochreiter and Schmidhuber 1997) layer with sigmoid recurrent activation function and $tanh$ activation function.
- Dropout with 0.2 ratio
- Another dense layer
- Softmax function at the end as activation function

For the generation part we used random sampling with different initial chords. We observed that initial chord do not change the quality of the generated song significantly. Our choice of initial chords only included chords that are started and ended by groups together. (There is no group sustains the note longer than the chord.)

Experiment

Big amount of our experiment time was on preparing proper data dealing with midi formatted files. So we begin with tha prepration of our dataset.

Dataset

Our dataset is from 2019 ITU A Capella (from Istanbul Technical University choir) repertoire and all the songs are midi formatted. The main reason we chose this choir is they use diversified songs from all over the world. That helped our dataset not to be limited with certain composers and their styles. Here is the origin list of the 14 songs that we used to create our dataset.

- Ajde Jano - Serbian Folk Song
- Amavolovolo - African Folk Song
- Ben Annemi İsterim - Turkish Folk Song
- Carol of the Bells - Mykola Leontovych
- Come and See the Baby - Ruth Morris Gray
- Dağlarda Kar Sesi Var - Ezgi'nin Günlüğü
- Dom Dom Kurşunu - İbrahim Tatlıses
- Mecburen - MFÖ
- Tourdion - Pierre Attaingnant
- South African Trilogy - African Folk Song
- Tumekuja Kuimba - Lynn Zettlemoyer
- Üç Kız Bir Ana - Turkish Folk Song
- Yonder Come Day - Paul John Rudoi

Data Processing

Some of the midi files have other tracks than main 4 singer groups like percussions, solo, etc. Tracks order were also different for each midi file and needed to be analyzed manually before extracting the piano roll. For the process of reading tracks and forming the piano roll matrix, we used pretty-midi module of Python.

We also studied some other detailed midi related modules of python like mido and python-midi, however they needed much more focus to be used properly, so we discard those choices from the list. We keep them in mind for a possible future work.

Generation and Evaluation

Singability

For evaluation of the generated music, our first choice is to check if the tracks are singable or not. A music to be choral, it must be suitable to human chords for singing.

Harmonic Change Detection

Beside this human evaluation, we also considered using method introduced by (Harte, Sandler, and Gasser 2006), Detecting Harmonic Change In Musical Audio.This method uses audio formatted data as input thus we transformed each midi file our method has generated into .wav format.

Result	Approach	f	l	P	Window Size	Vector Size	Epoch	Batch Size	Singability	Out of Scale
midi.mid	Notes as words.	3.33	10	None	10	50	20	2	**Impossible**	**Yes**
midi-2.mid	Chords as words.	3.33	10	None	5	100	20	2	**Impossible**	**Yes**
midi-3.mid	Chords as words.	=3.33	6	None	5	100	40	2	**Very Difficult***	**Yes**
midi-4.mid	Chords as words.	3.33	12	None	15	30	20	4	**Impossible***	**Yes**
midi-5.mid*	Chords as words.	3.33	3	10	20	100	20	48	**Impossible**	**Yes**
midi-6.mid	Chords as words.	6.66	10	10	5	100	20	2	**Impossible**	**Yes**
midi-7.mid	Chords as words.	6.66	10	5	10	50	5	64	**Very Difficult***	**Yes**
midi-8.mid	Chords as words.	6.66	10	5	10	70	5	128	**Impossible**	**Yes**
midi-9.mid	Chords as words.	6.66	12	5	10	100	10	32	**Singable***	**Yes**
midi-10.mid	Chords as words.	6.66	10	5	10	70	5	128	**Impossible**	**Yes**

Table 1 : (* : Singability can be better if the tempo slowed down.) Considering notes as words generated very bad results as seen in the table. Therefore we did not focus on that approach for a long time.

Their approach is to find a sequence for six dimensional tonal centroid vector ζ in the 6D space of Harmonic Network or Tonnetz (Cohn 1998) And then using Harmonic Change Detection Function they calculated the overall rate of change of the smoothed tonal centroid signal. We used random sampling for the generation process with constant 0.7 temprature value.

Results

Unfortunately, all of the generated music was far from human-composed-like. They lack both emotion and human friendly tempo. Harmonic Change is unnecessary since all the generated songs are out of scale. We also find that frequency parameter should be tuned properly for every type of tempo separately since all the songs in the training set have different tempos. Adding Padding at the end of each song-dataframe gave better results in generation. The iterations we applied padding gave us significantly different results on the positive direction. Midi files on the table are ordered chronologically of our experiments. First 4 midis are very robotic and fast, we figured that the model could not catch note durations on these iterations.

Also the midi-5.mid file has the closest emotion to human-composed-like music (or farthest from roboticness) among other, yet its tracks are not singable by any human since it has big gaps between some consecutive notes.

Conclusion and Future Work

This papers approach was to be a pioneer for analysing music with language vectorization methods. It is possible to consider **music as a language** and apply natural language processing methods on it. Although we failed to generate state-of-the art results, we observed that LSTM's and word2vec vectorization method can be used to generate choral music.

Future Work

Our time construction for training data is probably problemati since some of the generated song are insingably too fast. LSTM network could not catch the scales in songs. Another style of construction of the training data needed to be implemented for scale accuracy. Generated songs sound very robotic and nearly single rhythmed. Lacks emotion in another words.

- Another embedding method like glove, fasttext might be used for vectorization.
- A larger dataset can be used for training.
- Each track can be generated separately instead of generating 4 singer chords simultaneously.
- More musical rules notations like, modes, measures of bars and scale can be represented in the training data as well. (We only used, pitch, velocity and $1/f$ time signatures.)

R. Cohn. Introduction to Neo-Riemannian Theory: A Survey and a Historical Perspective. The Journal of Music Theory, 42(2), 1998.

References

Dong, H.; Hsiao, W.; Yang, L.; and Yang, Y. 2018. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In McIlraith, S. A., and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 34–41. AAAI Press.

Hadjeres, G., and Pachet, F. 2016. Deepbach: a steerable model for bach chorales generation. *CoRR* abs/1612.01010.

Harte, C.; Sandler, M.; and Gasser, M. 2006. Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, AMCMM '06, 21–26. New York, NY, USA: Association for Computing Machinery.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Huang, C. A.; Vaswani, A.; Uszkoreit, J.; Shazeer, N.; Hawthorne, C.; Dai, A. M.; Hoffman, M. D.; and Eck, D. 2018. An improved relative self-attention mechanism for transformer with application to music generation. *CoRR* abs/1809.04281.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space.

Moorer, J. A. 1972. Music and computer composition. *Commun. ACM* 15(2):104–113.

Sturm, B. L.; Santos, J. F.; Ben-Tal, O.; and Korshunova, I. 2016. Music transcription modelling and composition using deep learning. *CoRR* abs/1604.08723.

Yang, L.; Chou, S.; and Yang, Y. 2017. Midinet: A convolutional generative adversarial network for symbolic-domain music generation using 1d and 2d conditions. *CoRR* abs/1703.10847.

Pressing, J. (1988). Improvisation: Methods and models. In J. A. Sloboda (Ed.), Generative processes in music: The psychology of performance, improvisation, and composition (p. 129–178). Clarendon Press/Oxford University Press.

Payne, Christine. "MuseNet." OpenAI, 25 Apr. 2019, openai.com/blog/musenet

R. Cohn. Introduction to Neo-Riemannian Theory: A Survey and a Historical Perspective. The Journal of Music Theory, 42(2), 1998.