Chord Progression Generation in the style of Studio Ghilbi film music

Ling Qi
Vassar College
lqi@vassar.edu

Jack Rogers
Vassar College
jrogers@vassar.edu

Gabor Fu Ptacek
Vassar College
gptacek@vassar.edu

Abstract

This project aims to generate chord progressions in the style of Studio Ghibli film music using different computational models, such as n-gram model, hidden markov model (HMM), and recurrent neural network (RNN). With a corpus of around 60 MIDI files for Ghibli music, we run experiments like varying n for n-gram and generating different lengths of chord sequences. The generated outputs are evaluated in terms of their similarity to Studio Ghibli soundtracks through longest common chord subsequence and same root sequence number. All three models generate intriguing results that more or less model the style. However, we do acknowledge that future works should aim to include a more robust corpus with annotated labels and more evaluation metrics to obtain more conclusive results.

1 Introduction

1.1 Studio Ghibli

Studio Ghibli is famous for its Japanese animation films like *My Neighbour Totoro* (1988), *Spirited Away* (2001), and *Howl's Moving Castle* (2004). Composer Joe Hisaishi has produced award-winning soundtracks for most Ghibli films. Although diverse in tone and theme, the soundtracks contain recurring musical features that heighten a whimsical, nostalgic, and pure feeling, which aligns very well with the narrative of Ghibli films. As huge fans of Studio Ghibli film music, we want to explore the possibility of generating its characteristic chord progressions computationally.

1.2 Chord and Chord Progression

To understand this study it is not necessary to have taken advanced music theory courses, but some fundamental knowledge of musical chords and chord progressions may be beneficial. Put simply, a chord is a combination of three or more pitches.[1] The pitches that make up a chord give it a particular sound or feeling: some chords are inherently pleasing to the ear and can be associated with happy, pure, or uplifting emotions, while others contain more dissonance and may sound troubling, mysterious, or unpleasant. Chord progressions are sequences of chords that contribute to a piece's overarching structure.[2] Deciding what chords/chord progressions are appropriate for a piece is up to the composer, and this is what makes up the focus of our study. We plan to analyze the chord progressions that exist in Joe Hisaishi's scores and extract his style of composition in order to generate similar chord progressions.

1.3 Music and NLP Models

To conduct this study, we borrow techniques from the field of Natural Language Processing (NLP). While music is not (technically speaking) a language, other studies have demonstrated that computational models from NLP are effective in processing musical data and language data alike. Our models will be trained on sequences of tokens that represent chords. Because our goal is chord generation, we will use models that predict the next item in a sequence of chords. We plan to look at three computational models: an n-gram model, a Hidden Markov model (HMM), and a Recurrent Neural Network (RNN) model. N-gram models count the occurrences of specific sequences of tokens to estimate the next item. HMM may help to predict the next chord based on unobservable events, such as the harmonic function of a given chord. RNNs sequentially build a prediction based on the results of previous inputs. The results of each model will

[1] Definition of a chord found at: https://www.musictheory.net/lessons/40

[2] https://www.musicgateway.com/blog/how-to/chord-music-theory

be evaluated using appropriate techniques also borrowed from NLP.

2 Related Works

2.1 Music Generation

Several researchers have explored the possibility of computational music generation through techniques commonly used in NLP. Ponsford et al. (1999) investigated the use of probabilistic techniques from NLP on musical data. They used N-grams and Hidden Markov models to learn about significant characteristics of the music and confirmed that generating and inferring harmonic frameworks was possible with these characteristics.

A subset of previous research focuses on melody generation specifically, especially using recurrent neural networks (RNNs). Wu et al. (2019) demonstrates how hierarchical RNNs can produce long-term structured melodies that are rated well by human evaluators. This study demonstrated the importance of long short-term memory (LSTM) models in order to "capture the global music structure and improve the quality of the generated music". However, Madaghiele et al. (2021) used similar methods but instead with a transformer model to achieve automatically generated jazz improvisation. They concluded that models using LSTMs were in general unable to capture the relationship between chords and melody. Since our focus is not to generate melody but instead only chords, we may find that an LSTM will suffice.

2.2 Chord Generation

An important characteristic of film music from Studio Ghibli is the use of extended chords, so we decided to focus our research on chord and harmony generation. This is reflected in the remainder of our related works, which primarily focus on chord generation. In 2009, Anders and Miranda (2009) explored the creation of a rule-based computational model that relied on music theory knowledge. In 2010, Eigenfeldt and Pasquier (2010) presented a method for harmonic progression generation using a variable-order Markov model, which generates the prediction based not only on the current state but also variable numbers of previous states. Later in 2012, they published another paper in collaboration with Evon Khor and Adam Burnett (Burnett et al., 2012). It evaluated their Markov model with human participants who were instructed to determine if a musical sequence was human or computer composed. More recently, Shukla and Banka (2018) attempted to generate chord progressions with reinforcement learning. As a number of different models are used to generate chord progressions in past literature, it is of our interest to compare some of them, namely the n-gram model, Markov model, and recurrent neural networks (RNN), in our project to evaluate their efficacy on this specific problem.

2.3 Stylized Chord Generation

Beyond general chord generation, our research focuses on the style of Joe Hisaishi's composition for Studio Ghibli films. We want to see how we can generate chords that are not only musical but also stylized, so the following papers focus on jazz, a genre proximal to Hisaishi's style. Ogihara and Li (2008) studied using weighted N-gram models of chord sequences to create a chord profile for a composer. They compared a set of jazz composers to a corpus of jazz standards to determine their similarity. Chen et al. (2020) proposed a chord jazzification process to generate realistic chord configurations in jazz styles using deep learning.

3 Methods

3.1 Data Creation

MIDI is a standard file format for storing and processing musical data. Through libraries like *Music21*, one can parse MIDI files and access symbolic information such as pitch, duration, and beat of a note in a measure. For this project, we input the piano reductions[3] of about 60 film soundtracks composed by Joe Hisaishi for Studio Ghibli into *Noteflight*, a music notation software. The exported MIDI files are parsed into chords to constitute the main corpora.

3.2 Chord Corpora

There are many parallels between music and natural languages (Ponsford et al., 1999). For our project, chords are analogous to tokens. Correspondingly, segmentation of a tune into discrete units of chords is comparable to tokenization. The unit for a chord is generally per measure. However, since some Studio Ghibli music has faster harmonic rhythms, we

[3]Transcribed in a published piano scorebook: Hisaishi, Joe, こんにちわピアノ曲集/ピアノ・ソロ 宮崎駿 & スタジオジブリ ベスト・アルバム [Piano Music Score Collection of Joe Hisaishi and Studio Ghibli: Best Album], ケイ・エム・ピー, May, 2015.

also experiment with parsing two chords per measure for pieces in 4/4 times and 6/8 times[4]. Thus, we have corpora of two types: a) large chord unit - one chord per measure, and b) granular chord unit - two chords per measure for time signatures with more beats. We assume that the granular chord unit would capture more subtle changes of chords and also reduce the number of ambiguous ones, since identifying one chord per measure may mix two distinct chords together when a measure contains more than one chord.

We represent each chord as a counter of notes first, where the weight of each note depends on its accumulated duration in that unit. Additionally, if a note occurs on the bass line or on a downbeat, we would increment its weight, since bassline and downbeats are generally the most important indicators for chords. Then, we remove notes with insignificant weights as they are more likely embellishing tones that don't belong to the chords. Finally, we represent each chord as a string of the most highly weighted X notes from the counter, separated by spaces, and the notes are sorted in alphabetical order so that we consider the same note combination as the same chords.

We experiment with X, the maximum number of notes to include in a chord, in addition to the chord units mentioned above. This further divides our parsed data into a total of 4 chord corpora: *max3*, *max3_per_mm*, *max5*, *max5_per_mm*. As an example, *max3* parses the chords from all Ghibli soundtracks in granular chord units with a maximum of 3 notes, while *max3_per_mm* parses the chords per measure with a maximum of 3 notes. We hypothesize that a higher maximum number of notes may produce results that model the Ghibli style better because of its characteristic use of extended chords.

Although the dataset size is much smaller than corpora in the field of NLP, we believe that the small universe of total possible chords balances out the issue. The ratio between the total number of chords and the number of unique chords in our corpora (Table 1) supports the claim. There are enough repetitions of distinct chords that we need to train the models.

[4]Studio Ghibli music composed by Joe Hisaishi contains 2/4, 3/4, 4/4, 6/8 times. Only 4/4 and 6/8 times have enough beat to be divided into two chords meaningfully.

Corpus	# chords (tokens)	# unique chords (types)
max3	5223	254
max3_per_mm	3059	205
max5	5223	526
max5_per_mm	3059	451

Table 1: Sizes of Corpora

3.3 Data Normalization

A chord progression is a succession of chords. The most important musical feature of a chord progression is not the chords per se, but rather the intervals[5] between the chords. In other words, the chords should be encoded in terms of their harmonic function within the chord progression. To do this, some previous studies like Whorley et al. (2007) and Shukla and Banka (2018) normalizes chords into scale degrees[6] or Roman numerals[7]. However, since scale degrees discard information about tension notes[8], and the Roman numerals get complicated to denote extended chords, the present study normalizes chords with Ogihara and Li (2008)'s approach of transposing all chord sequences of interest into the same key. Accordingly, a chord is always associated with a particular harmonic function and has a fixed relationship with other chords as there is only one universal tonic[9].

3.4 Baseline

For the baseline model, we incorporate a few templates of common chord progressions in any genre of music. These templates contain placeholders denoting the scale degrees for chords. We also create mappings from each scale degree to a pool of candidate chords. Then, to generate a chord sequence, we randomly choose a template and fill in each placeholder by a random candidate chord of that scale degree. In this way, our baseline model outputs chord sequences that respect basic harmonic rules, while the genericity of the templates should prevent it from modeling the Ghibli style.

[5]An interval is the difference in pitch between two sounds.

[6]A scale degree is the position of a particular note on a scale relative to the tonic.

[7]Roman numeral is a way to represent scale degrees with additional notations of chord inversion and tension notes.

[8]A tension is an extra note that is not in the basic triad of a chord. Inclusion of tension notes adds colors in the sound of the chords.

[9]A tonic is a fundamental note of a key.

3.5 N-gram

N-gram models are one of the most widely used methods for text analysis in NLP, involving contiguous sequences of n items from a corpus. N-grams were first proposed by Shannon (1951) and Miller and Selfridge (1950) and have been used in countless NLP papers. They also have been used for music generation by Ponsford et al. (1999), Ogihara and Li (2008), and others.

Applying n-gram to chord progression is not that different from text analysis especially with the string representations of chords. A tri-gram model, for example, learns the context of any chord by looking at the two contiguous chords before it. During training, we build a probability matrix that stores the likelihood from any n-gram context to any possible chord in the corpus. To generate a chord sequence, we randomly choose the next chord from all possible chords associated with the current n-gram context according to their probabilities. The first n-gram context for each generation will be n-1 start symbols that are also included in the training data. Once an ending symbol is encountered or the specified sequence length is reached, the model stops and outputs the generated chord sequence.

N in n-gram is varied as an experimentation. Since interesting chord progressions can often range in length, it is hypothesized that a small n (i.e, one or two chords as the context) will provide us with significantly less insight on the composer's style than longer sequences. However, if n gets too large, we may start to copy exact sequences from the corpus.

3.6 Hidden Markov Model (HMM)

Hidden Markov model (HMM) is another common method used in NLP for generating sequential data. HMM was first proposed by Baum and Petrie (1966) and has been used for musical data generation research. Ponsford et al. (1999) generated harmony using HMM, citing it as "more powerful than n-gram based models."

HMM has an underlying Markov model for hidden states, and an emission matrix that maps any hidden state to the probability of any observed event. For the task of chord progression generation, the observed events are the chord strings. Since we only have the MIDI data of Studio Ghibli music but not any annotated labels like harmonic analysis, we rely on a *music21* function[10] to analyze the key for each measure as the hidden states. The key for a measure is conceptually equivalent to the root of the chord for that measure, which influences the way we interpret a chord, and thus a valid hidden state for a set of possible chords. Since the key-analyzing function operates by measure at its most granular level, the HMM training is only run on the corpora with the large chord unit (chord per measure).

It should be noted that the key-analysis returns a key for a measure with a confidence level between 0 and 1. Our implementation treats any key with confidence level lower than 0.6 as 'NC' (No Chord / No Confident Key) because a lower confidence level suggests that the measure is very ambiguous - it may contain two distinct chords (fast harmonic rhythm) or it may be a transition measure with no meaningful chord.

The special thing about our implementation of HMM is that we use a variable-order markov model for the hidden states. In a variable-order markov model, the next hidden state is predicted not just by the current state, but also a variable-length of previous states. This is suitable for learning chord progressions due to its dependency on a number of chords. The concept of order makes a variable-order markov model very similar to an n-gram model. For example, for an HMM of order 3, we will be looking at 3 recent hidden states (tri-gram context of keys) to predict the next hidden state. Thus, the experimentation of order will be referred to as the experimentation of N as well for HMM in the later sections. Similarly, a higher N is hypothesized to model the style better, with the caveat that HMM may introduce more variety (unpredictability) when it generates an NC chord.

To generate a chord sequence, we first generate a sequence of hidden states (keys) using the variable-order markov model, which is the same as how we generate a sequence of chords in N-gram model. Then use the emission matrix to randomly choose a chord string for each key according to the probability.

3.7 Recurrent Neural Network (RNN)

In the same way that there are dependencies in a sentence of words, there too exists dependencies in a sequence of chords. In both cases, preserv-

[10] Stream.analyze('key'): This function implements the Krumhansl-Schmuckler key determination algorithm.

ing the implications of previous tokens is necessary to producing a logical output, so RNNs are an obvious choice for generating chord progressions. RNNs have been used in a great number of musical generation studies, one of the earliest examples being Mozer (1994) which focused on using neural networks to compose music by prediction. More recently, Brunner et al. (2017) used LSTMs to improve upon the vanilla RNN model as a means of generating chords as part of their study.

In essence, we believe the neural network can learn from sequences of chords in the same way it would learn from sentences. A major difference between the RNN model and our other models is that it creates chords note by note instead of selecting candidate chords that already exist in the dataset, so this should be noted when evaluating. We utilize the Keras Python library, specifically the Sequential class and LSTM contained within the module in order to train our model. Several models are trained using varying sequence lengths of chord data. While we hypothesize that longer sequence lengths will yield more stylistically similar results, it will be interesting to see if we get the best results from a sequence length that matches the lengths of common chord progressions. Once the model is trained, we predictively generate chords of varying note lengths based on a starting input.

4 Results

Evaluation is a tricky and challenging topic for generative models due to their creative nature and the subjectiveness involved to assess the goodness of outputs. Most generative models in NLP rely on human evaluations. For example, Pudaruth et al. (2014) built a lyrics generation model and asked people to guess if a lyric is an existing or a generated one. Such Turing tests[11] are also used for music generation models, where people are instructed to identify music they consider to be composed by a human as opposed to a computer (Ariza, 2009; Yang and Lerch, 2020). However, human evaluation is expensive in terms of time and labor. Our project poses even more challenges to find enough human evaluators, as it would require familiarity with the Studio Ghibli music style. Thus, human evaluation is out of scope for now, but it remains to be a meaningful future study to carry out.

[11]The Turing test follows an intuitive concept that evaluates whether a machine is able to exhibit behavior indistinguishable from humans.

We perform two computational evaluation metrics on the outputs: 1) longest common subsequence (LCS) of chords and 2) the number of pieces that had the same sequence of roots as our generated chord sequence (same_sequence_number, SSN). For n-grams, HMM, and the baseline we generate 100 sequences for each category, and for RNN we generate 25, 35, and 50 note sequences (since they were generated note by note instead of chord by chord).

The LCS evaluation metric takes a generated sequence and a piece from our corpus and returns the length of the LCS. For each model, we take the maximum LCS between the entire corpus and each generated sequence, and then average the LCSs from all generated sequences in that category. The maximum score for LCS is equal to the length of the sequence that was used. For example, if the generated sequence is four chords long, the maximum LCS score would be four.

The SSN metric takes a generated sequence and a directory within the chords folder (max3, max5_per_mm, etc.) and compares the roots of the generated sequence to the roots for all pieces in the corpus. It returns the number of pieces that contain that root sequence. We average the same sequence numbers of all generated sequences for each category. However, it is important to note that we use the *.root()* function from the *music21* library, but because there are multiple ways to interpret any chord this function is not entirely reliable. Furthermore, we noticed that the .root() function struggled with correctly identifying the root for certain chord types, such as suspension chords. The maximum value for SSN is 63, the number of pieces in the corpus.

When considering the different scores, it is important to consider how the LCS and SSN scores compare to each other (Table 2). It is also important to consider that high and low scores for both metrics vary depending on the length of the chord sequence. For our data, the standards of good scores for LCS and SSN by sequence length are shown in Table 3.

Now that we have established what we are looking for in our results, we will discuss the overall trend of how different types of configurations affect the results. Since we run experiments with a combination of different configurations, the full table for the results can be found in appendix A.

	High SSN	Low SSN
High LCS	High LCS and SSN scores means the chord progression was likely copied directly and it is a very common sequence that appears multiple times in the corpus	A high LCS score points to likely copying, but a low SSN score means it's copying a sequence that only appears in a small number of tracks.
Low LCS	Our optimal result, low LCS likely meaning that the chords are not being copied directly, but a high SSN score meaning the root progression still is prevalent in the corpus.	A poor result, where neither the chord progression nor the root progression arises in the corpus.

Table 2: LCS and SSN scores in conversation with one another

Sequence Length	Good LCS	Good SSN
4	<3	>10
8	<6	>5
12	<8	>3

Table 3: Standards for good LCS and SSN scores by sequence length

4.1 Corpora difference 1: chord unit

The large vs. granular chord unit distinction could not be assessed by our metrics. However, the generated outputs do support our assumption that a granular chord unit is better at capturing chord changes and thus producing cleaner chords, while a large chord unit sometimes generates chords with more dissonance because the training data can be mixing two chords into one for measures with faster harmonic rhythms. The music scores of selected outputs can be found in appendix B[12].

4.2 Corpora difference 2: max number of notes in a chord

For both HMM and n-grams, having a max note value of 3 resulted in a higher SSN value, but for HMM max3 had a higher LCS value, while for n-grams max3 had a lower LCS value. It seems that for n-grams a smaller max note value produced better results, but HMM the results point to something more complex. We suspect that having fewer notes in the HMM sequences makes it more likely to repeat certain sequences, leading to both higher LCS and SSN scores for max3.

[12]The corresponding audio is included in Github: https://github.com/LingQi000809/Chord-Generation

4.3 N for N-grams and HMM

For n-grams, across the board, the LCS values are high, many falling above the thresholds listed above, and the SSN values are largely only above the SSN threshold when the LCS threshold is also high. This points to a large amount of copying, regardless of n value. However, n2 and n3 did have a small number of values that were considered "good" by our metrics outlined above. n5 and above seem to be copying too much, with LCS scores very near the sequence length.

Although the LCS for all n values in HMM are considered "good" by our metrics, HMM scores low for SSN across the board, resulting in a poor similarity score overall (still better than baseline on average). We hypothesize that this is due to the possibility of introducing No Confident Keys (NC), which are associated with a larger range of chords that can have totally different roots.

4.4 Sequence length

For sequence length, it seems obvious, but lower sequences had more matches for SSN, since it is more likely for a sequence to appear in a piece if it is shorter. However, the possibility that shorter sequences are copying is interesting to investigate. Because they are shorter, the chance that they have a higher average LCS score is unsurprising, but there is a chance that those sequences were generated without copying through n-grams and HMM. We want to draw the reader's attention to this and lower the weight of a "bad" LCS score if the sequence length is 4 and the LCS score is 4 compared to if the sequence length is 12 and the LCS score is 12. Generally, it was rare for longer sequences to exist within our thresholds for an ideal result, but did occur a small number of times.

4.5 RNN

Because the RNN methodology is so different from n-grams and HMM, we discuss those results separately in this paragraph. The first thing of note is that directly copying chord sequences is not as much of an issue for RNN, since it generates sequences note by note instead of chord by chord. Therefore, an adjusted "good" LCS score should be considered for the RNN results. We did notice a higher average LCS score all around when the model was trained on sequence lengths of 20 notes. This could indicate similar chord progressions were easier to produce when the sequence length is closer to the length of an actual chord progression. Finally, the SSN scores were generally low (similar to HMM), and we suspect this is because Hisaishi uses a variety of chord progressions for different pieces. This also could be because the RNN has the ability to generate chords outside of the original dataset, causing even more inaccuracies of determining the root note of a chord.

All of our models produced interesting results that could be investigated further with a more robust corpus, better tagging for training and evaluation, and a broader range of evaluation metrics. These additions would lead to more conclusive results and reveal more about the possibilities surrounding chord generation.

5 Future Works

Our project was a large undertaking for three undergraduate students, and we are proud of what we accomplished; however, there are lots of areas related to (Studio Ghibli) chord generation that we were unable to address in the time that we had. To begin with, we do not have annotated data for our Ghibli corpus, and this placed limitations on our HMM and our evaluation metrics. With sufficient time, the data could be annotated with the harmonic function of each chord. Another weakness of our study was the small size of our corpus. With only 63 pieces, the number of chords is limited for training and evaluation. Since Studio Ghibli music uses similar harmonic sequences and chord extensions as Jazz standards, including a background corpus of Jazz music[13] to offset the small dataset may provide significantly different results.

Another series of changes that could be done are changing the calculation of the root of each chord and adjusting the handling of confidence levels for the HMM. Currently, we have a threshold where under 0.6 confidence level we say the measure has no confident key. The emission matrix for an NC measure will have a larger variety of chords compared to the more confident keyes. This could be a possible reason for the HMM results. Finally, a comparison between large or granular chords is not covered by our evaluation metrics, and an exploration of how more granular chords might affect generation.

[13] A possible Jazz corpus was transcribed by Doug McKenzie: https://bushgrafts.com/midi/

References

Torsten Anders and Eduardo R Miranda. 2009. A computational model that generalises schoenberg's guidelines for favourable chord progressions. In *Proceedings of the Sound and Music Computing Conference*, pages 48–52. Citeseer.

Christopher Ariza. 2009. The interrogator as critic: The turing test and the evaluation of generative music systems. *Computer Music Journal*, 33(2):48–70.

Leonard E Baum and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563.

Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Jonas Wiesendanger. 2017. Jambot: Music theory aware chord based generation of polyphonic music with lstms. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 519–526. IEEE.

Adam Burnett, Evon Khor, Philippe Pasquier, and Arne Eigenfeldt. 2012. Validation of harmonic progression generator using classical music. In *ICCC*, pages 126–133.

Tsung-Ping Chen, Satoru Fukayama, Masataka Goto, and Li Su. 2020. Chord jazzification: Learning jazz interpretations of chord symbols. In *Proc. Int. Society for Music Information Retrieval Conf.*

Arne Eigenfeldt and Philippe Pasquier. 2010. Realtime generation of harmonic progressions using controlled markov selection. In *Proceedings of ICCC-X-Computational Creativity Conference*, pages 16–25.

Vincenzo Madaghiele, Pasquale Lisena, and Raphaël Troncy. 2021. Mingus: Melodic improvisation neural generator using seq2seq. In *22nd International Society for Music Information Retrieval Conference*.

George A Miller and Jennifer A Selfridge. 1950. Verbal context and the recall of meaningful material. *The American journal of psychology*, 63(2):176–185.

Michael C Mozer. 1994. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280.

Mitsunori Ogihara and Tao Li. 2008. N-gram chord profiles for composer style representation. In *ISMIR*, pages 671–676. Citeseer.

Dan Ponsford, Geraint Wiggins, and Chris Mellish. 1999. Statistical learning of harmonic movement. *Journal of New Music Research*, 28(2):150–177.

Sameerchand Pudaruth, Sandiana Amourdon, and Joey Anseline. 2014. Automated generation of song lyrics using cfgs. In *2014 Seventh International Conference on Contemporary Computing (IC3)*, pages 613–616. IEEE.

Claude E Shannon. 1951. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64.

Shipra Shukla and Haider Banka. 2018. An automatic chord progression generator based on reinforcement learning. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 55–59. IEEE.

Raymond P Whorley, Geraint A Wiggins, and Marcus T Pearce. 2007. Systematic evaluation and improvement of statistical models of harmony. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*, pages 81–88. Goldsmiths, University of London London.

Jian Wu, Changran Hu, Yulong Wang, Xiaolin Hu, and Jun Zhu. 2019. A hierarchical recurrent neural network for symbolic melody generation. *IEEE transactions on cybernetics*, 50(6):2749–2757.

Li-Chia Yang and Alexander Lerch. 2020. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784.