

# Network Analysis of Chord Progressions in Rock and Jazz Music

Nicholas Bien, Harper Carroll, Neel Ramachandran

December 9, 2018

## 1 Introduction

In this report we analyze the network structure of rock and jazz music. Most projects and research that have conducted computational music analysis use lyrics (i.e. text-based methods) or the music itself (i.e. audio-based methods) as inputs. Less literature has taken a computational approach to studying the notes or chord progressions of a song. We hypothesize that modeling music as a network—where nodes are chords and edges are between consecutive chords—will be an effective (and relatively novel) way to analyze musical structure.

Our dataset consists of chord progressions from 149 jazz songs and 212 rock songs. We define two types of graphs—the genre graph and the song graph—and analyze the basic network characteristics, communities, motifs, and roles present in these graphs. We introduce genre classifiers based on motif counts and node2vec walks. Finally, we experiment with using network-based properties to generate chord progressions in the styles of both genres. We find, as expected, that the jazz and rock genres differ in terms of chords commonly played. More interestingly, we find significant differences in the network structures of the genres.

## 2 Related Work

In “Recurring harmonic walks and network motifs in Western music”, Izkovitz et al. [1] analyze network motifs in graphs constructed from chord progressions in classical and contemporary pop music. They identify 13 consensus motifs, which occur in 60% of songs in their dataset, and develop a method for classifying songs into one of 7 families based on the motifs present in the song’s chordal graph. This paper provides strong motivation for applying network analysis to musical sequences and appears to be the first to do so. The authors explain the music theory concepts underlying the common motifs they found. The paper is lacking in analysis of the music graphs beyond the scope of motifs: the discussion of classification is based solely on motifs and is limited by the lack of an evaluation criterion. We classify subgraphs (songs) by genre and evaluate the classifier using extracted labels from our data source. We model our data using the same basic approach: the nodes are chords and the edges indicate a transition from the origin chord to the destination chord, weighted by frequency in the corpus. However, we use a different method for grouping similar chord types. Izkovitz et al. left off added notes above the triad, but these notes are much more common in jazz music and are indicative to the harmonic functioning of each chord. For Izkovitz et al., “A, A6 and A/C were all considered A”. We would consider still consider A/C as A, since the inversion of the chord doesn’t significantly affect its harmonic function. However, we would consider Amaj7 different from A7, since these chords are common in jazz and function very differently. We also perform similar network motif analysis on our corpus of rock and jazz chord progressions.

Building on Izkovitz’s ideas, “Guitar solos as networks” by Stefano Ferretti [2] discusses modeling music as a network. In his paper he proposes the network model for music which Izkovitz also uses but with notes as nodes, and analyzes some key network characteristics of famous guitar solos. Ferretti reports some interesting facts on differences between the network structure of different songs, but the analysis

is limited to simple metrics like degree distribution and clustering coefficient. He does, however, show that individual songs across different genres have dramatically different graph structures. He also suggests the potential of the network model as a basis for music generation. Here, we build on Ferretti’s work by comparing network structures across genres. Looking at more complex graph features like communities and roles and motifs, as well as the basic features provides a more thorough and interesting comparison. Feretti looks at the network structure of individual songs, which we incorporate into our motif analysis and genre classification. Finally, we expand on Feretti’s music generation idea by building a random walker that can generate songs in the style of different genres.

### 3 Dataset and Representation

#### 3.1 Dataset

We extracted chord progressions from two sources: 1) 212 of the most popular chord tablatures posted on ultimate-guitar.com in the "Rock" genre and 2) 149 song in Lilypond transcription format from openbook, an open-source repository of jazz standards. Since we extracted from the most popular songs rock songs on ultimate-guitar.com, and since jazz standards define the genre of jazz, we believe these datasets offer a sample of the most canonical and influential in these genres. Song titles in the rock dataset include "Hey Soul Sister" (Train), "Hotel California" (The Eagles), "Let It Be" (The Beatles), and "Wonderwall" (Oasis). Song titles from the jazz dataset include "Giant Steps" (John Coltrane), "Ornithology" (Charlies Parker), "St. Thomas" (Sonny Rollins), and "What a Wonderful World" (Louis Armstrong).

Chords tokens were extracted from text files and converted to a common schema:

[root note][triad quality][extension].[alterations]

where triad quality is one of (maj/m/dim/aug) [maj is left off if the chord has no extension], extension is zero or one of (6/7/9/11/13), and alterations is zero or any number of (5-/5+), (9-/9+), (11-/11+), or (13-/13+). For example,

C (C-E-G)  
 Cmaj9 (C-E-G-B-D)  
 C7.5- (C-Eb-Gb-Bb)

represent C major triad, C major ninth, and C dominant seventh flat five, also called C half-diminished seventh), respectively. Note that ninth chords and similar extension chords (elevenths, thirteenth) imply the seventh note as well. The rock dataset additionally includes suspended second, suspended fourth, add9 chords, and power chords. With root note C, these chords are:

Csus2 (C-D-G)  
 Csus4 (C-F-G)  
 Cadd9 (C-E-G-D)  
 C5 (C-G)

We then used the key of each song, also extracted from the text files, to transpose all songs to C major or the relative minor key A minor. In total, there are 139 distinct chords in the rock dataset and 241 in the jazz dataset, with 87 chords appearing in both datasets. The most common rock and jazz chords are shown in Figures 1 and 2.

#### 3.2 Genre Graph

We define a multigraph  $G = (V, E)$  where  $V$  is the set of all chords present in any song in a genre and  $E$  is the set of all chord transitions occurring in the songs in the dataset. Multigraphs allow for edges

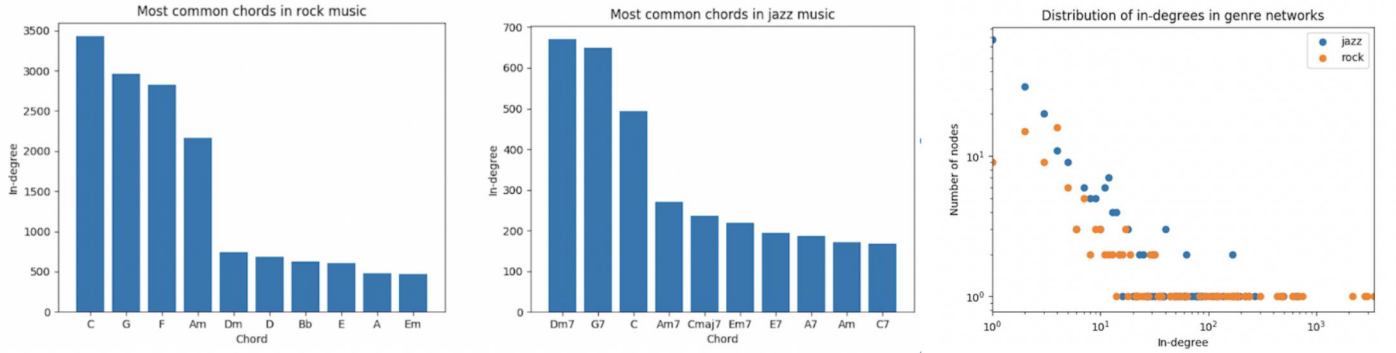


Figure 1: Left: Frequency of the most common chords in rock music (left) and jazz music (right), as measured by in-degree. Right: Degree distributions for rock and jazz music. Note that the distribution of jazz chords a longer tail, indicating that most of rock music is composed of a few common chords, while jazz music typically has a wider variety of chords.

between nodes to be added many times, helping to implicitly create edge weighting for our random walk generation discussed below. Aggregating the chord changes from a large set of songs in a given genre allows us to deduce canonical patterns of the genre from the properties of the genre graph. See Table 1 for basic statistics about our genre graphs, and see Figures 2 and 3 for the degree distributions of each genre graphs and their visualizations.

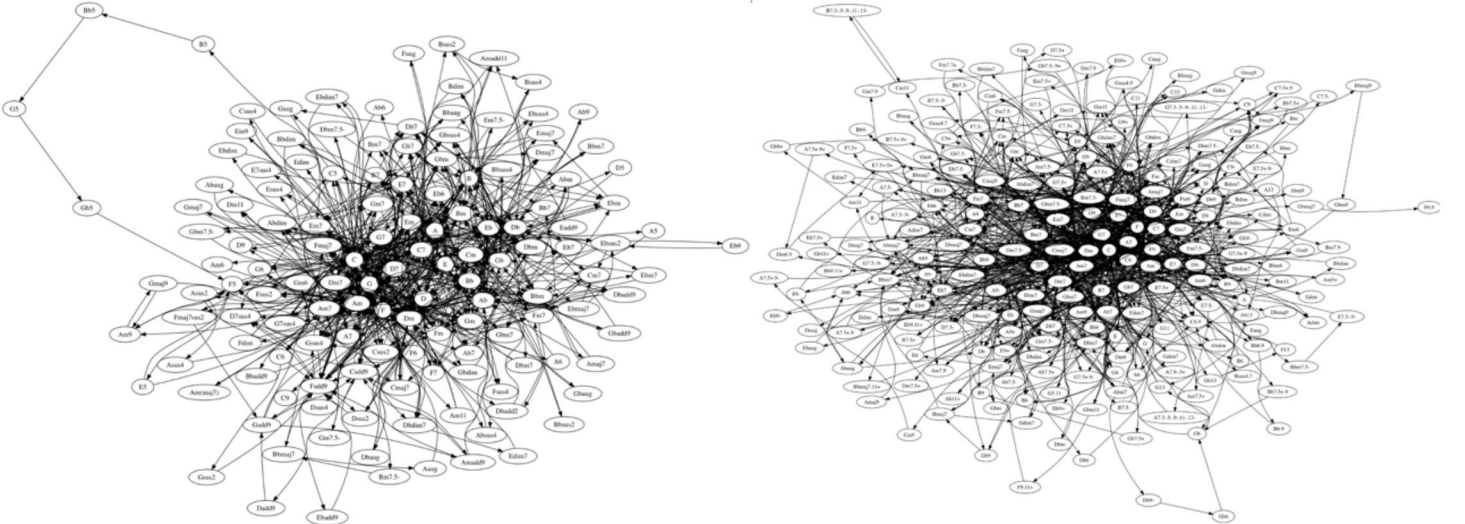


Figure 2: Visualizations of Genre Graphs for rock music (left) and jazz music (right).

### 3.3 Song Graph

For each song, we define a multigraph  $S = (V, E)$  where  $V$  are the chords in the song and  $E$  are directed edges corresponding to chord transitions within the song. Note that each Song Graph  $S$  is a subgraph of that song's genre's Genre Graph  $G$ . Analyzing individual songs as individual graphs is useful for motif detection, as Itzkovitz et al. found [1], as the genre graphs are densely connected and have messier motif structures due to interconnections between chords from across different songs. See Figure 4 for visualizations of example song graphs from each genre.





We can then find the most “similar” chords to a given reference chord by finding the chords with the highest cosine similarities to the reference chord. Results of this analysis are shown in Table 7.

## 4.4 Genre Classification

Previous approaches have classified songs based on the notes or chords present using language methods like Naive Bayes. We wanted to see if it is possible to classify songs based solely on their graphical structure without any chord labels.

We propose two methods for creating features for genre classification: 1) using motif counts and 2) using node2vec embeddings. For motif counts, we represent each song  $S$  as a vector  $x$  where  $x_i$  is the number of times motif  $i$  appears in song  $S$ . We concatenate counts for 3-node motifs and counts for 4-node motifs. We also trained embeddings for each song graph using node2vec [5]. Specifically, each song embedding is the mean of all node embeddings within the song. For a song graph  $S = (V, E)$ ,

$$x_S = \frac{1}{n} \sum_{i \in V} \text{embedding}_i,$$

where  $\text{embedding}_i$  is the node2vec embedding for node  $i$  in the graph  $S$ .

We split our combined dataset (rock and jazz music together) into a training set and a test set. We train a support vector machine to predict the genre of a song given its node2vec song embedding. We compare our results to a simple baseline with single-dimensional embeddings using the number of unique chords in the graph:  $x = |V|$ . We also compare our model to a chord-aware classifier; specifically, we implement a Bernoulli Naive Bayes model, analogous to a document classifier with chords instead of tokens. Given a feature vector  $x_S$  for song  $S$  including an index for each chord type (across both datasets) and having a 1 for the indices of chords that appear in  $S$  and a 0 for all indices of chords that do not appear in  $S$ , the output of the Naive Bayes classifier is:

$$\arg \max_y \prod_i p(y|x_i) = \arg \max_y \prod_i p(y)p(x_i|y),$$

where  $p(y)$  is the probability of seeing label  $y$  (jazz or rock) in the training dataset and  $p(x_i|y)$  is the learned likelihood of seeing chord  $x_i$  in genre  $y$  in the training set. We compare the performance of our node2vec classifier to the simple baseline and the Naive Bayes model at training-to-test splits from 10% to 90% intervals. We also evaluate our classifier using node2vec walk lengths from 10 to 100.

## 4.5 Generating Chord Progressions

We are also interested in using chord progressions modeled as networks to generate new chord progressions. We crafted 3 different algorithms for this purpose. For every set, we kept SONG\_LENGTH (or chord progression length) and NUM\_GENRE\_SONGS constant, both at 100. As our first delve into this topic, we used our graphs as MDPs and created 100 rock songs and 100 jazz songs by randomly walking from the first node in each genre’s multigraph through chords via out edges until reaching either an end node (i.e. a chord with no chords after it) or the maximum chord progression length. The directed multigraph representation of each genre was the most appropriate model to use, as randomly choosing any edge from a node implicitly adds more weight to duplicate edges. With the multigraph, the probability of choosing an edge to a neighbor node is proportional to the popularity of that neighbor node.

**Data:**  $G = \text{"rock", "jazz" genre MultiGraphs}$   
**Result:**  $song$ , generated song of genre  
 $node = \text{first node of } G$ ;  
 $song = []$ ;  
**while**  $length(song) < MAX\_LENGTH$  *and*  $nodeOutDegree > 0$  **do**  
    add  $node$  to  $song$ ;  
     $neighbors = neighbors(node)$ ;  
     $dstID = \text{random.choice}(neighbors)$ ;  
     $node = getNode(dstID)$ ;  
**end**

**Algorithm 1:** overview of Random Walk algorithm

Our second approach uses the biased walks from the Node2Vec algorithm. We hypothesized that such biased random walks on our genre graphs could potentially generate realistic genre-based chord progressions; biased random walks are like another approach to the local-search versus exploratory search ratios aimed for by our second approach.

**Data:**  $G = \text{"rock", "jazz" genre graphs}$   
**Result:**  $song$ , generated song of genre  
 $probs = [\text{list of node ids} \times \text{their out-degree}]$   
 $node = \text{random.choice}(probs)$   
 $p, q = 5, 100$   
 $song = \text{runNode2VecWalk}(MAX\_LENGTH, node)$

**Algorithm 2:** overview of Biased Walk algorithm

The third approach built off of our community detection algorithm, which takes an undirected graph as input and outputs a list of community sets. For each genre set, we extracted communities and then chose a random chord from the largest community to begin our song. With some probability  $OUT\_THRESHOLD$ , we step exclusively to neighbors within the cluster, and otherwise, we step to any neighbor or any previously-visited node. The idea was, like songs moving between choruses and verses, to stay mostly within communities, and every now and again return to a previously-explored community. See Algorithm 3 in the appendix for the detailed pseudocode.

## 5 Results & Analysis

### 5.1 Graph Characteristics and Genre Comparison

	rock	jazz
number of nodes	139	241
number of edges	20281	6520
clustering coefficient	0.1269	0.064171
density	2.1146	0.2254

Table 1: Comparison of basic statistics between rock and jazz Genre Graphs.

According to Table 1, the jazz graph has more nodes, fewer edges, a lower clustering coefficient, and significantly lower density. This confirms our intuition that jazz music features a wider variety of chords arranged in more complex patterns in comparison to rock music.

C	G	Am	C	G7	Am7
Am	F	Em	Dm7	Ebdim7	A7
C6	G7	Dm	Cmaj7	G9-	Em
D7	Am7	D7	C6	A9-	F
Em	Dm7	Fadd9	Dm7.5-	G9	A9-
C7	D	D	Dm	A7	F6

Table 2: Most similar chords based on Rolx features for rock music (left) and jazz music (right). The chords used for comparison are in the heading. The five chords with the highest cosine similarity to the comparison chord are under the heading. For the rock chords, we notice that chords similar to C are often “home” chords in C major/A minor in both the rock and jazz graphs. Chords similar to G in the rock graph are chords that could be one step away from a home chord. Chords similar to G7 (the most common dominant form in jazz music) are all highly unstable and would be expected to resolve to a home chord.

Table 2 contains role detection analysis for 3 common chords from each genre. We see that the notes in a chord clearly influence that chord’s role in the graph: looking at the first column, Am, C6, Em, and C7 all share at have 2 of the 3 notes in the C chord, for instance. We also see that the predominant tonic chords in the relative minor key (A minor and A minor seventh) are both very closely related to the dominant in A minor, the E minor triad. Another interesting observation is that G7 is similar to G9- (G dominant seventh flat nine) and G9 (G ninth), which both contain all four notes of G7 (G-B-D-F). We do see some noise in our similarity measure; for example, Ebdim7 does not share any notes with G7 but is most similar according to our features.

See Tables 1 and 2 in the appendix for communities in the rock and jazz genre graphs. We observe generally that communities rock music tend have similar number of accidentals (black keys), while communities in jazz music have similar numbers of alterations and stacked notes.

In Table 3, we report a subset of the results of motif detection, showing motifs which had the highest Z-scores in the jazz and rock genre groups. We observe significant Z-scores in both groups; however, the Z-scores for jazz motifs are generally much higher than the rock motifs (especially among size four motifs), signifying that motifs are over-represented to a higher degree in jazz. From a musical perspective, this aligns with common perceptions of jazz as a genre with intricate, structured patterns and progressions. We also observe that the motifs with the highest Z-scores (ie 3-7, 4-31, 4-50, 4-116, 4-159) involve bi-directional edges, which suggests that songs often alternate between a small subset set of chords for a period of time (ie during a chorus), another commonly held notion. Finally, we can also see that there are motifs that are integral to the jazz genre but not the rock genre (ie 4-29, 4-50), and likewise, motifs that are integral to the rock genre but not the jazz genre (ie 4-116, 4-159), suggesting that there are fundamental differences in the patterns present in the genres. We leverage this insight to build a genre classification model based on the motif counts in the next section

## 5.2 Genre Classification

As seen in Figure 4, the motif count-based SVM always slightly outperforms the unique chord count-based SVM. As shown in Table 3, there are significant differences in the motif structures of the Jazz and Rock genres, which is why using the motif counts as an input works effectively. The motif counts also roughly capture the amount of chord variety present in the song, which is probably why these classifiers perform similarly. The node2vec-based classifier performs nearly as well as Naive Bayes with sufficient training data. This is noteworthy because the node2vec model is unaware to the chord types present in each song—it learns features solely based on graph structure. This result is somewhat surprising and indicates that the nature of chordal movement is perhaps as integral to a genre as the chords themselves.

Another noteworthy aspect of our node2vec+SVM classifier is that it requires at least 60% the training

















Motif Id	Motif Pattern	% songs occurring in (Jazz)	Z-scores (Jazz)	% songs occurring in (Rock)	Z-scores (Rock)
3-0		95%	48.0	56%	31.3
3-1		99%	100.2	78%	31.93
3-6		70%	80.9	66%	58.1
3-7		26%	316.3	40%	138.7
3-8		63%	115.0	49%	52.9
3-9		29%	54.9	57%	51.3
3-11		8%	37.0	51%	74.7
4-29		59%	157.49	25%	46.83
4-31		21%	543.0	19%	23.9
4-44		21%	353.0	16%	22.4
4-50		32%	356.3	17%	34.7
4-51		28%	148.2	23%	30.3
4-61		31%	104.5	18%	26.9
4-103		34%	26.3	8%	25.8
4-116		2%	19.7	11%	156.3
4-159		3%	33.0	14%	163.0

Table 3: Motif analysis.

data (about 200 songs) to outperform the simple unique chord counts-based classifier. We hypothesize that there are a limited number of representative graph structures for the two genres, and that once the classifier sees enough of these examples it can distinguish well between the two genres.

The effect of walk length on the node2vec classifier is also interesting. The average number of chords per song is 7.26 for rock and 17.27 for jazz. However, in order for node2vec embeddings to be effective in classifying an entire graph, it appears that the walk length needs to be 2-3 times longer than the number of nodes in the graph. This long walk length relative to the size of the song subgraphs means that each node2vec embedding captures information about the entire graph. In fact, using the max of the node2vec node embeddings instead of the mean, or even choosing a random node embedding, works just as well because all of the node embeddings capture similar information.



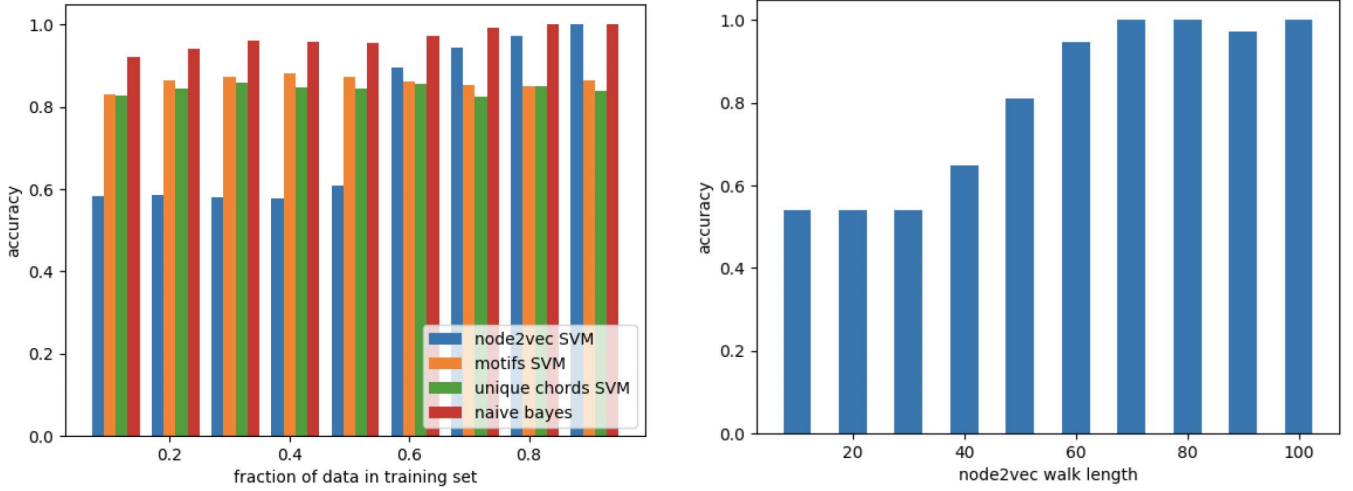


Figure 4: Left: Classifier accuracy at various training/test data splits using walk length 80. Right: Accuracy of the node2vec+SVM classifier with node2vec walk lengths ranging from 10 to 100 (using training/test split 75%).

### 5.3 Music Generation

To test and compare the chord progressions generated with our three algorithms, we used the Lilypond program to generate mp3 outputs and judged by ear. We also generated graphs for each of our chord progressions and passed these graphical representations through our genre classification model to see how well our SVM predicted our randomly-generated songs’ genres. We found that our generated results reflect the patterns typical to their respective genres, with minute differences resulting from the different algorithms, which we compare below. For each music generation algorithm below, we select a randomly-generated song from each genre.

#### 5.3.1 Results: Random Walk Algorithm

Random Walk rock song #19 (left) and jazz song #18 (right):

C Dm Em G7	Cmaj7 E7 Am F
C Dm C Dm	B7 Em A7 D9
C G F G	Dm7 G7 Em7 A7
F G F C	D7 G7 C6 Am7
	Dm7 G7 Cmaj7

These progressions exhibit some of the characteristic features of each genre: the rock song utilizes mostly simple triads, with a lot of movement around the I-IV-V chords (C-F-G). The jazz song uses “jazzier” seventh chords and extensive circle of fifths movement (VI-II-V-I). Specifically, the second line (Dm7-G7-Em7-A7) and the last four chords (Am7-Dm7-G7-Cmaj7) are both VI-II-V-I progressions—the first being a modulation to D major and the second returning to the home key of C major. This sort of movement along the circle of fifths, especially involving modulations to different keys, is a cornerstone of jazz chordal theory.

#### 5.3.2 Results: Biased Walk Algorithm

Biased Walk rock song #2 (left) and jazz song #92 (right):

C F G Am	Am F C Em F
Em C D G	Fmaj7 Cmaj7 G7 Dm7 C
F C Dm F	Am7 D7 Dm7 G7 C
G C Bb Gm F	F Gbdim7 G7 C

These excerpts exhibit more consistent harmonic movement than the Markov-generated excerpts and also fewer out-of-place chords than the Smart Walks. Some cool aspects include modulation to F major in the last line of the rock example and a chromatic F-Gbdim7-G7 movement in the last line of the jazz example. Though sometimes producing very *un-song-like* chord progressions, this algorithm also produces the *most* unique and song-like chord progressions.

### 5.3.3 Results: Smart Walk Algorithm

Smart Walk rock song #11 (left) and jazz song #98 (right):

Fmaj7 C Am G	Fm6 Em7 Am Fm6
C Am C G	G7 C G9 Cmaj7
Am G C F	Cm7 F9 Bm7.5- Cmaj7

This algorithm performed slightly worse than we expected it to as more highly unusual chords and sequences were played than in the random walk. This is probably because of our community detection algorithm: since we are grouping nodes into only two very large clusters, the cluster assignments are likely to be less nuanced than those in a graph with many possible cluster assignments. This algorithm also incorporates another level of randomness that the random walk does not: teleportation is possible, which means chord sequences unseen in the genre graph can be formulated, which can cause performance in genre classification methods to suffer.

## 6 Conclusions

In this report, we show that modeling songs and genres as networks is an effective and novel approach to music analysis.

In general, we found that motif analysis gives better results than community- or role-detection in musical graphs. This possibly reflects the fact that a chord’s meaning changes depending on the chords around it, leading to highly overlapping communities and roles. However, motif structures are common because they capture the rules and patterns associated with chord changes.

We were surprised to find that a classifier based on the structure of song’s chord graph alone, without actual chord labels, could perform about as well as a classifier using chord types. More research is needed to assess graph-based classifiers of music on larger, more diverse datasets.

We were not surprised that song generation proved to be difficult. We also struggled due to the lack of a scientific evaluation metric for generated songs. In our opinions, however, we were successful in generating unique songs using graph-traversal techniques like biased random walks. By encouraging BFS over DFS at an approximate ratio, we can return to old chord clusters the same way real songs do.

A limitation of our dataset is that we have no sense of time between each chord change. Our data captures chord progressions but does not give any indication of how long each chord should be played. Thus, when we model our randomly generated songs, each chord is by default played for the same amount of time, which is unlike most real songs. A dataset with information about each chord’s length would allow us to generate far more realistic and potentially enjoyable songs. Further, we chose to work with chord progressions for this report, but the same work could easily be extended to note progressions. Future work could build upon the presented results by taking advantage of these two limitations.

## 7 Repository

<https://github.com/neelr11/cs224w>

## 8 References

1. Shalev Itzkovitz, Ron Milo, Nadav Kashtan, Reuven Levitt, Amir Lahav, Uri Alon. Recurring harmonic walks and network motifs in Western music. Advances in Complex Systems. 2006. <https://www.worldscientific.com/doi/abs/10.1142/S021952590600063X>.
2. Stefano Ferretti. Guitar solos as networks. IEEE International Conference on Multimedia and Expo. 2016. <https://ieeexplore.ieee.org/document/7553001>.
3. Mathieu Barthet, Mark Plumbley, Alexander Kachkaev, Jason Dykes, Daniel Wolff, Tillman Weyde. Big Chord Data Extraction and Mining. 9th Conference on Interdisciplinary Musicology. 2014. [http://www.staff.city.ac.uk/~sa746/Barthet\\_et\\_al-2014\\_Big-Chord-Data-Extraction-And-Mini.pdf](http://www.staff.city.ac.uk/~sa746/Barthet_et_al-2014_Big-Chord-Data-Extraction-And-Mini.pdf).
4. Aaron Clauset, M. E. J. Newman, Cristopher Moore. Finding community structure in very large networks. Physical Review E. 2005. <http://ece-research.unm.edu/ifis/papers/community-moore.pdf>.
5. Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. Knowledge Discovery and Data Mining. 2016. <https://cs.stanford.edu/~jure/pubs/node2vec-kdd16.pdf>.
6. Natasha Jaques, Shixiang Gu, Richard E. Turner, Douglas Eck. Generating Music by Fine-Tuning Recurrent Neural Networks with Reinforcement Learning. Deep Reinforcement Learning Workshop, NIPS. 2016. <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45871.pdf>.

## 9 Appendix

#1: 41 nodes	#2: 28 nodes	#3: 13 nodes	#4 29 nodes	#5: 24 nodes	#6: 4 nodes
C	F	G7	Eb	Bm	B5
G7sus4	D	D7	Bb	A	Bb5
G	Dm	B7	Bbm	Gbm7	G5
Am	Em7	Em	Ab	Gb	Gb5
Cadd9	Em9	Gm7.5-	A5	Gbsus4	

Table 4: Chord communities in rock music.

#1: 55 nodes	#2: 60 nodes	#3: 55 nodes	#4 60 nodes	#5: 2 nodes	#6: 9 nodes
Cmaj7	Em7	Ebdim7	Bb13	Gm7.9	G
Dm7	Am	G7	Am7.5+	Gb7.5-.9+	Abdim
Eb7	C7	C6	Am7		A13
Am9	Fmaj7	C	A7		Ab13
D7	Gbdim7	D9	Bm7		G13

Table 5: Chord communities in jazz music.

**Data:**  $G$  = "rock", "jazz" genre UnDirected Graph  
**Result:** *song*, generated song of genre  
*clusters* = get\_communities( $G$ );  
*song, visited* = [ ], [ ];  
*cluster\_index* = index of largest cluster in *clusters*;  
*node* = random choice in *clusters*[*cluster\_index*];  
**while** *song length* < *MAXLENGTH* **do**  
    add *node* to *song*;  
    add *node* to *visited*;  
    *neighbors* = neighbors(*node*);  
    *probs, possible\_next* = [ ], [ ];  
    *rand* = random float between 0 and 1;  
    **if** *rand* < *OUT\_THRESHOLD* **then**  
        | *possible\_next* = *neighbors*  $\cap$  *clusters*[*cluster\_index*] ;  
    **else**  
        | *possible\_next* = *neighbors*  $\cup$  *visited* ;  
    **end**  
    **for** *x* in *possible\_next* **do**  
        | *probs*.append( $x \times x.outdegree$ ) **end**  
    **end**  
    *dstID* = random.choice(*probs*);  
    *cluster\_index* = index of *dstID* in *clusters*;  
    *nodeID* = *dstID* ;

**Algorithm 3:** overview of Smart Walk algorithm