Math 168 Project Report

Zhiqi Ma, Judy Zhu, Kunal Kulkarni
Department of Mathematics, University of California, Los Angeles
{zhiqima, judyz, kkulkarni}@g.ucla.edu

March 20, 2024

Abstract

In this report, we focus on exploring music similarity in guitar songs using network theory. It analyzes chord progressions by constructing chord networks from selected songs and employing various methods like eigenvalue centrality, earth mover's distance, and matrix multiplication for similarity comparison. Furthermore, the project delves into generating chord sequences through Markov chain-based random walks, influenced by the patterns in the analyzed data.

1 Introduction

In the past decades, Music Information Retrieval (MIR) has evolved into a broad research area, due to its numerous applications such as browsing, retrieving, and recommending music from digital databases. Among this field, one of the major open problems is the estimation and analysis of music similarity. Based on the intuition that similar melodies are often accompanied by a similar chord progression, our main goal for this project focus on comparing guitar songs' melodies from constructing a chord network. But before we present our own work, let's firstly discuss two previous related studies on harmonic similarity measures: Tonal Pitch Step Distance (TPSD) [3] and Chord Sequence Alignment System (CSAS) [2].

1.1 Related Works – What others did?

On the one hand, TPSD is based on Lerdahl's Tonal Pitch Space (TPS), a music-theoretical model that correlates with psychological findings and integrates pitches, chords, and keys into a cohesive framework [1]. The TPSD method efficiently measures the harmonic changes over time between sequences of chord labels by creating step functions that represent the change of harmonic distance in TPS over time. The researchers demonstrate the effectiveness of TPSD in retrieving similar jazz standards from the Real Book (one of the best-selling jazz books of all time), showing that it aligns with human intuitions about harmonic similarity. The study contributes a key-invariant, length-independent measure that allows for partial matching and is computationally efficient, grounded in cognitive modeling of tonality.

On the other hand, CSAS adapt alignment algorithms, traditionally used in string matching and bio-informatics, to compare chord sequences. The system accounts for

variations in chord progressions by considering deletions, insertions, and substitutions of chords. The key contribution is that this algorithm can handle multiple local transpositions, making the system invariant to key changes—a crucial aspect for accurately comparing chord progressions across different musical pieces. Through experiments on jazz standards and various genres, the system has shown effectiveness in music retrieval, notably surpassing existing methods in handling transpositions and providing a more discriminant tool for harmonic structure-based retrieval.

1.2 Motivation – What we will do?

While they differ in methodologies—one leveraging a cognitive musicological model and the other employing a computational alignment strategy—they share the common goal of enhancing the accuracy and robustness of music similarity assessments. However, neither of them (or other existing works on this field) have attempted to use network theory to model chord progressions and then compare chord similarity, which is what we are going to explore in this project.

We first select 6 songs consisting of only 4 guitar chords – C, G, F, Am – and create the chord sequence for each song. We then formulate a network where each node is a chord sequence like CCG and an edge represents the chord progression (e.g.: from CCG to CGF). Based on the network, we conduct several similarity comparison tasks such as calculating the node centrality and using the earth movers metric, corresponding with visualizations, analysis, and discussions. Even beyond that, we will also apply random walk to generate a chord sequence and then turn it into a piece of music melody to test the effectiveness of our model.

A rough outline of the paper is as follows. Section 2 will illustrate the detailed methodologies, theories, and algorithms. Afterwards, Section 3 contains major results, visualizations, and figures with analysis. Later, a discussion on the contribution, limitation, and possible further research directions are summarized in Section 4. Finally, we end by giving a conclusion in Section 5.

2 Methodology

2.1 Data Selection

The first step is to gather the data, i.e., chord sequences that each song uses. For simplicity reason, we looked for songs that only use four specific chords (C, Am, F, G) – possibly the four most popular and most common chords used in modern popular music. We selected 6 such songs, including: *All Too Well* by Taylor Swift, *Baby* by Justin Bieber, *Hey Soul Sister* by Train, *Riptide* by Vance Joy, *Story of My Life* by One Direction, and *Yellow* by Coldplay. We then represent the chord progression for each song as a string of chords with a format like CGFCCGFC...CGFCGFCGF for Yellow.

2.2 Data Processing

Next, based on the chord string sequence, we created a square adjacency matrix in which both rows and columns contain every possible combination of the four chords we are using in a 3 chord sequence. This ultimately resulted in 64 (4^3) possible combinations for both the rows and columns. The order of nodes is as follows:

CCC, CCG, CCF, CCAm, CGC, CGG, CGF, CGAm,
CFC, CFG, CFF, CFAm, CAmC, CAmG, CAmF, CAmAm,
GCC, GCG, GCF, GCAm, GGC, GGG, GGF, GGAm,
GFC, GFG, GFF, GFAm, GAmC, GAmG, GAmF, GAmAm,
FCC, FCG, FCF, FCAm, FGC, FGG, FGF, FGAm,
FFC, FFG, FFF, FFAm, FAmC, FAmG, FAmF, FAmAm,
AmCC, AmCG, AmCF, AmCAm, AmGC, AmGG, AmGF, AmGAm,
AmFC, AmFG, AmFF, AmFAm, AmAmC, AmAmG, AmAmF, AmAmAm

To construct the adjacency matrix that is representative of each song, we iterated through the string of chords and capture 3 chords, shift over 1 chord, and capture another 3 chords. The first 3 chords correspond to the row whereas the second 3 chords captured correspond to the column of the matrix. The connection is represented by adding a 1 to this (row, col) entry.

For instance, in chord string above for Yellow, the first 3-chord sequence we consider is 'CGF', and we move forward to get the next 3-chord sequence 'GFC', then 'FCC' and so on. In this case, since we move from 'CGF' to 'GFC', then we add a 1 on the entry (CGF, GFC) in our adjacency matrix. Notice that here we use 3-chord sequence for the aim to not only increase the complexity of our network that will be analyzed in next section but also to include as much information as possible from the chord progression sequence. Denote this adjacency matrix as matrix A, then it represents the connections between nodes in a directed graph. The element a_{ij} of the matrix is non-zero if there is an edge from node i to node j. Thus, we can use matrix A to help us count which 3-chord sequence is used the most frequently and to find if there is some pattern for certain chord to transit to other chords. For instance, Figure 1 represents the visualization of the network generated by the adjacency matrix of the song *All Too Well*.

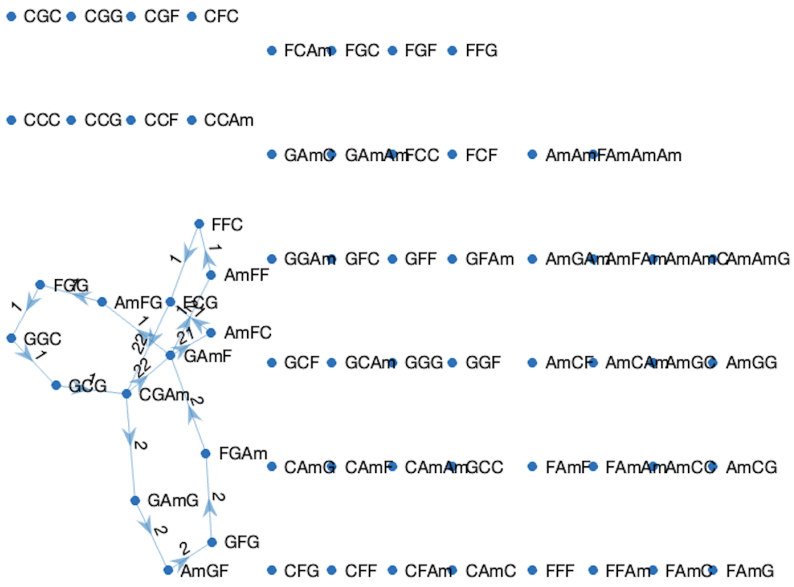

Figure 1: Network of All Too Well

Furthermore, given the adjacency matrix A, we also create the transition matrix. In the context of a Markov chain, matrix A can be seemed to represent the number of transitions or the weight of the transition from state i to state j, which can be converted to a transition matrix T, which is obtained by normalizing each row of the adjacency matrix to sum to 1. This normalization makes each row of T represent the probability distribution of transitioning from state i to any other state. Specifically, the element t_{ij} of the transition matrix is calculated as $t_{ij} = a_{ij} / \sum_k a_{ik}$, where the denominator is the sum of the elements of row i in the adjacency matrix. So for each row in the adjacency matrix, we first calculate the sum of the elements in that row and then divide each element in the row by the sum calculated in the previous step. These transition matrices will also be used in later analysis.

After calculation, both the adjacency matrix and transition matrix for all 6 songs are stored in csv files for further use.

2.3 Network Similarity Comparison

As aforementioned, the adjacency matrix we set up represents the information for a directed network, then the next step is to compare the similarity between different networks – compare the similarity of chord progression for different songs. It has come to our attention that, upon attempting to merge various networks into a single, large network, only one giant component emerges. This suggests that there is at least one node common to multiple songs. The question logically followed is what are those nodes and how important are these nodes in their original network?

During this subsection, we will illustrate 3 different methods for comparison: network node centrality (specifically the eigenvalue centrality and frequency), earth mover distance, and matrix multiplication.

2.3.1 Eigenvalue Centrality and Frequency Analysis

Eigenvalue Centrality involves the development of a method for ranking eigenvectors to assess the significance of nodes within a network. Utilizing this analysis provides information about the significance of the chord strings. This method leverages the versatility of pandas, a software library that is useful for conversion of data structures and data manipulation as well as numpy, a software library that performs computation on matrices. With pandas, a data frame is constructed and the initial step involves the extraction of chord combinations, a crucial precursor to subsequent ranking procedures. Subsequently, the string representations of chords are eliminated to enable numerical computation. Following data preprocessing, the data frame is converted into a numeric data type, paving the way for conversion into a numpy array. With this transformation complete, the eigenvalues and eigenvectors are computed. The key focus lies in identifying the maximum eigenvalues, whose corresponding indices are stored for subsequent use in ranking the associated eigenvectors. This metric serves as a valuable determinant of node centrality or popularity within the network. The methodology underscores a rigorous approach to eigenvalue analysis, offering insights into the relative importance of chord progressions within the created network.

Frequency Analysis, a far simpler approach of matrix analysis, involves ranking these nodes based on their total out degrees to other nodes. The function for this analysis adeptly captures chord information from the dataset stored in the CSV file, facilitating

the subsequent conversion of data into a numpy array as before. Leveraging the capabilities of numpy, the sum function is applied to compute row sums efficiently, yielding an array encapsulating the aggregate row-wise contributions. Subsequently, the row sums are systematically ranked based on their respective indices, facilitated by the argsort feature. Utilizing the indices derived from the row sum ranking, a secondary ranked list is constructed to delineate the hierarchy of chord sequences. This hierarchical arrangement provides valuable insights into the relative significance and prominence of chord progressions within the dataset. By employing a comprehensive approach that integrates data manipulation techniques with analytical tools, this methodology furnishes a refined understanding of chord progression dynamics and their implications in musical analysis.

Figures for both of these results are represented in Figure 3.

2.3.2 Earth Mover Distance

The Earth Mover's Distance (EMD), also known as the Wasserstein metric, is a measure of the distance between two probability distributions over a region D. It's often used in various applications like image retrieval, computer vision, and machine learning to quantify how different two distributions are. The idea is to calculate the minimum amount of work needed to transform one distribution into the other, where "work" is defined as the amount of distribution weight that must be moved, multiplied by the distance it has to be moved. As a way to measure how similar two sets of data are, EMD can thus be used to compare the similarity of chord progressions between two songs. Imagine you have two piles of sand representing the chord progressions of two different songs. The EMD tells you the least amount of effort you'd need to make one pile of sand look exactly like the other by moving the sand around. We use this article [4] for reference.

As for the implementation of using EMD for similarity comparison of the chord progression, we used the adjacency matrix calculated in Section 2.2. Notice that to calculate EMD, we need one-dimensional data, which is why we first need to transform our matrices using the NumPy array method "flatten": it takes a two-dimensional array and turns it into a one-dimensional array (a flat list). Imagine you have a grid of numbers, like a matrix, flattening it would mean putting all the rows together, end-to-end, into a single line of numbers. The second thing to do is to make sure these flat lists add up to 1 – each song uses all its chords with the total 'weight' being equal to 1 – by dividing each element in the list by the sum of all elements in the list. This is because EMD is particularly useful when the two distributions have the same overall mass. After this step, we can go on to measure how different each pair of songs is by calculating the EMD between their chord progression lists, using "wasserstein_distance" method. The calculation involves solving an optimization problem that finds the most efficient "flow" of mass from one distribution to the other.

After calculating the EMD between each pair of 6 songs, the output is a table where each cell tells us the EMD between two songs.

2.3.3 Matrix Multiplication

The third approach to assessing network similarities between selected songs involves utilizing the classic matrix multiplication. This method highlights the similarities of specific nodes used in each song. It is the measure of interaction or correlation between the corresponding patterns of the two songs, which is how the node (the sequence of three chords) is used. Non-zero elements can indicate that there is a certain type of connection

or pathway that is common in both songs. High values of certain entries in the multiplied matrix indicate the corresponding nodes in both songs have similar patterns of connectivity, while low values indicate a weaker relationship. Therefore, we could use it as a way to analyze the overall similarity between the two songs, in particular, it reveals which sequence of the chords of the songs is more similar or different. Additionally, we multiplied the matrix both ways since the matrix product depends on the order of multiplication, as it is inherently non-commutative. Therefore, we are measuring both how the elements of song A might transition into the elements of song B, as well as how elements of song B might transition into the elements of song A.

In order to do so, we first normalized our adjacency matrix by subtracting the μ (mean) and then dividing by the σ (square root of the variance) of each entry of the adjacency matrix calculated in Section 2.2. We then iterate over each unique pair of matrices and multiply them using the function `dot()` in `numpy` library built-in Python. Lastly, we convert the result back to a `DataFrame` and save the files into CSV files accordingly for further analysis.

3 Results & Analysis

In this section, we will present visualizations and analysis for those three similarity comparison methods discussed in Section 2.3. Code, matrix, figures, and other results can all be found via the Google Drive link.

3.1 Data Visualization For All Networks

Figure 2 presents the visualization of the song networks, constructed based on their adjacency matrices as described in Section 2.2. These visualizations were generated using Matlab. The networks depicted are fully connected in a single giant component, illustrating the progression of one chord to another within each song. Along the periphery of these network structures, nodes are displayed which represent chords that were not utilized in the respective songs.

Figure : Song Networks

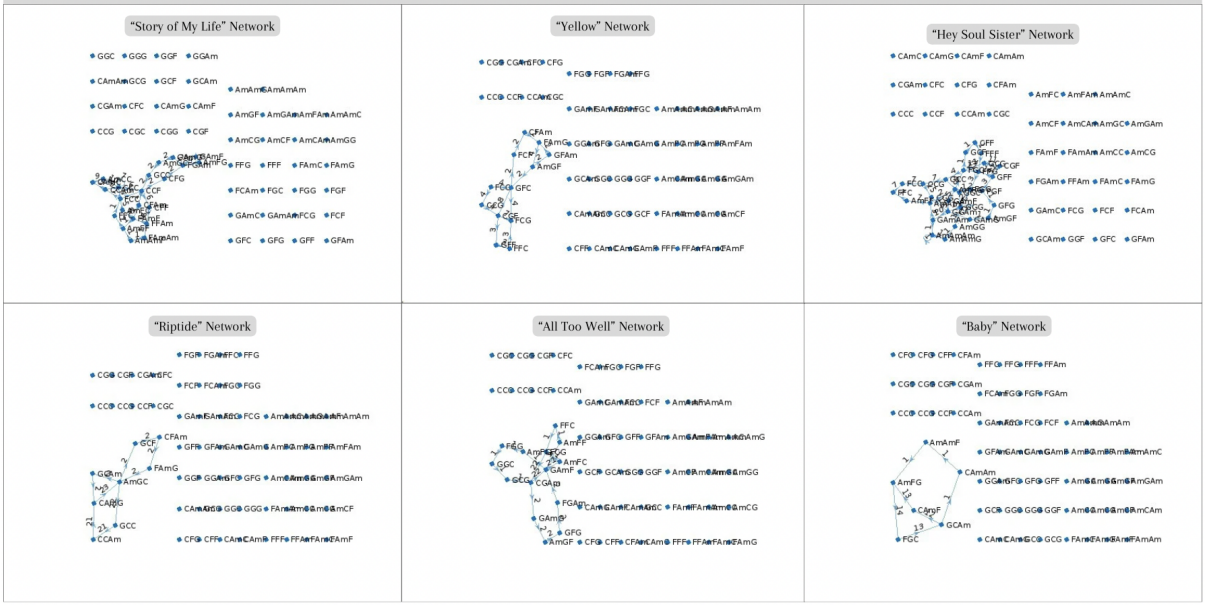

Figure 2: Networks of All Songs

3.2 Eigenvalue Centrality and Frequency Analysis

Figure : Eigenvalue Centrality Rankings

"Story of My Life"	"Yellow"	"Hey Soul Sister"	"Riptide"	"All Too Well"	"Baby"
Rank: 1 Chords: CAmG	Rank: 1 Chords: FCG	Rank 1: Chords: GFF	Rank: 1 Chords: AmGCF	Rank 1: Chords: FCG	Rank 1: Chords: CAmF
Rank: 2 Chords: CCAm	Rank: 2 Chords: CGF	Rank 2: Chords: CGF	Rank: 2 Chords: GCF	Rank 2: Chords: CGAm	Rank 2: Chords: AmFG
Rank: 3 Chords: AmGC	Rank: 3 Chords: GFC	Rank 3: Chords: GCG	Rank: 3 Chords: CGFC	Rank 3: Chords: AmFC	Rank 3: Chords: FGC
Rank: 4 Chords: GCC	Rank: 4 Chords: CCG	Rank 4: Chords: FFG	Rank: 4 Chords: FC	Rank 4: Chords: GAmF	Rank 4: Chords: GCAm
Rank: 5 Chords: GCAm	Rank: 5 Chords: FFC	Rank 5: Chords: FGC	Rank: 5 Chords: CFC	Rank 5: Chords: FGAm	Rank 5: Chords: AmAmF
Rank: 6 Chords: FAmG	Rank: 6 Chords: FCC	Rank 6: Chords: AmFG	Rank: 6 Chords: FCFC	Rank 6: Chords: FFC	Rank 6: Chords: CAmAm
Rank: 7 Chords: CFAm	Rank: 7 Chords: AmGF	Rank 7: Chords: FGF	Rank: 7 Chords: FCGC	Rank 7: Chords: GCG	Rank 7: Chords: CCC
Rank: 8 Chords: GCF	Rank: 8 Chords: GFF	Rank 8: Chords: CCG	Rank: 8 Chords: GCF	Rank 8: Chords: GFG	Rank 8: Chords: FAmG
Rank: 9 Chords: FFAm	Rank: 9 Chords: FAmG	Rank 9: Chords: GAmF	Rank: 9 Chords: CCGC	Rank 9: Chords: AmFF	Rank 9: Chords: FAmC
Rank: 10 Chords: FFG	Rank: 10 Chords: CFAm	Rank 10: Chords: CGG	Rank: 10 Chords: CGF	Rank 10: Chords: GGC	Rank 10: Chords: FFAm

Figure : Frequency Rankings

"Story of My Life"	"Yellow"	"Hey Soul Sister"	"Riptide"	"All Too Well"	"Baby"
Rank 1: CCC (Sum: 21)	Rank 1: CGF (Sum: 11)	Rank 1: GGAm (Sum: 16)	Rank 1: AmGC (Sum: 26)	Rank 1: CGAm (Sum: 24)	Rank 1: AmFG (Sum: 14)
Rank 2: AmCC (Sum: 10)	Rank 2: GFC (Sum: 10)	Rank 2: FGC (Sum: 16)	Rank 2: CAmG (Sum: 23)	Rank 2: GAmF (Sum: 23)	Rank 2: CAmF (Sum: 13)
Rank 3: CCAm (Sum: 9)	Rank 3: FCG (Sum: 7)	Rank 3: CCG (Sum: 15)	Rank 3: CCAm (Sum: 21)	Rank 3: FCG (Sum: 22)	Rank 3: FGC (Sum: 13)
Rank 4: CAmC (Sum: 9)	Rank 4: CCG (Sum: 4)	Rank 4: CGG (Sum: 15)	Rank 4: GCC (Sum: 21)	Rank 4: AmFC (Sum: 21)	Rank 4: GCAm (Sum: 13)
Rank 5: CCF (Sum: 9)	Rank 5: FAmG (Sum: 4)	Rank 5: FFG (Sum: 15)	Rank 5: FAmG (Sum: 2)	Rank 5: GFG (Sum: 2)	Rank 5: CAmAm (Sum: 1)
Rank 6: FFAm (Sum: 6)	Rank 6: FCC (Sum: 4)	Rank 6: GFF (Sum: 14)	Rank 6: GCF (Sum: 2)	Rank 6: GAmG (Sum: 2)	Rank 6: AmAmF (Sum: 1)
Rank 7: FGAm (Sum: 6)	Rank 7: AmGF (Sum: 4)	Rank 7: CGF (Sum: 11)	Rank 7: GCAm (Sum: 2)	Rank 7: FGAm (Sum: 2)	
Rank 8: FAmF (Sum: 6)	Rank 8: GFF (Sum: 3)	Rank 8: GCG (Sum: 11)	Rank 8: CFAm (Sum: 2)	Rank 8: AmGF (Sum: 2)	
Rank 9: CFF (Sum: 6)	Rank 9: FFC (Sum: 3)	Rank 9: GAmAm (Sum: 10)		Rank 9: FFC (Sum: 1)	
Rank 10: FCC (Sum: 6)	Rank 10: CFAm (Sum: 2)	Rank 10: AmAmF (Sum: 9)		Rank 10: FGG (Sum: 1)	

Figure 3: Eigenvalue and Frequency Rankings

Figure 3 represents eigenvalue centrality and frequency rankings of the chord sequences for each song. Certain songs such as "Riptide" include fewer rankings in the frequency rankings because entries of the rankings that are of sum zero are disregarded. Upon observation of these tables, it becomes clear that most of the entries share two or all characters

(i.e. in "Yellow", when observing the first two entries of the Eigenvalue Centrality Rankings (FCG, CGF) and Frequency Rankings (CGF, GFC)). This is likely a consequence of construction of the adjacency matrices (in which 3 chords are captured before shifting over one chord and capturing 3 nodes again). There are however larger implications about the repetition of specific chord sequences in these songs. The repetitiveness of certain chords, especially among the higher ranked chord sequences, is suggestive of the melodic nature of these songs since chord progressions that are used throughout the song reinforce both the centrality and frequency of nodes in the network.

While there is an intersection between the results generated by eigenvalue centrality rankings and frequency rankings, there is an equally clear distinction in the results generated by these analyses. Primarily, node importance drastically affects the rankings of chords, ensuring that nodes connected with other popular nodes are favored in these eigenvalue centrality rankings (i.e. GFF, "Hey Soul Sister") whereas chord sequences that often traverse to any node in the networks often are favored in frequency rankings (i.e. GGAm, "Hey Soul Sister). As such, eigenvalue centrality is useful in scenarios where one is attempting to search for a chord sequence that can be arranged between two other chord sequences. On the contrary, frequency can be a useful metric for determining a chord sequence that may precede a specific sequence. It is however, important to acknowledge that these sequences can be harmonically arranged between a select few sequences, even in a network of four chords.

3.3 Earth Mover Distance

	1	2	3	4	5	6
1	0	0.000058	0.000208	0.000251	0.000263	0.000033
2	0.000058	0	0.000261	0.000309	0.000321	0.000033
3	0.000208	0.000261	0	0.000096	0.000104	0.000229
4	0.000251	0.000309	0.000096	0	0.000043	0.000276
5	0.000263	0.000321	0.000104	0.000043	0	0.000288
6	0.000033	0.000033	0.000229	0.000276	0.000288	0

Figure 4: EMD Matrix

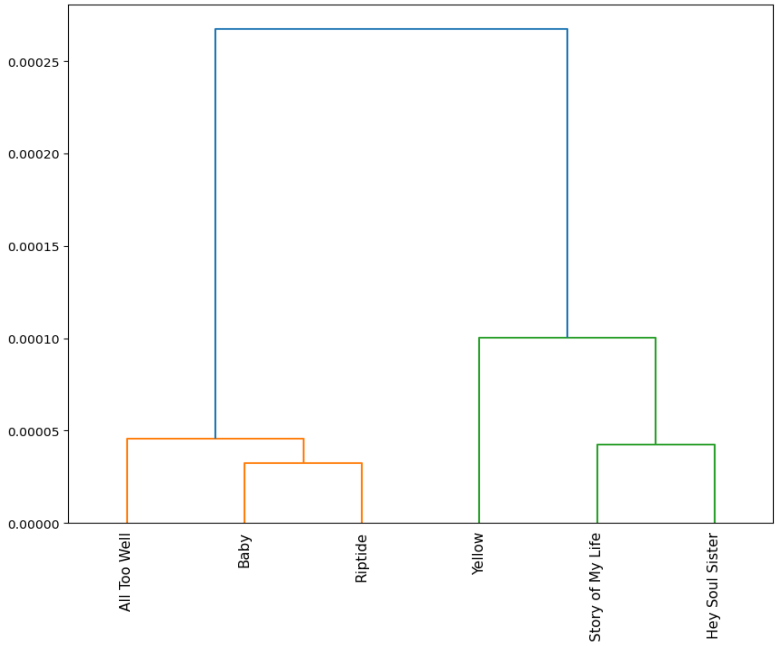

Figure 5: Dendrogram

The matrix in Figure 4 represents the Earth Mover's Distance (EMD) between the chord progressions of six different songs. Notice that all values in the matrix are quite small. That's because during the data pre-processing part before EMD, we divide all original data by the sum of all values in the original 64*64 matrix, and the later EMD calculations cause the values to be even smaller. Diagonal values are zero, which is expected since it compares each song with itself. Looking at off-diagonal values, smaller numbers indicate more similar chord progressions. For instance, the EMD between songs 1 and 2 is particularly small (0.000058), suggesting they have very similar chord structures. On the other hand, the EMD between songs 3 and 6 is 0.000229, implying a greater difference in their chord progressions. Therefore, songs with the smallest EMD values are more stylistically close to each other than those with larger values, providing insights into the musical and structural similarities between these pieces.

Based on the EMD result matrix, we also build up a dendrogram in Figure 5, which is a visual representation of the hierarchical clustering analysis. The dendrogram plots out the EMD values to show the relative similarity between the songs. In this tree-like diagram, the root represents the point at which all songs are considered in a single cluster, each leaf represents a song, and the branches show how songs are grouped based on the EMD. The height at which two branches merge represents the EMD between clusters: the lower the merge point, the more similar the songs are. By examining the dendrogram, we can easily identify which songs are similar to each other in terms of chord progression and how they are grouped into larger clusters. This gives an insightful visualization of the relationship between songs, providing a hierarchical structure of musical similarity that can be interpreted later and offering a clear visual understanding of how songs relate based on their chord progressions.

3.4 Matrix Multiplication

We generated all possible pairs of matrix multiplication, which resulted in 30 distinct matrices. The first thing we noticed about the result of matrix multiplication is different

for the song A transitions into song B and song B transitions into song A, which is what we expected since matrix multiplication is on-commutative. Furthermore, a non-zero value at position (i, j) in the product matrix indicates that the i-th element of the first song relates to the j-th element of the second song through some pathway in their respective network structures. High positive values indicate a strong and direct connection, while negative values indicate an inverse relationship.

We then take the result in Section 3.2 and compare the nodes in the most similar songs, which is *Baby* and *Riptide*, as well as *Story of My Life* and *Hey Soul Sister*.

3.4.1 *Baby* and *Riptide*

Matrix going from *Baby* and *Riptide* has 384 non-zero entries, which means that there are a variety of interactions when the structure of *Baby* influences *Riptide*. Matrix going from *Riptide* and *Baby* has 512 non-zero entries, suggesting even more points of interaction when the process is reversed. This indicates that *Riptide* has a greater impact on *Baby* than the other way around.

Additionally, the strongest positive connection from *Baby* to *Riptide* is approximately 61.13, which suggests a very strong link from the progression GAmC in *Baby* to the progression CAmG in *Riptide*. The strongest negative connection is thus between nodes GAmC and FAmC, with a value of approximately -2.87. On the other way around, the strongest positive connection from *Riptide* to *Baby* is approximately 4.62, suggesting a strong link from *Riptide*'s FAmC to *Baby*'s CAmAm progression, and the strongest negative connection is around -2.34, suggesting that FAmC in *Riptide* diverges significantly from *Baby*'s FFAm progression.

3.4.2 *Story of My Life* and *Hey Soul Sister*

We then continue with a similar analysis on *Story of My Life* and *Hey Soul Sister*.

Matrix from *Story of My Life* to *Hey Soul Sister* has 1323 non-zero entries, indicating a variety of interactions. The strongest positive connection is approximately 62.50 between nodes AmFG and GAmF, and the strongest negative connection is approximately -3.80 between nodes GCAm and FGG.

Matrix from *Hey Soul Sister* to *Story of My Life* has 1827 non-zero entries, suggesting a more substantial impact of *Hey Soul Sister* on *Story of My Life*. The strongest positive connection is approximately 62.38 between nodes FFAm and GCF, and the strongest negative connection is approximately -3.25 between nodes CCC and FGC.

These findings suggest a complex interplay between the structures of the two songs, with *Hey Soul Sister* potentially exerting a greater influence on *Story of My Life*.

3.5 Chord Sequence Generator

As a culmination of our previous work, we try to generate a sequence of chords based on the transition matrix calculated before, using knowledge related to markov chains & random walk.

Here we first sum up all 6 transition matrices for each of the 6 songs and then take the average. We then design the 'load_and_normalize_matrix' function to process the average transition matrix. This function reads the matrix into a DataFrame, normalizing and ensuring that each row, which corresponds to a particular chord, is normalized so that the sum of probabilities of transitioning to any other chord is 1. If a chord doesn't lead

to any other chord (resulting in a row sum of 0), the function adjusts this by allowing the chord to transition to itself, ensuring there's always a valid next step in the progression.

Next, we use 'simulate_markov_walk' function that takes this normalized transition matrix and generates a sequence of chords by simulating a Markov chain. We also need to pick a node (a 3-chord-sequence) as a starting point. Technically speaking, any node can be the starting point, but here we choose 'FCG' as a illustration since FCG seems to rank the 1st twice in the eigenvalue centrality rankings, as shown in Figure 3. Afterwards, this function then uses the probabilities in the matrix to pick the next chord in the sequence, repeating this for a specified number of steps. The result is a randomly generated, but statistically informed, chord progression that reflects the patterns encoded in the original transition matrix. This simulated walk mimics the process of composing music by stringing together chords in a way that is likely based on the input data. For example, if we take step size to be 30, starting from the node 'FCG', then we get the following chord sequence: 'FCGAmFCGAmFCCGGAmFGAmFCGFCFAmFCGAmFCGAmG'.

4 Discussion & Future Work

Implication: We entered this project with the expectation that there are certain connections between musicology and mathematics and were looking to model this relationship on a small scale. Our initial approach did not set specific expectations for the type of network we would construct based solely on music, the results it would yield, or the music that could be generated from the data. However, we anticipated uncovering the centrality of the most commonly used chord sequences through the network analysis, leveraging what we had learned over the quarter and supplemented by our own research online. Ultimately, we succeeded in translating chords into network models, calculating the corresponding adjacency and transition matrices. This enabled us to analyze the nodes within the network and embark on generating random walks to compose new music.

Limitation: One of the most obvious limitations of our project is that we only take 4 chords into consideration, simplifying what the reality is – generally most songs include much more than only 4 chords and there are lots of chord variations. Another limitation is that we only work with 6 songs due to the time constraint, so our results may not be too generalize. Plus, in this project we focus on chord progression similarity comparison, but actually there are numerous other factors that can affect music similarity, such as drumming pattern, rhythm and tempo, arrangement and structure, as well as the length of the songs. Moreover, what generated by the chord sequence generator is also not ideal.

Future Work: Possible future directions of research can explore aforementioned aspects not included in our project. Specifically, more songs as well as more chords can be included to generalize the network model, and other factors that can affect the music similarity between two songs can also be investigated. After all these works, the performance of the chord sequence generator may be better or we can even include other features like strumming pattern, rhythm, and tempo into the generator to make the results more reasonable and interpretable. Lastly, to generate better music, we could employ machine learning techniques such as diffusion model or LLM to better predict the random walk.

5 Conclusion

In conclusion, the project encapsulates a compelling intersection of musicology and mathematics, aiming to explore the realm of music similarity through chord progression analysis. Synthesizing ideas of music theory and complex techniques from network analyses, our team delved into the construction of chord networks and comparison of chord sequences for six guitar songs. Through meticulous methodology and analysis, valuable insights into the structural and stylistic similarities between these songs were uncovered.

The investigation led to three distinct methods of similarity comparison: eigenvalue centrality and frequency analysis, Earth Mover's Distance (EMD), and matrix multiplication. Eigenvalue centrality and frequency analyses led to effective conclusions about how chord sequences can be arranged into music. Earth Mover's Distance analysis led to insightful conclusions about song similarity based on comparisons between distributions of flattened representations of data for these songs. Matrix multiplication was especially useful in determining song correlation by comparing chord sequence connections between two songs. Specifically, this analysis recognized similarity between "Baby" and "Riptide" as well as "Story of My Life" and "Hey Soul Sister"

Each method offered unique perspectives on the relationships between chord progressions, shedding light on the underlying patterns and connections within the musical compositions. From visualizations of chord networks to hierarchical clustering analysis, the data was visualized and interpreted to discern meaningful patterns and relationships.

Furthermore, the exploration extended to the realm of chord sequence generation, where Markov chain-based random walks were employed to create new chord progressions inspired by the analyzed data. This innovative approach provided a glimpse into the potential applications of computational techniques in music composition and generation, which specifically may be more optimal in the generation of melodies.

While the project yielded promising results, it is not without limitations. The team acknowledges the simplifications made in focusing on a limited set of chords and songs, as well as the exclusion of other musical elements that contribute to overall music similarity. Nonetheless, the findings lay the groundwork for future research directions, including the incorporation of additional musical features and the expansion of the dataset for more comprehensive analyses.

In essence, the project underscores the rich interrelation between music and mathematics, demonstrating the potential of computational methods in unraveling the complexities of musical structure and similarity. By bridging disciplines and leveraging innovative techniques, this project reinforces the potential for the use of mathematical tools in music theory

6 Contribution

Zhiqi Ma: Worked on the idea and structure of the project, selected the related works and data to be used, and collaborated with Judy on the coding part. Worked on the Introduction, Data Selection & Processing, EMD, and discussion parts of the report, and the corresponding slides for the presentation.

Judy Zhu: Worked on the group work of the approach of modeling music into the network and ways of analysis. Worked on Matlab and Python coding for data processing, Earth Mover Distance, and Matrix Multiplication. Worked on the Matrix Multiplication

parts of the write up as well as the general discussion, and the corresponding slides for the presentation.

Kunal Kulkarni: Contributed to the conceptualization of the project and how music data may be collected and represented. Produced methodology for eigenvalue centrality and frequency rankings using python to analyze music data. Worked on Data Selection, eigenvalue centrality and frequency analysis, designing figures to represent networks and analyses, as well as conclusions drawn from the research (for the paper and presentation).

References

[1] F. Lerdahl, *Tonal Pitch Space*. Oxford University Press, 2001.

[2] P. Hanna, M. Robine, and T. Rocher, "An Alignment Based System for Chord Sequence Retrieval," pp. 101-104, 2009, doi: 10.1145/1555400.1555417.

[3] W. de Haas, R. Veltkamp, and F. Wiering, "Tonal Pitch Step Distance: A Similarity Measure for Chord Progressions," in *Acta Psychologica - ACTA PSYCHOL*, pp. 51-56, 2008.

[4] Y. Rubner, "The Earth Mover's Distance," 2000. [Online]. Available: `https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/RUBNER/emd.htm`.