UTILIZING KALE AND CHORD STRUCTURE IN CONTENT-BASED MUSIC RECOMMENDATION

An Undergraduate Research Scholars Thesis

by

GREGORY KRUPIT

Submitted to the Undergraduate Research Scholars program
Texas A&M University
in partial fulfillment of the requirements for the degree of

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Tiffani Williams

May 2017

Major: Computer Science

ABSTRACT

Utilizing kALE and Chord Structure in Content-Based Music Recommendation

Gregory Krupit
Department of Computer Science
Texas A&M University

Research Advisor: Dr. Tiffani Williams
Department of Computer Science
Texas A&M University

With the popularity of digital entertainment applications such as Pandora, Spotify, Amazon Music, etc., comes the need for sophisticated music recommendation algorithms. Many algorithms depend on collaborative filtering, determining relevant song suggestions for a particular user by analyzing similar tastes in the rest of the user base. This method comes with the "cold start" problem, such that a system does not have enough data on a new user to generate meaningful suggestions. A solution to this problem is a content-based method, utilizing the structure and features of the music itself to determine similar songs rather than user-based data. By using audio analysis tools and performing wavelet analysis on songs to extract relevant acoustic features (chord progression), we will define a new measure of similarity between songs. We show that using similarity of chord structure as a basis for content-based music recommendation is a quality metric that can be introduced into existing music recommendation applications.

DEDICATION

To a kit and a pup.

ACKNOWLEDGMENTS

I would like to extend a very warm thanks to my undergraduate research advisor, Dr. Tiffani Williams, for not only providing me with the tools, guidance, and motivation I needed to complete this research, but also for providing inspiration and direction in the realm of a topic that is important to both of us: education. I would also like to thank my roommates, Brian Burrows and Renee Swischuk, for leading by example when it comes to staying committed to research goals.

I would also like to thank Texas A&M's Undergraduate Research Scholars Program for providing the structure I needed to explore an interesting piece of the intersection of computer science and music, as well as Texas A&M's Computer Science Department for providing the opportunity to complete an undergraduate thesis as my senior project.

NOMENCLATURE

MIR	Music Information Retrieval
kALE	k-gram AnaLysis Engine

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

Current music recommendation algorithms subscribe to three methods: collaborative filtering, content-based recommendation, or a mixture of the two [1]. Collaborative filtering is a method in which a system recommends music to a user based on similar users. This method yields an issue known as the "cold start" problem, where new users are provided poor recommendations due to insufficient data. This provides motivation that improving the second method, content-based recommendation, is an important pursuit. Music Information Retrieval (MIR) is a field dedicated to techniques for gathering useful information about songs and artists. Within the context of content-based music recommendation, MIR can be used to extract song metadata (genre, artist, album, composition year, etc.) and social media based information (comments, tags, reviews, etc.) in their determination of similar songs. An alternate form of MIR focuses on acoustic analyses of songs, using features such as beat, tempo, pitch, instrumentation, mood, etc. to determine similar songs of potential interest [1]. Utilizing acoustic features of a song in lieu of user data and/or song metadata is one solution to the cold start problem and is where this research will hold its focus.

When determining song suggestions similar to a given set of songs, one feature currently not used is the chord structure of the song. In an analysis of chords and progressions used in some of history's most popular songs, the distribution of chords and progressions found is significantly skewed [2], providing motivation that similar chords and chord structures may be a key indicator of suitable song recommendations. Thus, we will investigate the use of chord structure when determining song similarity in order to generate suggestions. Figure 1.1 illustrates the flow of our entire system beginning with set of MP3's and resulting in an MP3 player that uses our recommendation system.

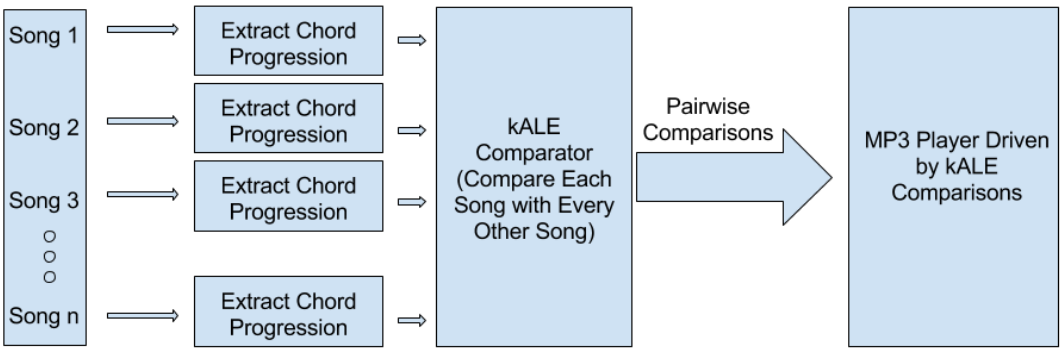

Figure 1.1: Overview of the transition from a list of songs to an MP3 player that provides recommendations based on comparison of those songs.

We have attempted to improve on current methods of finding the chord progression of an arbitrary audio recording with a novel method using Fast Fourier transforms and frequency analysis. In addition, we provide a machine learning method for determining the key of a given audio recording by providing a chord histogram as a feature vector. Once chord progressions for each MP3 in the library are determined, pairwise comparisons are performed between chord progressions using an original distance measurement, k-gram AnaLysis Engine (kALE).

We also compare how providing song recommendations to users using kALE compares with providing users with a random sequence of songs. By analyzing trends in the ratings

that users gave songs during experiments, we show that kALE performs better in providing quality recommendations than randomnly generating songs.

2. RESEARCH METHODS

We extracted acoustc features from and used kALE on 74 indie pop songs found on NoiseTrade.com, a site providing free and legal MP3's from independent artists. We built an application that tests the effectiveness of kALE regarding the quality of the application's recommendations; the application asks subjects to listen to a song, collects a rating the song and feedback about the song, and generates a new song based on the subject's previous rating. Using quantitative analysis of successive ratings and qualitative analysis of subjects' written feedback, we determined that kALE provides better recommendations than when randomly ordered songs are presented to subjects.

3. MUSIC THEORY BACKGROUND

Basic knowledge of music theory is required to understand certain aspects of this work. Therefore, we present introductions to the music theory topics involved with the methods presented in this paper.

3.1 Music

All sound is made up of individual frequencies played in certain grouping and in certain sequences. Music is the organization of the types of groupings and sequences of frequencies that can occur. For our purpose, we define music as a set of rules that define which groupings of frequencies are more likely to occur together, as well as which sequences of frequencies are likely to occur.

3.2 Notes

Sound waves tend to follow a repeating, oscillating pattern, alternating between high and low. Sound waves that we hear in popular music follow a predictable, repeating pattern. Therefore, we can classify different waves based on wave properties. We define the wavelength of a wave as the distance between adjacent peaks. We define the period of a wave to be the length of one full oscillation of high to low. A cycle is defined as the the distance between a point in a wave period and the same point the next occurring period. Each of these properties helps define frequency, which is the number of periods that occur in one second. As illustrated in figure 3.1, a wave with a small period has a higher frequency than a wave with a large period. We commonly used the unit Hertz (Hz) as the unit of frequency.

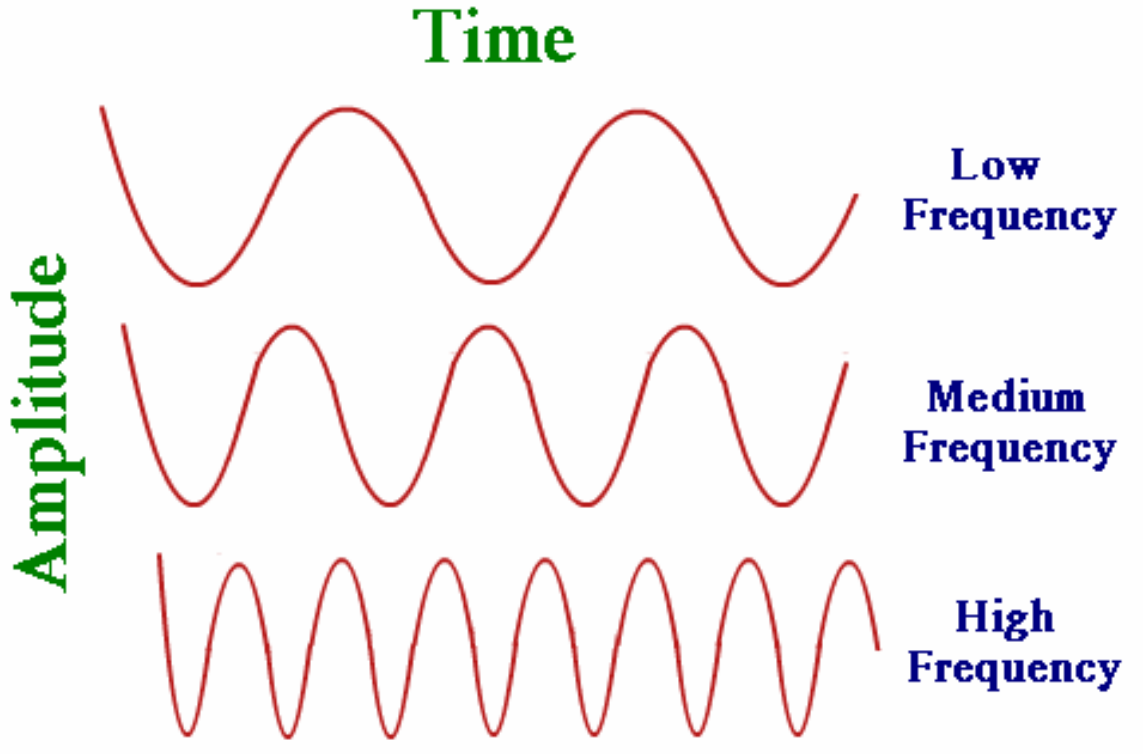

Figure 3.1: Illusration of the difference between high frequency wave, medium frequency, and low frequency waves.

In Western music, we classify frequencies using a logarithmic scale: if we thus, if we take an arbitrary frequency f, we categorize $f, 2 \cdot f, 2^2 \cdot f, \cdots, 2^n \cdot f$ as the same frequency class. We call these frequency classes notes.

In Western music, there are 12 important notes. Each note has a letter (A,B,C,D,E,F,G) and an optional accidental (♯=sharp or ♭=flat). (Note: there are more accidentals than ♯ and ♭, but are not required for the understanding of this work.)

Humans can hear in the interval [20 Hz, 20,000 Hz]. Thus, we define the notes' assigned frequencies of each note starting around that range:

Notes	Frequency	Wavelength
C	16.35	2109.89
C♯/D♭	17.32	1991.47
D	18.35	1879.69
D♯/E♭	19.45	1774.20
E	20.60	1674.62
F	21.83	1580.63
F♯/G♭	23.12	1491.91
G	24.50	1408.18
G♯/A♭	25.96	1329.14
A	27.50	1254.55
A♯/B♭	29.14	1184.13
B	30.87	1117.67
C	32.70	1054.94

Table 3.1: Equivalent notes and their corresponding frequencies and wavelengths.

If we move down table 3.1 by one row, we move up the note by one half step. If we move down the table by two rows, we move up in note by one whole step. Similarly, moving up the table by one row moves the note down one half step, while moving up the table by two rows moves the note down by two half steps, or one whole step. The purpose of ♯ is to take the letter that precedes it and move the note up one half step. The purpose

of ♭ is to move the note down one half step.

3.3 Scales

We define a scale as a sequence of notes moving in the same direction. The frequencies get higher in an ascending scale while the frequencies get lower in a descending scale. We define the two important scales for this work: major and minor. Major and minor scales each start and end with the same note, and contain 7 unique notes. To build an ascending major scale starting from some note, we take the following sequence of steps:

1. whole step
2. whole step
3. half step
4. whole step
5. whole step
6. whole step
7. half step

For example, a C major scale starts with C and takes the following steps:

1. whole step $\rightarrow$ D
2. whole step $\rightarrow$ E
3. half step $\rightarrow$ F
4. whole step $\rightarrow$ G
5. whole step $\rightarrow$ A

6. whole step $\rightarrow$ B

7. half step $\rightarrow$ C

Thus, the notes in a C major scale are C,D,E,F,G,A,B,C.

To build an ascending minor scale starting from some note, we take the following sequence of steps:

1. whole step

2. half step

3. whole step

4. whole step

5. half step

6. whole step

7. whole step

For example, a C minor scale starts with C and takes the following steps:

1. whole step $\rightarrow$ D

2. half step $\rightarrow$ E$\flat$

3. whole step $\rightarrow$ F

4. whole step $\rightarrow$ G

5. half step $\rightarrow$ A$\flat$

6. whole step $\rightarrow$ B$\flat$

7. whole step $\rightarrow$ C

Thus, the notes in a C minor scale are C,D,E♭,F,G,A♭,B♭,C.

3.4 Chords

We define a chord as two or more notes played at the same time. In this work, we only consider two types of chords: major and minor. To determine which chord a set of notes creates, we look at the triad created by those notes. A triad is a set of three unique notes: the root note is the name of the chord, the middle note is two notes higher. and the top note is two notes higher than the middle note. If a chord is major or minor, it always takes 7 half steps to move from the root to the top note. In a major chord, it takes four half steps to move from the root note to the middle note. In a minor chord, it takes three half steps to move from the root note to the middle note.

To build a C major chord, we define our root as C, move up four half steps to E, and move 7 half steps from C to reach the top note, G. Thus, the notes C,E, and G make up a C major chord. To build an B minor chord, we define our root as B, move up three half steps to D, and move 7 half steps from B to reach the top note, F♯. Thus, the notes in a B minor chord are B,D, and F♯.

3.5 Keys

The key of a song defines a template for which notes are generally used in that song. We label each with a letter, optional accidental, and either major or minor. We identify which key a song is in by the the accidentals of the notes used. For example, a C major scale is C,D,E,F,G,A,B,C. Because there are no accidentals used in the C major scale, we define C major to be the key that has no accidentals. Similarly, if we write out an A minor scale, we see A,B,C,D,E,F,G,A, which also has no accidentals.

An A major scale is as follows: A, B, C♯, D,E,F♯,G♯,A. Thus, the major key with 3 sharps (C♯, G♯, F♯) is A major.

We use the following notation to write major and minor chords: a note with no label following is assumed to be major, while a note with a lowercase 'm' following it is minor. For example, C represents C major, and Cm represents C minor.

3.6 Chord Structure/Chord Progressions in a Specific Key

Using the key of a song (the starting note for the scale and the set of accidentals in that key), we can determine which set of chords belongs in that key. For each note in the scale of a key, build a triad with the note as the root. These triads make up the chords to be used in that key. For example, looking at the F major scale, we define a triad for each of F,G,A,B♭,C,D,E, using only the notes that exist in that scale:

1. F,A,C → F
2. G,B♭,D → Gm
3. A,C,E → Am
4. B♭,D,F → B♭
5. C,E,G → C
6. D,F,A → Dm
7. E,G,B♭ → E diminished (our method does not distinguish between minor and diminished chords)

Thus, the chords used in the key of F major are F, Gm, Am, B♭, C, Dm, and E diminished.

A chord progression is a sequence of chords. Though there are exceptions, the most popular chords used in a chord progression are those defined by a specific key's triads. Therefore, the most likely chords found in a song in F major are F, Gm, Am, B♭, C, Dm, and E diminished.

3.7 Generalized Chord Progressions

We can generalize the chords used in a chord progression by analyzing the individual chords' relationship to the key of the song. For example, in a song in C major, we can generalize the chord progression C, G, Am, F, Dm, G, C by noting that C is the first note in the C major scale, G is the fifth note, A is the sixth note, and so on. We label these relationships with roman numerals. Thus

$$C, G, Am, F, Dm, G, C$$

becomes

$$I, V, vi, VI, ii, V, I$$

(Note that we used upper case letters for major chords and lower case letters for minor chords.)

Because the roman numerals identify relationships to a key instead of the chords themselves, we can rewrite that chord progression in any key by identifying the sequence of chords that have the same relationships with the new key as the roman numerals prescribe. To write the same chord progression,

$$I, V, vi, VI, ii, V, I$$

in G major, we write

$$G, D, Em, C, Am, D, G$$

Thus, even though C, G, Am, F, Dm, G, C and G, D, Em, C, Am, D, G are different chord progressions, they represent the same generalized chord progression I, V, vi, VI, ii, V, I.

4. EXTRACTING RELEVANT DATA FROM MP3/WAV FILES

4.1 Normalizing Audio Input

The library of music used to determine the utility of kALE contains in songs in MP3 format. The methods used to extract relevant metadata, however, require the WAV file format as input. Therefore, before analyzing an MP3, we convert it into a WAV file for use in our algorithms.

We chose to convert a library of MP3 files to WAV format rather than beginning with a library of WAV files. Because the MP3 format is compressed, it removes some of the noise that occurs in an uncompressed file format such as WAV. By inspection of the extracted chord progressions, the method we use to extract chord progression from a waveform performed better (more accurate chord progressions) with a compressed waveform than its uncompressed counterpart. Therefore, we chose to use a library of MP3's to exploit this benefit.

4.2 Structure of MP3/WAV File Formats

Metadata included in an MP3 file includes information about artist, album, genre, audio duration, and the sampling rate (samples/second) of the MP3. The sampling rate of a recording defines how many samples per second should be sent to an audio device. The sampling rate of all recordings in our library is 41,000 Hz, or 41,000 samples/second. In our method, sampling rate is the only piece of MP3 metadata we use.

The audio portion of a WAV file consists of an array of pairs (samples): the values in a given pair represent the frequency to be played in the left and right speakers respectively. When the samples in the array are played sequentially at the intended sampling rate, we hear the audio recording as it intended to be heard. For our method, we only use the left band of frequencies to determine the chord progression.

4.3 BPM Extraction

To determine the BPM of an arbitrary WAV file, we use the method presented by George Tzanetakis. [3]

4.4 Chord Progression Extraction

To determine the chord progression of an arbitrary MP3, we perform a harmonic analysis on discrete frames of a recording, where each frame consists of a constant number of samples (frequencies).

4.4.1 Subdivision of MP3's into Sub-Beats

Because our chord finding method splits up an MP3 into discrete timesteps (frames) to analyze, we use a given recording's extracted BPM to create frames according to the natural rhythm of the song. To determine how many samples belong in each frame to analyze, we use the following equation:

$$samples\_per\_frame = \left\lfloor 60 \cdot \frac{sampling\_rate}{BPM \cdot 4} \right\rfloor \tag{4.1}$$

which is equivalent to the number of samples in each quarter of a beat. By analyzing a recording one quarter beat at a time, we allow for more complex chord progressions to be extracted, specifically those that use syncopation (songs in which the change in chord does not always fall exactly on the beat).

4.4.2 Use of Chromagrams in Chord Detection

Once a waveform is split into discrete sections, each section is analyzed with the use of chromagrams and chord detection. For each partition of the waveform, the intensities of each of the twelve notes are discovered through the use of a fast fourier transform and harmonic analysis. These intensities are stored in a chromagram. Thus, once a chroma-

gram for each partition of the waveform is generated, each chromagram is analyzed for the likeliest chord. For ease of comparison, we only classify chords as major or minor, excluding all other chord forms.

4.5 Key Extraction

Key extraction is a necessary step in song comparison using kALE: before comparing songs, all songs must be transposed into the same key. Key extraction was performed using trends present in western chord progressions, namely that when a song is in a fixed key, some chords are more popular than others. According to an analysis of 1300 popular songs over the last 50 years, chords I, V, vi, and IV are most popularly used, followed by ii and iii. Thus, in order to determine the key of a song, we look to the distribution of chords used in its chord progression representation. We trained a One vs. All classifier on known chord progression/key pairs to use as a classifier for unknown chord progressions. The input to the classifier is a chord distribution and its output is the predicted key of the song.

4.6 Our Implementation

We implemented and refined our method in Python2.x over the course of 6 months. Because all songs are analyzed independently, we parallelized the process of analyzing the songs. In addition, because we analyzed independent subsections of the waveform, we parallelized the extraction of a chord progression. Without parallelization, finding the BPM of a single song takes about 30 seconds and finding the chord progression, predicting the key, and transposing a single song takes about 4 minutes. With parallelization, Analyzing 100 songs takes about 1.5 hours, depending on the BPMs and lengths of the songs.

5. SONG COMPARISON

5.1 Distance between Chord Progressions

We define similarity measure of two songs as the mutual distance between their chord progressions. The motivating assumption for the validity of kALE is that if two chord progressions share similar sub-progressions, it is likely that the two songs will affect listeners in similar ways. Therefore, we find the mutual distance between two songs using k-gram analysis, comparing k-length sub-progressions from each song.

Using kALE, each chord in a chord progression plays some role in the distance to another song. Let us define two chord progressions for song X, and song Y, such that X consists of a sequence of chords $x_1, x_2, x_3, \cdots, x_n$ and Y consists of a sequence of chords $y_1, y_2, y_3, \cdots, y_m$. We compare all pairs of k-grams from X and Y by computing the edit distance between the k-grams. Thus, the minimum distance between two k-grams is 0 (in the case that the two k-grams represent the same sub-progression) and the maximum distance between two k-grams is k (in which case the i^{th} index of the k-gram from X never matches the i^{th} index from Y). The distance from song X to song Y is defined as the sum of the minimum edit distance found containing chord x_i.

$$d_{X \text{ to } Y} = \sum_{i=1}^{n} \text{minimum distance containing } x_i \tag{5.1}$$

Thus, if x_i is a part of multiple k-grams in X and/or x_i is part of a k-gram that is compared with multiple k-grams in Y, we greedily choose the lowest edit distance it is a part of. For example, if x_i is part of a sub-progression that gets compared with some sub-progression in Y, the edit distance of which is 0, we assert that because x_i exists in an exactly matching sub-progression in the other song, x_i should not add to the distance

of the two songs. The same process is performed in the reverse direction, obtaining the distance from Y to X. Let us call the distance from song X to song Y d_X and the distance from song Y to song X d_Y.We normalize the distances by scaling by the relative lengths of the songs and find the average:

$$d = \frac{\frac{d_X}{|X| \cdot \min(|X|, |Y|)} + \frac{d_Y}{|Y| \cdot \min(|X|, |Y|)}}{2} \tag{5.2}$$

With kALE, the minimum distance is 0 and the maximum distance is 1, such that the closer the distance is to 0, the more similar the songs.

One useful property of kALE is that even though as k grows, growing the distance between song X and song Y, the relative distances between sets of songs remains consistent. For example, with some k, if the distance between song X and song Y is d_{XY}, and the distance between song X and song Z is d_{XZ}, and $d_{XY} > d_{XZ}$, no matter which k we use to determine mutual distances, $d_{XY} > d_{XZ}$ will be maintained.

For our application, we required that each song be associated with a list of all other songs in the library, sorted by distance. Because k does not matter when generate relative song comparisons, we arbitrarily chose $k = 5$ in our implementation.

6. EXPERIMENTAL METHODS/PROCEDURES

To explore the use of kALE as a method by which to recommend music to listeners, we created an application that queues and plays MP3's based on user ratings of the previous song. The application has two modes:

1. Random selection of the next song from a library.

2. Selecting the next song based on the previous song's rating.

We define kALE's success in music recommendation as the increase in performance from random recommendations to our recommendation algorithm. We selected 20 Texas A&M students in various disciplines to use our application and provide recommendations for songs. 10 students were provided randomly selected songs, while the other 10 were provided recommendations based on the previous rating; we alternated mode of operation between experiments. In each case the experiment procedure is as follows:

1. The subject completes a semi-structured interview relating to their relationship, history, and experience with music. Questions include styles of music normally listened to and the settings in which music is normally listened to, musical talent and experience with music theory and analysis of chord structure, and listening experience (e.g. parts of the music the subject normally gravitates toward).

2. The subject is instructed that they will listen to 10 songs. When the subject decides a 1-10 rating for a given song, they may let the proctor know, who then stops the song play and asks further questions about the subject's opinion of the song.

3. The subject listens to and rates 10 songs as described in the previous item.

We wanted to make sure that subjects were basing their ratings on musical content alone and not past experience with the music. Therefore, the songs chosen for the study were written and performed by indie artists, ensuring that subjects would not have bias toward any particular song(s). We downloaded 100 random songs from NoiseTrade, a service that provides royalty free music from indie artists, and eliminated twenty-six songs that had long non-musical introductions and/or did not prominently feature chord progressions. We extracted the chord progressions from and pairwise compared the remaining 74 songs for use in the music playing application.

The recommendation algorithm uses the previous song's rating to generate the next song. In the application, each song is associated with a list of all other songs in our library ordered by distance to that song (the earlier a song occurs in a list, the closer the song). To determine which song to choose, we use a logarithmic scale skewed toward high ratings.

The formula for the curve shown in figure 6.1 is

$$number\_of\_songs \cdot \frac{exp(-rating/4) - exp(-10/4)}{exp(-1/4) - exp(-10/4)} \tag{6.1}$$

This formula takes the rating, derives a score for it, then scales the rating by the minimum rating, 1, and the maximum rating, 10. Then, the index in the ordered list is found by taking the result $(0 \leq result \leq 1)$, and multiplying by the number of songs, which in our libary is 74.

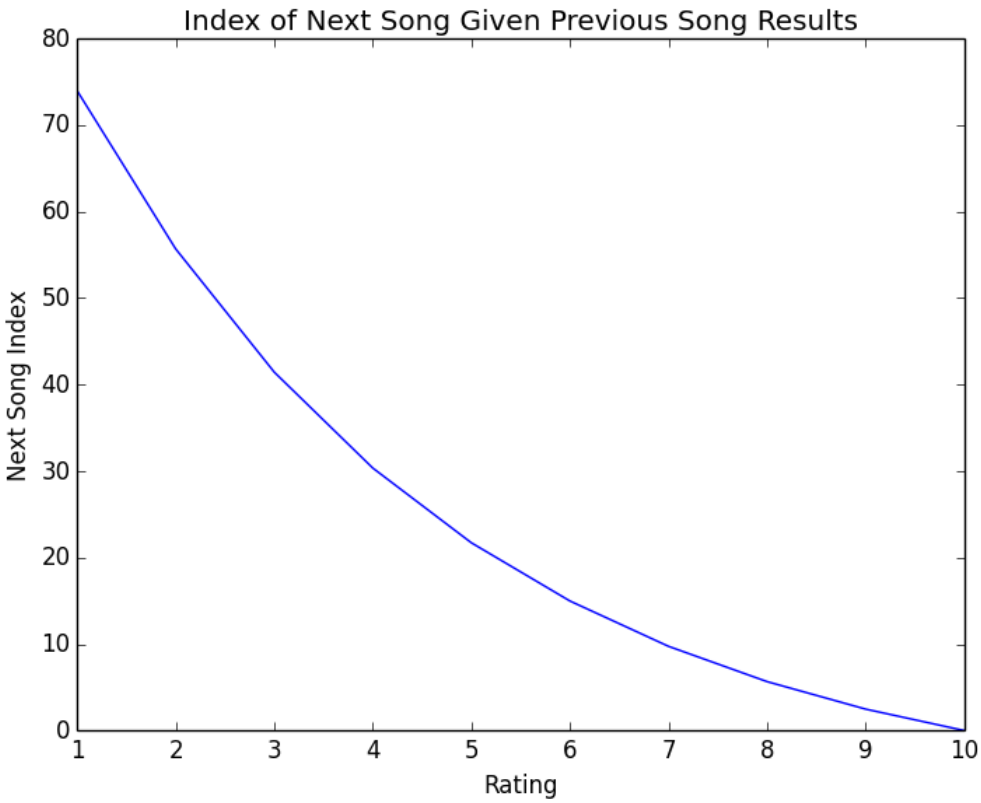

Figure 6.1: Next song index given the current song's rating.

7. RESULTS

The following graph (7.1) represents the frequency of songs played during the study.

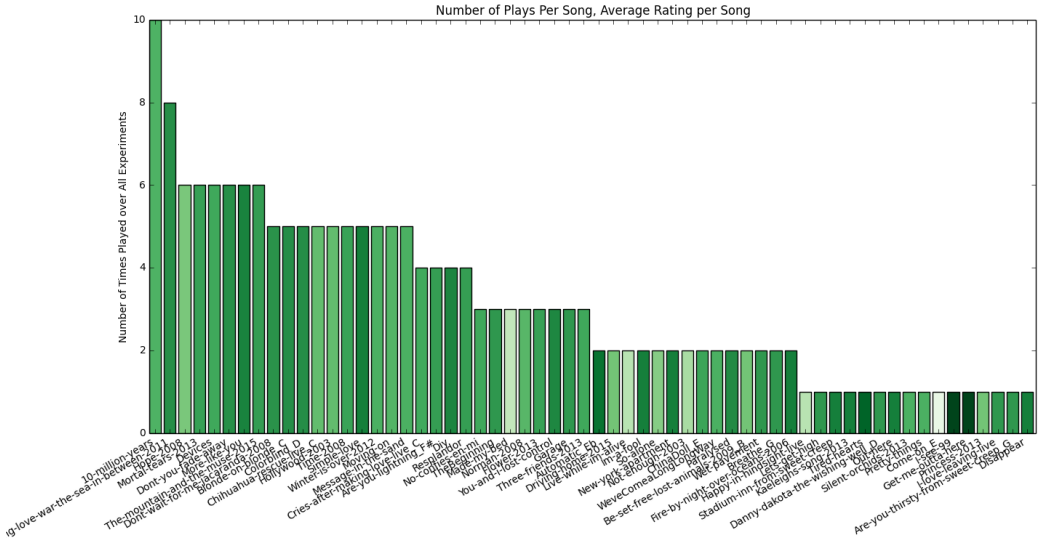

Figure 7.1: Frequencies of all of the songs played in at least one experiment.

The color on the bars represent average rating of the song over all experiments: the darker the green, the higher the rating. Though we attempted to give all songs equal probability of playing, some songs were selected more often than others, which we explore later.

After using our application with 20 subjects, 10 with randomly generated songs and 10 with the kALE recmmendation system, we compared rating transitions from song to song.

Plot 7.2 shows the four possible scenarios:

1. Low $\rightarrow$ Low: both ratings were low,

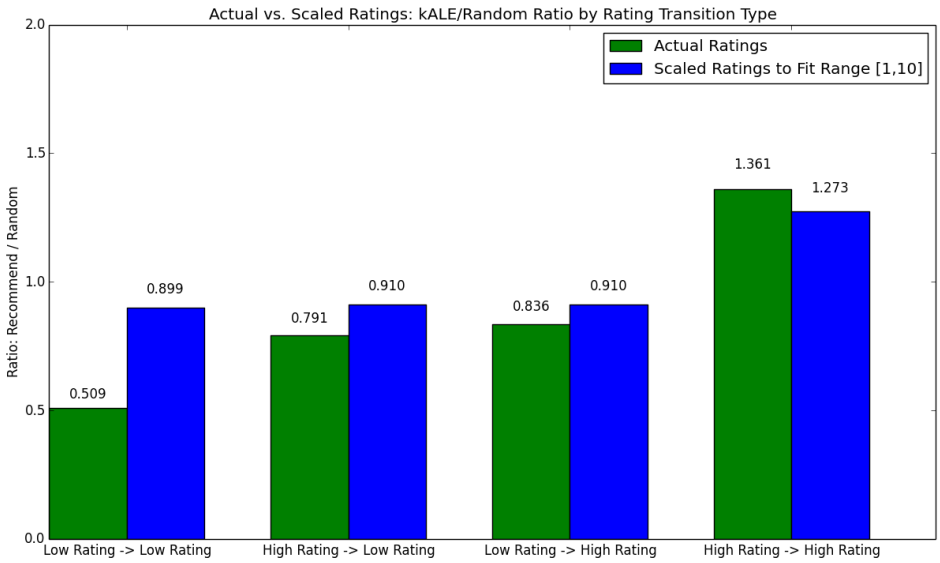

Figure 7.2: Successive rating trends from song to song, comparing randomly generated songs to recommendations using kALE.

2. High $\rightarrow$ Low: a rating is lower than the previous,

3. Low $\rightarrow$ High: a rating is higher than the previous

4. High $\rightarrow$ High: both ratings were high

Given that ratings were in the range [1,10], we classified anything above 5.5 as high and below 5.5 as low. We found the percentage of occurrences in each of the random case and kALE case, and divided the kALE percentage of each transition type by the corresponding type in the random case. Thus, a result of 1.0 shows no change in that category, a result more than 1.0 shows a rise in that category from random to recommendation, and a result below 1.0 shows that the random case had a higher rate of occurrence for that category than recommendation. In addition, we plotted two kinds of rating data: the green bars represent raw ratings provided by users and the blue bars represent scaled ratings per user

to fit a range [1,10], thus illustrating rises and falls in ratings within users.

We also recorded how long each user listened to each song to pair with the ratings provided. The following graph shows a trend we found significant:

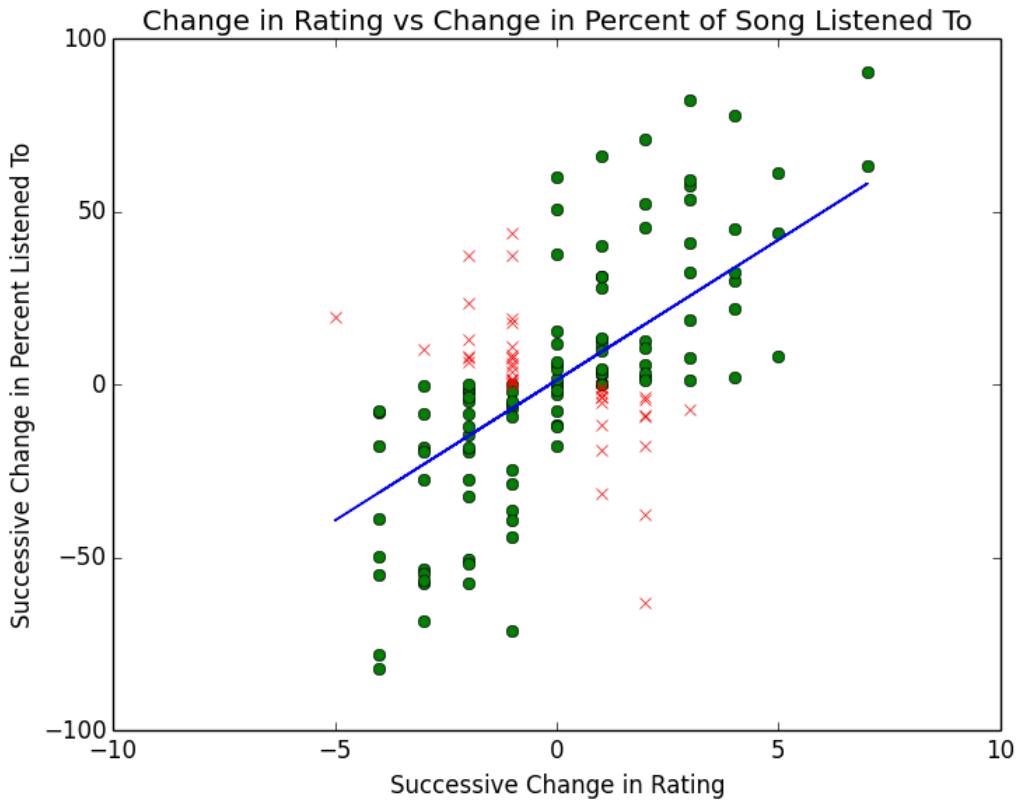

Figure 7.3: Successive rating trends compared with change in time listened from song to song.

In plot 7.3, two scenarios were found: the rating and time listened moved in the same direction (denoted by green circles), or the rating and time moved in opposite directions (denoted by red x's). The first case can be broken down into two categories: the rating and time listened both increase (quadrant II) or the rating and time listened both decrease

(quadrant III). The second scenario is illustrated by quadrants I and IV. The blue line shows the linear regression of these data.

8. DISCUSSION

8.1 kALE in Music Recommendation

We performed our music recommendation experiment with 20 users (10 with randomly generated songs, 10 with songs recommended using kALE). Because most ratings across the random and kALE cases were relatively high (the average is 6.52 and the median is 7), we removed bias from the songs themselves. Many subjects only rated songs in the range 5-9, so we scaled each subject's ratings to fit the range [1,10]. Scaled ratings show rating transition trends within a user, providing further insight to the behavior of the individual subjects.

We analyzed trends in the changes in successive ratings looking at four specific scenarios in successive ratings:

1. Low Rating $\rightarrow$ Low Rating
2. Low Rating $\rightarrow$ High Rating
3. High Rating $\rightarrow$ Low Rating
4. High Rating $\rightarrow$ High Rating

where Low is characterized by ratings below 5.5 and High is characterized by ratings above 5.5.

In our recommendation system, if a user gives a low rating, a very different is provided next. Because different does not necessarily mean better (i.e. there is no way of knowing if the subject will like the next song; it could be better or worse), scenarios 1 and 2 are less meaningful. When a subject gives a high rating, however, we provide a song with a similar chord progression with the hopes that the subject will like that song, as well.

Thus, a successful recommendation system will show an increase in High $\rightarrow$ High, and a decrease in High $\rightarrow$ Low when compared with the random case.

As is seen in figure 7.2, we see that the ratio of recommendation to random percentages of High $\rightarrow$ High is 1.361 in the raw ratings, and 1.273 in the scaled ratings. Because most raw ratings were above 5.5, there is a high likelihood of adjacent raw ratings are high; thus, the raw rating increase in this area is expected. The scaled rating increase in the category High $\rightarrow$ High shows that if a user liked a song, they would like the next song in terms relative to their own minimum and maximum ratings. This result, in particular, is an indication that using kALE to recommend songs yields an increase in user enjoyment from the random case.

When performing our experiments, we asked users why they did or did not like songs to gain insight into why some songs appeal to them more than others. We received a variety of responses: in addition to overall feel of a song (which is likely what our method targets), many subjects gave low ratings for voices they found "annoying," genres they did not like, the language a song is in, lyrical content, and/or other various reasons. In addition, some subjects liked songs for reasons other than the music itself, such as lyrical content or that a song reminded them of an artist or song they already like. Thus, the fact that our we only increased our High $\rightarrow$ High scenario from random to recommend by a factor of 1.361, and only decreased our High $\rightarrow$ Low scenario by a factor of 0.791, we can point to these other influencing factors as the reason. As our method competes with other factors important to our subjects, any increase at all is a success.

8.2 Our Music Recommendation Algorithm

In our analysis of songs used in the experiments, we found that some songs were played more often than others. After investigation, we found this was due to our recommendation algorithm preferring certain songs to others. The recommendation algorithm chooses the

next song to play for the subject by selecting the index in a sorted list of songs, ordered by increasing distance with the last song played (determined by kALE). To prevent repeating songs within an experiment, if the selected song has already played, the surrounding songs in the list are explored until a new song has been found. Using the formula presented in equation 6.1, the following table describes which index is initially chosen for each rating:

Rating	Index of Next Song
1	74
2	55
3	41
4	30
5	21
6	14
7	9
8	5
9	2
10	0

Table 8.1: Index of the next song to be played when the corresponding rating is provided by a subject.

There 74 songs in our library and each song is associated with an ordered list. We looked at the most popular songs played and found that they exist in many lists at or around the indices chosen by the algorithm. For example, the most popular song played, "10 Million Years" by Travis Roman, occurs at index 9 in 7 lists and at index 5 in 5 lists. Because 7 and 8 were the most common ratings provided, and because ratings 7 and 8 yield the indices 9 and 5 respectively, "10 Million Years" had a high probability of being selected when compared with other songs. Similarly, the second most popular song used by our algorithm, "Farther Along Love War the Sea in Between" by Josh Garrels, occured at index 1 in 6 lists and at index 4 in 5 lists. As described in table 8.1, if a rating of 8

were provided and the song at index 5 were already played during that experiment, the algorithm would choose index 4. Thus, "Farther Along Love War the Sea in Between" also has a high probability of being played. This pattern illustrates why some songs were played significantly more often than others.

8.3 Rating vs. Time Listened

After noticing some patterns within users, we explored the relationship between the rating a user provided and the amount of time spent listening to that song. Though we did not find a direct correlation between the rating a user provided and the time listened (specific listening behavior between users varied greatly), we did find evidence for the general case: users tended to listen to a larger parentage of songs they liked than songs they did not. Figure 7.2 shows this correlation: the rise and fall of rating from song to song and time listened from song to song tended to move in the same direction.

9. CONCLUSION

Because moving from randomly generated songs to songs generated using kALE increased subjects' enjoyment of the songs they were provided, we feel that kALE song comparisons is a valid metric to utilize in content-based music recommendation. Because of factors other than the feel of the music itself (which could point to chord progression) found to influence subjects' enjoyment of songs, kALE can have its best impact in recommendation systems when used in conjunction with additional content-based metrics.

REFERENCES

[1] Y. Song, S. Dixon, and M. Pearce, "A survey of music recommendation systems and future perspectives," in *9th International Symposium on Computer Music Modeling and Retrieval*, 2012.

[2] J. A. Burgoyne, J. Wild, and I. Fujinaga, "An expert ground truth set for audio chord recognition and music analysis.," in *ISMIR*, vol. 11, pp. 633–638, 2011.

[3] G. Tzanetakis, G. Essl, and P. Cook, "Audio analysis using the discrete wavelet transform," in *Proc. Conf. in Acoustics and Music Theory Applications*, 2001.