Music Classification - Pick a beteer title later

Christian Johnson
Electrical Engineering Department
United States Coast Guard Academy
New London, Connecticut 06320
Email: Christian.S.Johnson@uscga.edu

Daniel Nusraty
Electrical Engineering Department
United States Coast Guard Academy
New London, Connecticut 06320
Email: Daniel.Y.Nusraty@uscga.edu

Joshua West
Electrical Engineering Department
United States Coast Guard Academy
New London, Connecticut 06320
Email: Joshua.C.West@uscga.edu

***Abstract*—This is an abstract**

I. Introduction

With the advent of music subscription services, many people have become accustomed to automatically generated playlists, tailor-made to their own personal taste. These services provide such playlists at the push of a button, analyzing the user's listening history to continuously recommend similar songs. Providers such as Pandora, Spotify, and LastFM maintain massive databases containing context information on millions of songs in order to present the best recommendations. For those who prefer to maintain their own local music library however, the options for tailored music recommendations are dramatically reduced. This paper seeks to explore the application of digital signal processing techniques in order to implement similar music recommendation functionality that will work with local music files.

II. Theory

A. Background

The majority of music analysis is based on a family of functions known as Fourier Transforms. A Fourier Transform is used to transform time-domain audio information into a frequency-domain representation. In the time domain, sound is represented as amplitude in terms of time. In the frequency domain, sound is instead represented as a function of magnitude based on frequency. What is important to recognize, is that in both cases the signal is still continuous, meaning unbroken, with theoretically infinite data between two points. A continuous signal contains a great deal of information that makes it difficult to perform operations on. In order for a computer to be able to process such a signal, that amount of information must be reduced through an operation known as *sampling*. Sampling replaces the continuous signal with a discrete representation; an array of values at (typically) evenly spaced indeces of time. The normal Fourier Transform cannot operate on discrete signals however, which is where a specific type of Fourier Transform known as the Discrete Fourier Transform (DFT) comes into play. The DFT is comprised of a single summation operation, which will produce an array of complex number representations.

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn} \tag{1}$$

Each value in the DFT output $S[k]$ is a complex number that represents a point in an N dimensional vector space. The magnitude and phase of that point represent the content of the time-domain signal at frequency $\frac{2k\pi}{N}$, where N is the length of $S[k]$. These values depend a great deal on a concept known as *sampling frequency*, which refers to the time taken between each sample of the original time-domain signal. Each 'bin' of $S[k]$ represents a collection of frequencies, $\frac{-F_s}{2N} + \frac{k*F_s}{N} < f < \frac{F_s}{2N} + \frac{k*F_s}{N}$. As such, it is important to recognize that $S[1]$ does not directly correspond to a specific frequency value; in order to find the DFT output for, e.g. 20Hz, one must determine which bin this frequency will fall into.

1) The FFT: In its original form, the DFT is quite computationally expensive and complex, comprised of $\mathcal{O}(n^2)$ operations. An algorithm known as the Fast Fourier Transform (FFT) dramatically simplifies this complexity, reducing the expense to $\mathcal{O}(n*log(n))$ operations. It does this by exploiting symmetry within the DFT. Equation (1), the basic representation for the DFT, is periodic about N, i.e. $X_{k+l*N} = X_k$ for any integer l. The FFT uses this relationship to break the DFT into even and odd components.

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \\ &= \sum_{m=0}^{N/2-1} x(2m) * e^{-j2\pi km/(N/2)} \\ &+ \sum_{m=0}^{N/2-1} x(2m+1) * e^{-j2\pi km/(N/2)} \end{aligned} \tag{2}$$

This particular implementation is known as the Cooley-Tukey FFT, and is one of the most widely used algorithms in the world.

B. Analyzing Music

In order to analyze and classify music, it is important to understand how it is structured. Fundamentally, music is simply composed of air molecules vibrating at different frequencies and amplitudes. This means that there are a theoretically infinite number of ways to structure and divide these different frequencies. In western music, the most common method uses what are known as major scales. The major scales are composed of 7 unique notes, A through G ordered based on their pitch interval. While advanced music theory is outside the scope for this paper, it is important to recognize that a single major scale is composed of 7 notes, with a predetermined order of pitch difference. These pitch differences are somewhat unintuitive however, since music is arranged on a logarithmic scale This means that the note typically known as A0 (The note A in the first recognized octave) is defined as 27.5 Hz, A1 is 55 Hz, A2 is 110 Hz, and so on. This is significant, because it means that the most commonly used frequencies in music will typically be grouped towards the lower half of the established musical range. In general, musical notes are defined from C0 at 16.35 Hz all the way to B8 at 7902.13 Hz. Most types of music will use notes toward the center of this range, as the extremes are considered somewhat unpleasant; however, different genres of music may use varying distributions of these frequencies. For instance, an EDM, or electronic, artist may choose to use frequencies in the 0 or 1 octave in order to produce a more physical feeling. These frequencies are considered "super-low", since they are so low that, when combined with other tones, they are more often *felt* than heard.

III. Experimental Procedure

Using this information, it becomes relatively simple to start processing audio files and comparing them to each other. There are several key steps to this process; audio information should be read from a file, then that audio information should be compressed and generalized in order to compare two songs. This generalized audio information should be collated into some structure that will allow simple access later. The simplest application for this concept is to compare a collection of songs to a single different audio file and return a list of songs from the collection that are most similar to the original audio file.

A. Processing a Single Audio File

Digital audio files are simply a collection of amplitude information arranged based on time. This information is sampled from the original continuous audio source, typically at a rate of 44,100 Hz (Meaning that there should be 44,100 samples making up a single seconds worth of audio information). This format is incredibly useful for a computer which must play recorded audio, since the data will simply tell a computer the *amplitude* at which to vibrate a speaker diaphragm, and the time at which to do so. It is less applicable to examining and classifying the underlying composition of the audio however, and must be tranformed using the FFT to a more applicable format. Taking the FFT shifts the sampled data from amplitude in terms of time to amplitude in terms of frequency, as discussed earlier.

B. Interpreting Data

The FFT output is formatted in a much more accessible fashion than the time domain data, providing an array of complex bin values that represent a range of frequencies as discussed earlier. In order to use this information to compare two song files, it is important to generalize the information somewhat. Comparing the raw FFT of two files can hold some value, but they contain such a high volume of information that such a comparison could result in too high a degree of specificity. One method for generalizing this data is to separate the frequency spectrum into so-called 'bands'. These bands can be chosen based on a variety of different factors, but it seems most logical to divide them based on musical structure. Referencing the scale structure discussed earlier in this paper, a reasonable set of frequency breakpoints could be $0 - 60$ Hz to represent 'superlow' frequencies, $61 - 240$ Hz for lows, $241 - 500$ Hz for mids, $501 - 2000$ Hz for highs, and $2001 - 8000$ Hz to represent 'superhighs'. Since the FFT bin indeces do not correspond to actual frequencies, it is necessary to convert these frequencies in order to determine the closest bins that they correspond to. Calculating the bin into which a specific frequency will fall relies upon the relationship between sampling frequency and sequence length. If N represents the number of samples in an FFT signal, f represents the frequency value, and r represents sampling rate then bin number can be found from the following equation:

$$\text{Bin Index} = \frac{f * N}{r} \tag{3}$$

Using this equation, it becomes a trivial operation to determine the necessary breakpoint indeces, then to segment the FFT data into a set of bands. This segmented FFT groups together several series of frequencies which make up similar ranges. Finding the bin with the largest magnitude within each segment presents a way to identify which frequencies are most prominent within a given audio file. Given the segments defined earlier, this compresses 44,100 samples per second of frequency information to an array of 5 complex numbers of the form $a + bj$, which can each be rewritted as polar coordinates. Polar coordinates, in general, represent a point or vector on a two-dimensional plane. The distance between the endpoints of two vectors can be computed using the Euclidean Distance Algorithm, wherein each polar coordinate is converted to cartesian coordinates of the form (x, y). The Euclidean Distance is simply the length of the line segment connecting the two coordinates, using the following relationship:

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4}$$

Thus, the similarity between two songs can be approximated from each song's array of 5 polar coordinates. The distance between each element is found and concatenated into an array of 5 distance values. The average of these values can be approximated as a single similarity metric to compare two songs.

C. Storing Comparison Data

Above, it is shown how a song can be characterized through its frequency data and compared to a single other song. This comparison takes the form of a distance metric, inversely representing the proximity of the two songs' primary component frequencies. The larger this distance metric, the less similar the two songs are. In order to construct a system which can handle more than two songs, it is necessary to construct a method for storing and accessing data. The storage and access functions, while related, can be constructed seperately. The system can 'take in' new audio files - performing the FFT, segmenting, and gathering an array of polar coordinates for each file. These polar arrays are stored as values in a dictionary structure, with the filename as the key. This dictionary structure allows for an extremely fast lookup time when accessing values by key, and serves as persistant storage for frequency data. Upon an entry into the dictionary, the new entry is compared to every other entry in the dictionary, producing a three element array with the first two elements being the names of the two songs being compared, and the third value being the comparison metric; the distance between the two songs. This 3 element array is concatenated into a second datastructure - a list comprised of three element arrays. This list serves as storage for comparison data, comprised of every possible pair of songs that the system has recorded. In order to produce a list of 'similar' songs, the system can take a file as input, checking whether that file is already present in the dictionary of frequency data. If the file is not present, the FFT is computed, the information is added to the dictionary, and comparison data is added to the list. If the data is already present, the FFT is skipped. The system searches for all elements of the list that contain the name of the song being searched for and removes that song to produce a list of two element arrays. These arrays are sorted by distance metric to produce an ordered list of songs. The songs at the top of the list have the smallest distance, and therefore are most similar to the baseline song.

IV. Results

Because of the qualitative nature of music, it is somewhat challenging to evaluate the system's actual performance. In order to actually examine how well the system performs, it is necessary to track the results produced using a relatively diverse collection of music files. Using a dataset of around 21 songs, comprised of several songs each from about 7 artists, the system was consistently able to organize the songs into seemingly logical orders based on similarity. One example output for this system is shown in Table I, producing songs that are similar to 'Bridge Over Troubled Water', by Simon and Garfunkel. Songs written by Simon and Garfunkel appear towards the top of this list; 'Mrs. Robinson', 'El Condor Pasa', 'A Hazy Shade of Winter', 'Homeward Bound', and several others. These are all composed by the same author that wrote the original song, 'Bridge Over Troubled Water', and share simular instrumentations, melodic structure, and tonal quality. Songs that are ranked similar but not written by Simon and Garfunkel include 'The View Between Villages' and 'Stick Season' (both written by Noah Kahan), 'Heading South' (Zach Bryan), and 'Mr. Forgettable' and 'Miserable Man' (David Kushner). These are all of a similar genre and tonality to Simon and Garfunkels' music, sharing similar instrumentation to 'Bridge Over Troubled Water'. Songs written by artists such as Breaking Benjamin and Nickelback (both rock artists) appear towards the end of the table, with significantly larger distance values. Performing the same operation on 'Numb' by popular rock artist Linkin Park produces the output shown in Table II. Songs by other rock artists appear prevalent at the top of this table, featuring heavy use of electric guitar and melodic structure. Beyond this surface level analysis, there is not much to quantify exactly how effective this method is at comparing songs. Musical analysis like this is inehrently qualitative, and the best method for determining the efficacy of this comparison method will simply be to apply it to as wide a variety of music as possible, and note any discerpancies that might appear in the tables that it generates.

TABLE I
Comparison Output - A Bridge Over Troubled Water

Song Name	Distance
MrsRobinson.mp3	2991.601247
TheViewBetweenVillages.mp3	3056.214826
StickSeason.mp3	3556.499786
ElCondorPasa.mp3	3823.472017
AHazyShadeOfWinter.mp3	4217.250688
HomewardBound.mp3	4245.070162
HeadingSouth.mp3	4341.300415
ScarboroughFair.mp3	4867.695298
TheBoxer.mp3	5639.500292
TheDanglingConversation.mp3	6817.155551
MrForgettable.mp3	8602.562881
Someday.mp3	9474.584567
MiserableMan.mp3	9654.682392
It'sNotMyTime.mp3	10554.550396
Burn.mp3	10640.759481
SomethingInTheOrange.mp3	11096.156537
DuHast.mp3	11332.417538
LastOfTheAmericanGirls.mp3	12020.358250
FromAustin.mp3	12213.250311
VivaLaVida.mp3	12465.632844
NorthernAttitude.mp3	12583.604234
IWillWait.mp3	13405.201120
OrangeJuice.mp3	13510.787237
BurnBurnBurn.mp3	14431.161942
IntoTheNothing.mp3	15818.285350
21Guns.mp3	16665.654127
Kryptonite.mp3	19614.481553
21stCenturyBreakdown.mp3	20456.969334
LetMeBeMyself.mp3	20523.500544
AwakeMySoul.mp3	20909.033851
HolidayBoulevardOfBrokenDreams.mp3	22022.934161
AnthemOfTheAngels.mp3	24924.491533
WithoutYou.mp3	26450.681219
JesusOfSuburbia.mp3	27148.855861
WakeMeUpWhenSeptemberEnds.mp3	28272.608190
DearAgony.mp3	32904.681492
Numb.mp3	33977.307564
SomewhereIBelong.mp3	34766.015645
IWillNotBow.mp3	38595.117180
AmericanIdiot.mp3	40137.556831
MeinHerzBrennt.mp3	43060.259956
NeverGonnaBeAlone.mp3	45824.397707
IfTodayWasYourLastDay.mp3	47578.342873
BreakingTheHabit.mp3	76704.242089

V. Conclusion

This paper has explored the application of digital signal processing techniques to the classification and comparison of audio files, and the efficacy of these techniques for generating song recommendations. Analyzing the frequency content of audio signals and comparing them based on distance metrics proved to be an effective method for organizing songs, and

TABLE II
COMPARISON OUTPUT - NUMB

Song Name	Distance
DearAgony.mp3	2970.683300
SomewhereIBelong.mp3	4686.380957
WakeMeUpWhenSeptemberEnds.mp3	7347.389843
JesusOfSuburbia.mp3	8834.466303
AnthemOfTheAngels.mp3	9464.236452
WithoutYou.mp3	9656.667749
IWillNotBow.mp3	11976.448806
NeverGonnaBeAlone.mp3	12275.252101
HolidayBoulevardOfBrokenDreams.mp3	12408.432227
AwakeMySoul.mp3	13651.289895
Kryptonite.mp3	15442.333784
21stCenturyBreakdown.mp3	16368.045047
IntoTheNothing.mp3	18786.807675
21Guns.mp3	19094.048465
OrangeJuice.mp3	20480.090985
BurnBurnBurn.mp3	21385.618375
NorthernAttitude.mp3	21416.057050
FromAustin.mp3	21786.282400
IWillWait.mp3	21997.100687
LastOfTheAmericanGirls.mp3	22587.683924
SomethingInTheOrange.mp3	23149.128868
AmericanIdiot.mp3	23491.271419
Burn.mp3	23646.102266
VivaLaVida.mp3	24377.637756
MiserableMan.mp3	24498.824598
It'sNotMyTime.mp3	24629.107152
DuHast.mp3	25359.161402
Someday.mp3	25848.268380
MrForgettable.mp3	25910.026811
LetMeBeMyself.mp3	27539.609328
StickSeason.mp3	30621.406412
TheDanglingConversation.mp3	30911.407496
IfTodayWasYourLastDay.mp3	31312.307911
MeinHerzBrennt.mp3	31700.800905
HomewardBound.mp3	32754.494395
ScarboroughFair.mp3	33704.533482
BridgeOverTroubledWater.mp3	33977.307564
TheBoxer.mp3	34604.245269
MrsRobinson.mp3	34977.982150
TheViewBetweenVillages.mp3	35506.735113
HeadingSouth.mp3	36104.552918
AHazyShadeOfWinter.mp3	37514.935606
ElCondorPasa.mp3	37784.515983
BreakingTheHabit.mp3	43554.306409

demonstrates the application of frequency analysis for song recommendation. Experimental results show promising outcomes, with songs sorted in a seemingly logical order based on musical attributes, genre, and general composition. While the qualitative nature of music analysis presents challenges in evaluating system performance, this approach offers a promising foundation for further research and development in personalized music recommendation systems. Future work could involve refining the algorithm to provide results that better differentiate between extremely similar songs, improving the speed of the system, and exploring additional features in order to enhance the accuracy and relevance of music recommendations.

VI. Bibliography

References

[1] Dmitry Pastukhov, "How the Spotify Recommendation System Works: A Complete Guide," Available online: https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022.

[2] Jovan Jovanovic, "Shazam It!: Music Processing, Fingerprinting, and Recognition," Available online: https://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition.

[3] "What Is the Meaning of the DFT?," Available online: https://dsp.stackexchange.com/questions/5915/what-is-the-meaning-of-the-dft.

[4] "What Is a Frequency Bin?," Available online: https://dsp.stackexchange.com/questions/26927/what-is-a-frequency-bin.

[5] "FFT Bin Width Clarification," Available online: https://stackoverflow.com/questions/10754549/fft-bin-width-clarification.

[6] "Fast Fourier Transform in Python," Available online: https://pythonnumericalmethods.studentorg.berkeley.edu/notebooks/chapter24.03-Fast-Fourier-Transform.html.

[7] "Note Frequencies," Available online: https://muted.io/note-frequencies/.

[8] "Major Scales," Available online: https://rwu.pressbooks.pub/musictheory/chapter/major-scales/.

[9] "Common Music Scales," Available online: https://www.onlinepianocoach.com/common-music-scales.html.

[10] "Frequency Spectrum of Common Musical Instruments," Available online: https://www.petervis.com/hi-fi-info/jvc-sea-graphic-equalizers/frequency-spectrum-of-common-musical-instruments.html.