RASM : A Balanced Dataset For Artist Similarity Measure

Submitted in partial fulfillment of the requirements for

the degree of

Master of Science

in

School of Music

Anirudh Mani

Master of Science in Music and Technology

Carnegie Mellon University
Pittsburgh, PA

May 2018

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor Professor Roger Dannenberg. One of the major reasons why I was able to undertake a project of this scope was because of his never ending support, wealth of knowledge and wisdom. There are lessons I have learned from him which go beyond this thesis project, and have shaped my professional path and will continue to do so. Thank you Professor for your patience with me, your guidance and direction when I needed it the most and didn't even know it. I will forever be grateful for this opportunity to work with you!

Thank you Riccardo, Professor Richard Stern, Professor Tom Sullivan and rest of the Music and Technology board for mentoring us, showing us the right direction and making us learn outside the classroom as well.

My family has always been the bedrock of every endeavor I've undertaken in my life. I'd like to acknowledge my grandparents, parents and my sister Aditi, who made sure that I never felt too far away, and were always there for support from the other side of the world.

I am indebted to my friends who helped me immensely by hearing out my ideas related to this thesis, apart from giving me feedback on all my mock presentations. It is hard for me to imagine this thesis project without their moral support. A big thank you goes out to Swetha and Ajaya for enduring me during the more challenging times and pushing me to improve everyday, Mayam for the never ending encouragement, Akhil for making sure I always got that extra dose of motivation when I needed it, and Dareen for being one of my pillars during this time of professional and personal growth.

A special mention goes out to my friends who were also my batch mates - Nicholas Pourazima, Michael Bridges, Craig Hesling and Garrett Osborn for being my sounding board for ideas related to this thesis project. These two years at Carnegie Mellon would not have been the same without your company and I hope we never stop having conversations about music technology, life and more.

Finally, my heartfelt gratitude goes out to Carnegie Mellon University for this enriching experience. I knew I was making the right decision two years back to move countries and come here, and I could not have learned more from being in the company of so many amazingly talented people.

Abstract

Estimating artist similarity from audio content is an interesting problem for music information retrieval. Artist similarity judgments depend upon individual preferences, so meaningful similarity ratings must be based on general trends observed in a large population of listeners. In this thesis, we explore the possibility of using Spotify's related artists feature to establish a "ground truth" for similarity between artists in the free music archive (FMA). We believe Spotify's large user data is useful for the artist similarity problem, and FMA provides full audio content for content-based artist similarity estimation research. Due to the non-uniformity of the data (some artists in FMA are unknown to Spotify, some similar artists are absent in FMA, and so on), we carefully subset FMA for our related artists similarity measure (RASM) dataset such that: (1) every artist in the dataset should have similar artists in the dataset, (2) the dataset has a number of songs by each artist, and (3) genre in the dataset are well represented by artists in the dataset. Moreover, these numbers should be balanced as much as possible. We describe the creation of a dataset that meets these criteria and offer the dataset as an open resource for MIR research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Humans love to make music recommendations based on their own tastes. We all have favorite compositions and performances, but we also have favorite artists. Just as we might think “if you like this song, you will like this one too,” it is common to think “if you like this artist, you will like this one too.” This is the basis and motivation for artist similarity research. When we hear new artists, we can associate them with artists we are familiar with, at least in many cases. These associations, which can be thought of in terms of a distance between different artists, are certainly very subjective. The concept of universal artist similarity raises epistemological issues, but at the same time, the concept is quite desirable. Artist similarity is directly applicable to music inventory organization and music recommendation.

As with many big data problems, including those of music information retrieval, research studies rely on some definition of "ground truth" serving as training data and enabling evaluation and comparison of automated systems. The artist similarity problem is no different; we need some notion of ground truth here as well. Our ground truth should be a function of music listening and recommendation patterns. Listeners’ preferences can be compared to those of other listeners to construct a "ground truth" (we intentionally use quotation marks here because it should be apparent that there is nothing absolutely true about our "ground truth," but this "ground truth" is based on actual listening habits, so we can at least argue that it has practical application). While every listener is unique, we assume that there are trends and commonalities in music perception and preference. By looking for trends, we attain a global perspective [2]. This is achieved by the Spotify's related artists feature set which can be accessed from their Web API. As part of this project, artists from the FMA dataset are used to query Spotify API in order to create RASM, which stands for Related Artist Similarity Measure. *Our goal is to achieve a balanced and labeled microcosm where artist similarity is based on many listeners’ music consumption patterns.*

1.1 Background

Before listing out the details of the dataset, it is important to establish the definitions of a few concepts used in this thesis project.

1.1.1 Estimation of Artist Similarity Based on Content

The main motivation for creating our dataset is to enable research on artist similarity. In particular, the full audio content available in the FMA data combined with 'ground truth' artist similarity should facilitate studies on content-driven artist similarity estimation. Of course, similarity can be estimated from listening habits, but when new works are created, there are no records of listening available, and even many of the works and artists in FMA cannot be located on Spotify. If we could learn to judge artist similarity using numerical methods that depend only on audio content, then we would have a new and interesting tool for large-scale music inventory management and organization.

1.1.2 Song Similarity versus Artist Similarity

One might question the importance of artist similarity given all the work on Song Similarity. The latter refers to a measure of how subjectively close songs are. Studies of Song Similarity often use acoustic and text features to estimate Song Similarity. Artist Similarity on the other hand is an overall measure of closeness between two musical artists based on a bigger picture of their musical work [3]. This measure, also highly subjective, can differ from Song Similarity by considering more factors such as the cultural sound and context of the artists, overall themes, and the nature of musical instrumentation and orchestration. In other words, overall artistic sensibility can come into play as well as sonic details of particular songs. Perhaps future research will tell us more about the relative importance of different factors involved in the judgment of Artistic Similarity and Song Similarity. For now, we only claim that these are *potentially* different concepts and therefore Artistic Similarity is a concept worth pursuing.

1.2 Prior Work

In the past, people have studied musical similarity from the score level, the audio level, cultural level [1], and at the level of listening patterns. We have even seen text-based approaches [4]. However, the models we have are difficult to compare to each other for various reasons. To begin with, it is difficult to obtain a ground truth.

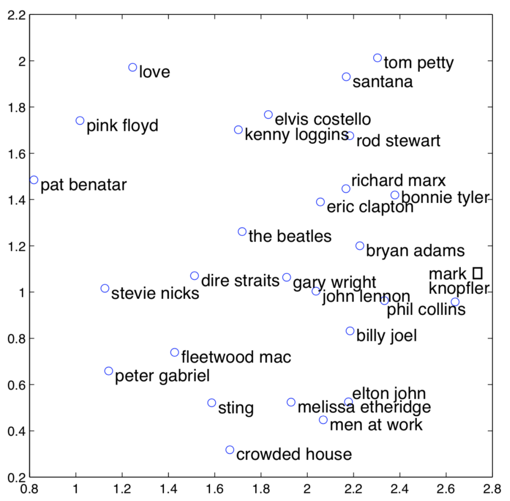


Figure 1.1: Subset of Artists embedded in 2-D space using Erdos Measure (from the original paper [1]).

Each of these previous models has a different approach to the calculation of a similarity measure. In score-based methods, the main idea is to extract performance characteristic like key and repetition of progressions from the musical piece and use them to compare with other pieces. In audio-based methods, spectral information and various spectral features are used extract style, genre, mood, and other such labels which define how similar sounding two artists are. Cultural similarity as described in [1] focuses on more innate features, which requires the availability of community meta data, reflecting the perception of artists in public.

In [3], a comparison of various subjective assessments of artist similarities is performed and the results are discussed. One of the distance metrics, Erdos measure, as shown in Figure 1.1, seems to perform the best when compared against a survey of listeners by the authors of the papers. Another measure that is investigated in this paper is the collected user data from OpenNap [5], a popular music sharing service at the time of this publication. This measure brings in an aspect of cultural similarity in the ground truth; however, due to the nature of this data, it cannot be compared well to the above-mentioned listener survey data. An apparent observation at this point is that not only are the measurements of artist similarity quite subjective, but so is creating an absolute ground truth for it. Therefore, we should instead move toward finding the most agreeable ground truth, and that is one of the core ideas behind this thesis because we try to change a subjective problem to an objective one.

Semantic-based approaches for artist similarity calculation, as described in [4], make use of any free text information available, such as artist biographies. The authors do not directly compare their text-based approach to any acoustic or collaborative filtering-based approaches. However, the results show that such a method can certainly be used as another signal into a music recommendation system.

Spotify has about 140 million active users (as reported last on June 15, 2017). It performs the calculation and keeps a record of artists that are similar to one another based on its users' listening patterns. Historically, Spotify has relied predominantly on collaborative filtering-based approaches to power their recommendations. The idea of collaborative filtering is to determine the users' preferences from their usage data. For example, if two users listen to largely the same set of songs, their tastes are probably similar. Conversely, if two songs are listened to by the same group of users, they probably sound similar. This information can be exploited to make recommendations.

1.2.1 Problems with Estimating Artist Similarity

As discussed in [1], some of the problems with calculating a single global artist similarity measure are:

Individual Variation

That people have individual tastes and preferences is central to the very idea of music and humanity. By the same token, subjective judgments of the similarity between specific pairs of artists are not consistent between listeners and may vary with an individual's mood or evolve over time. However, while it may not hold true for all users, it is possible to recommend a group of artists who form a set and sound similar to the given artists in many ways, based on the listening patterns of millions of music listeners.

Multiple Dimensions

The question of the similarity between two artists can be answered from multiple perspectives: Music may be similar or distinct in terms of genre, geographical origin, instrumentation, lyric content, historical time frame, and so on. While these dimensions are not independent, it is clear that different emphases will result in obtaining different artists. However, a collaborative filtering-based approach for ground truth estimation is useful here as well. If a *Listener A* likes *Paul Anka* and *Drake* because he or she is Canadian, then these recommendations are suggested to *Listener B* only if *Listener A* and *Listener B* have significant overlap in their prior music consumptions patterns.

Variability and Span

Few artists are truly a single 'point' in any imaginable stylistic space, but undergo changes through their careers, and may consciously span multiple styles within a single album. Trying to define a single distance between any artist and widely-ranging long- lived musicians such as David Bowie or Prince seems unlikely to yield satisfactory results. However, when a *Listener A* follows this journey of sound evolution of artists and finds two of them similar, the recommendation in the ground truth is made to *Listener B* if *Listener A* and *Listener B* have significant overlap in their prior music consumptions patterns.

1.2.2 On Machine Learning with Audio

Machine Learning has been used in the domain of music in various capacities, from analysis and music information retrieval, to algorithmic composition and music performance. Automatic classification of music in its various forms has been a popular task in the research community for quite some time now. Supervised classification problems employ models ranging from Support Vector Machines (SVM) to Deep Neural Networks and solve various problems in the space of music audio, sound event detection, and many others. Unsupervised Learning methods, such as Gaussian Mixture Models, have also found much use in speech related applications.

A more traditional pipeline of a machine learning tasks in the domain of audio involves deriving handcrafted features from audio segments and then passing these features through a classifier like an SVM. These handcrafted features need to be chosen carefully depending on the nature of the task at hand. However, many features have been known to work better for certain kinds of problems. For example, mel-frequency cepstral coefficients (MFCC) have been known to work well in speech audio tasks such as automatic speech recognition (ASR) and for music genre detection. In more recent years, with the advent of deep learning methods, this traditional pipeline has been receiving much competition from something known as end-to-end learning, where the deep neural network not only learns to approximate the classification function but learns its own features that are used for classification as well. As one would guess, the input to these networks is raw audio. Various examples include SampleRNN [6], WaveNet [7], WaveGAN [8] used for audio generation. With advancements in machine learning and deep learning specifically, an increasing amount of data is required for these complicated models to learn the underlying relationship effectively. The FMA is a large dataset with full-length audio files, and when combined with the related artists' information, it serves as a valuable large dataset for research in the area of artist similarity using such end-to-end deep learning methods as well.

1.3 Related Work

1.3.1 Free Music Archive Dataset

The FMA is an open and publicly accessible dataset suitable for evaluating several tasks in MIR, a field concerned with browsing, searching, and organizing large music collections. It was made with the community's growing interest in feature and end-to-end learning in mind, with the goal of making large audio datasets available for MIR research. The FMA provides *917 GiB* and 343 days of Creative Commons-licensed audio from *106,574 tracks* from *16,341 artists* and *14,854 albums*, arranged in a hierarchical taxonomy of *161 genres*. It provides full-length and high-quality audio, precomputed features, together with track- and user-level meta data, tags, and free-form text such as biographies [9].

1.3.2 Spotify Web API

Spotify has about 140 million active users (as reported last on Jun 15, 2017). It performs the calculation and keeps a record of artists that are similar to one another based on users' listening patterns. Historically, Spotify has relied predominantly on collaborative filtering-based approaches [10] to power their recommendations. The idea of collaborative filtering is to determine the users' preferences from historical usage data. For example, if two users listen to largely the same set of songs, their tastes are probably similar. Conversely, if two songs are listened to by the same group of users, they probably sound similar. This information can be exploited to make recommendations.

The Spotify Web API lets user applications fetch data from the Spotify music catalog and manage users' playlists and saved music. Based on REST principles, the Web API endpoints return meta data in JSON format about artists, albums, and tracks directly from the Spotify catalog.

Chapter 2

Methodology

We want to combine data on artist similarity with a large dataset of music audio. The data from Spotify's Web API is based on the listening patterns of millions of users. The FMA is a large dataset, recently published and made available to the public. Our main goal and contribution is to combine Spotify similarity data with FMA data to form a useful dataset for artist similarity research and evaluation.

2.1 Identifying Common Artists

Spotify and FMA do not use the same audio files, song identifiers, or artist identifiers, and the datasets only partially overlap. We began by tabulating the artists present in the FMA dataset. Each artist from the FMA is searched for using the Spotify Web API online public portal. If the artist is found, then the Spotify Artist ID is stored. This ID is then used to get related artists from the Spotify Web API. The API returns a list of about 20 artists whose works are similar to those of the queried artist.

The first step is for a Spotify developer account to be set up, and an application to be registered which is used to query the Spotify API. There are various kinds of authorizations flows which let an application access the API, but we use the client credentials flow. It requires a secret generated key for every GET request. The key is provided by the Spotify Web API at the time of registering a new application. The Web API practices rate limiting to share access bandwidth across all applications, but using the client credentials flow enables access to higher rates, which is helpful because we have a large number of artists in the FMA dataset. Some artist names are anomalies and there can be multiple queries for the different possibilities of the same artist.

This project used 'Spotipy', a light-weight library in Python for the Spotify Web API.

FMA Artists in Spotify	Small	Medium	Large
Training - Overall	1031	1930	5137
Validation - Overall	123	228	393
Test - Overall	107	232	439

Table 2.1: Breakdown of Number of FMA Artists present in the Spotify Database.

FMA Artists in Spotify	Small	Medium	Large
Training - Full Set	396	920	2322
Validation - Full Set	51	109	149
Test - Full Set	45	108	224

Table 2.2: Breakdown of Number of FMA Artists present in the $fullset$.

2.2 Combining Data

Not all FMA artists are known to Spotify, and not all related artists returned by Spotify are in FMA, so we filter the list of artists in FMA to contain only artists known to Spotify, and we filter their related artists lists to contain only other artists within FMA. We refer to this result as the *full set*.

Spotify's related artists lists are not rated or ranked, so we take the simple approach that if Spotify lists artist B as one of the artists related to A, our 'ground truth' is "A is similar to B" and "B is similar to A." Thus, we define similarity as a symmetric binary relation as opposed to a continuous function describing degree of similarity.

The FMA dataset is divided into training, validation, and test datasets. Each one of these has three different sizes: small, medium, and large. Therefore, there are a total of nine subsets of the complete FMA dataset i.e., training (small, medium, large), validation (small, medium, large) and testing (small, medium, large). Taking the small training set as an example (Figure 2.1), there are a total of 1,885 artists to begin with. Each one of these artists is queried with the Spotify API to get an artist ID if the artist is present in the Spotify catalog, leading to 1,026 artists. The Spotify API is once again queried with each of those 1,026 Artist IDs to get a set of about 20 related artists for each one of them. Out of the 20 related artists for every artist, we only keep the ones present in the subset of 1,026 artists and discard the rest. If there is an artist who has empty related set after the previous process, then it is discarded as well. Therefore, we are left with 396 artists, and each artist is related to at least one other artist in the subset. This is leads to a 396 x 396 adjacency matrix describing the graph of how artists are connected to each other.

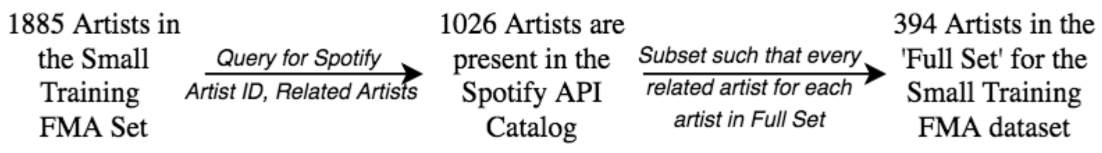

1885 Artists in the Small Training FMA Set
Query for Spotify Artist ID, Related Artists
1026 Artists are present in the Spotify API Catalog
Subset such that every related artist for each artist in Full Set
394 Artists in the 'Full Set' for the Small Training FMA dataset

Figure 2.1: Example Process for the Small Training FMA dataset to obtain the $fullset$.

2.3 Dataset Criteria

We could have stopped with the combined Spotify and FMA data (the full set), but we believe it is very desirable to provide data that is focused on artist similarity. Thus, we want each artist in the dataset to contain many similar artists, we want the artists to cover a wide range of genres, and we want all artists in the dataset to have substantial amount of content in the form of songs. Furthermore, we would like to see a balance in all these numbers. Ideally, every artist should have about the same number of similar artists, every artist should have a similar number of songs, and every genre should be represented by a similar number of artists.

Therefore, we decided that rather than taking every possible artist, we would strive to create a subset with as much data as possible within the soft constraints of offering good balance and coverage according to these three criteria related to,

1. similar artists per artist,
2. songs per artist, and
3. artists per genre.

2.4 Calculating the Goodness Score

To select an ideal subset of the full set of artists and songs, we define a "goodness score" G based on the criteria just described. Given an objective measure of overall quality, we could in principle try all subsets to find the best one in terms of G. However, it should be apparent that the number of subsets grows exponentially with the total number of artists, which is hundreds in the full set. Furthermore, we know of no fast optimization algorithm for interesting definitions of G.

Assuming that it is intractable to compute the optimal artist subset, we take an iterative 'greedy' approach to at least find a desirably balanced subset. Starting with a random subset of artists, we consider swapping each artist with one who is not in the subset. Given one artist in the subset, we swap with each artist not in the subset and calculate the resulting value of G. One of these swaps will maximize G, so we perform that swap to form an improved subset (no swap is performed if no improvement can be achieved). Then we move to the next artist in the subset and repeat the swapping procedure. By iterating through *all* the artists in the subset, we incrementally improve G. When a full pass through all the artists fails to find any improvement, we stop.

The initial artist subset is random, and we try to avoid getting "stuck" in local optima by running the optimization algorithm on different random initializations. There may be better approaches to optimization, but we will show below that this procedure is effective in making substantial improvements over random selection and that different random starting configurations tend to converge to roughly equal "goodness." In the following paragraphs, we explain how we define G, the measure of goodness that we are optimizing.

2.4.1 Formulation

One goal is to get a partition where classes in the partition are of uniform size. To measure variability is size, we typically use variance, the expected value of the squared deviation from the mean. Note that variance is monotonic with standard deviation, so we should think about which one is better. Also, variance should be expected to grow with the mean. For example, if we simply have twice as many items to group into N classes while keeping N fixed, the mean will double, the variance will increase four-fold. The coefficient of variation (also known as relative standard deviation, or RSD) is the standard deviation divided by the mean, and is scale invariant, so that seems like the thing we want.

RSD varies from 0 to $(\sqrt{N}-1)$, where N is the number of categories or classes.

We define a function $E(x)$, for equally distributed, where x is a vector of counts, e.g. number of songs per artist. Therefore,

$$E(x) = 1 - \frac{\sigma(x)}{\mu(x)(\sqrt{N}-1)} \tag{2.1}$$

where, E is 1 when elements of x are equal, and 0 if all elements but one are zero. N is the dimension x, i.e. the number of elements in x.

2.4.2 Similar Artists Criterion

S is supposed to measure the number of similar artists for every artist. We want artists to have as many similar artists as possible and we want the number of similar artists to be about equal. For the first part, let's start with vector a that tells how many similar artists there are for each artist ($dim\ a = N$). Then $E(a)$

goes from 0 to 1 for equal distribution, and let's multiply by the average number of similar artists per artist to get

$$S = E(s)\frac{\sum_a a}{N} \tag{2.2}$$

2.4.3 Distribution of Songs per Artist

D is supposed to measure the idea that artists have songs and the number of songs per artist is similar across all artists. To measure similar number of songs per artist, we can start with vector d that records the number of songs for each of N artists. Then $E(d)$ tells us how equally distributed they are but does not give credit for more songs overall. We can multiply by the number of songs per artist, so,

$$D = E(d)\frac{\sum_d d}{N} \tag{2.3}$$

where, N is the number of artists, i.e. number of elements in s.

2.4.4 Genre Coverage

C is supposed to measure the number of artists per genre. We want to have roughly equal distribution across all genres, so we measure that using g which is the number of artists in each genre. The number of elements in g is the number of genres G. Hence, $E(g)$ expresses the equality of distribution across genres. We also want more artists to consider it a better set overall, so

$$C = E(g)\frac{\sum_g g}{G} \tag{2.4}$$

While computing $E(g)$, we get,

$$E(g) = 1 - \frac{\sigma(g)}{\mu(g)(\sqrt{G} - 1)} \tag{2.5}$$

2.4.5 Overall Goodness Score

Overall, we have

$$G = D \times S \times C \tag{2.6}$$

The reason for multiplication is that, firstly, D, S, and C are not normalized to 1. Therefore in the expression $D + S + C$, one component might dominate the rest. Secondly, because a balance of qualities is desired, and the product has the property that if any one of the terms is close to zero, it becomes the most important term. If, for example, D is close to zero but S and C are much larger, and there is some way to improve D to ($D + \epsilon$), then that is going to improve the overall G, even if it reduces S and C by ϵ.

Chapter 3

Experiments, Results and Analysis

In this chapter, we first present the experiments conducted to choose the best subset based on the criteria discussed in previous chapters. We will examine the behavior of the optimization function being used, and finally we look at the various individual components of the goodness formula and how they contribute to the overall score.

3.1 Optimization Algorithm

3.1.1 Estimating the best N

N refers to the number of artists in our set which we obtain using our goodness score formulations. For each dataset, and for many values of N, we run our optimization algorithm to estimate the best set.

Let us take the FMA small training dataset as an example. It has 393 artists in the $full\ set$. To calculate the best subset size, the goodness score needs to be calculated for every smaller subset size so that we can compare and choose the best one. We start at $N = 50$ and initialize our set with 50 randomly selected artists. Now, we take each artist in this selected subset of 50 and replace it with one of the 343 artists not present in this set, that maximizes the goodness score of the selected set. This is an iterative process. We repeat the whole process until we have converged to the maximum goodness score for the set of 50.

The same process is carried for a selected set of 60, and then 70, 80, and so on up to 390. For the small training FMA dataset, the step size is 10, but it is different for every dataset dependent on the number of artists present in it. The overall objective is to obtain about 35 to 40 different Ns. At the end we are left with best goodness values for each of the N, and then we choose the best N, based on these goodness values. The same algorithm is used for each of the FMA datasets (small, medium, and large) for training, validation, and testing.

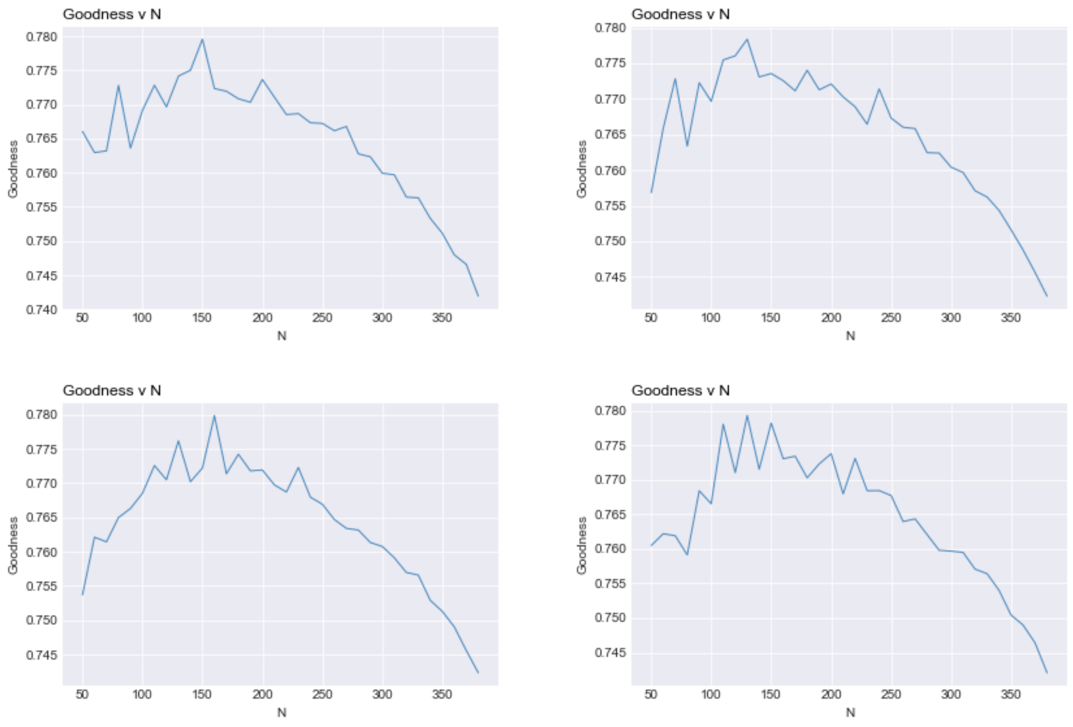


Figure 3.1: Goodness vs N, an example of four different runs of the optimization algorithm for the small training FMA dataset.

In 3.1, we have an example of four different runs of the above-mentioned algorithm for the small training FMA dataset. As we can see, we obtain a range of best values for N in each of iteration, with high goodness values for high and low values of N. There is no clear winner and this performance is as expected. Ideally, we would want our optimization process to give higher goodness values to a range of sizes, with a peak at a certain size, while giving low scores to the either extremes (i.e., N is either too big or too small). In the figure we can see the peaks lie within a certain range.

3.1.2 Multiple Iterations of the Optimization Algorithm

We run the optimization algorithm multiple times on the same dataset. For the small training FMA dataset, we run our algorithm for 25 times, and 20 times for the medium training FMA dataset. and 10 iterations for the large training FMA dataset. These numbers were decided based on a couple of factors. First, these computations take a lot of time and resources, and these numbers were the largest practical values we could manage. Second, the number of comparisons made is very large for each of the dataset and gives us a reasonable and acceptable representation of the nature of optimization function we are trying to approximate here.

In Figure 3.2, we can see how 25 different iterations of the optimization algorithm on the small training

Dataset Size	**Dataset Type**	**Full Set Size**	**Step Size**	**Number of Observations**
Small	Training	393	10	34
Small	Validation	51	2	20
Small	Test	45	2	19
Medium	Training	920	75	12
Medium	Validation	109	5	24
Medium	Test	108	5	23

Table 3.1: Optimization algorithm parameters for different datasets.

FMA dataset compare with each other. The overall pattern observed holds for all the iterations. However we do get different values of best Ns. The range of the best Ns, which lies approximately in the middle-left section, almost looks noisy, not giving one true peak or clear global optimum. This can be due to a couple of reasons. First, our optimization function which calculates the goodness score gets pretty close to the optimum, as showcased by the consistency of convergence, it does not approximate the true function. Second, the optimization function is fairly flat, within 1%, in the range of best converged Ns that satisfy the conditions used to build the goodness score formulations. More light is shed on this topic later in Section 3.2.

Figure 3.4a shows average goodness values for each N. For the small training FMA dataset, we ran the iterative process for 25 iterations. Mean value for those 25 values for each N was calculated and is shown here. Similarly, Maximum value out of those 25 values for each N was selected and is shown in Figure

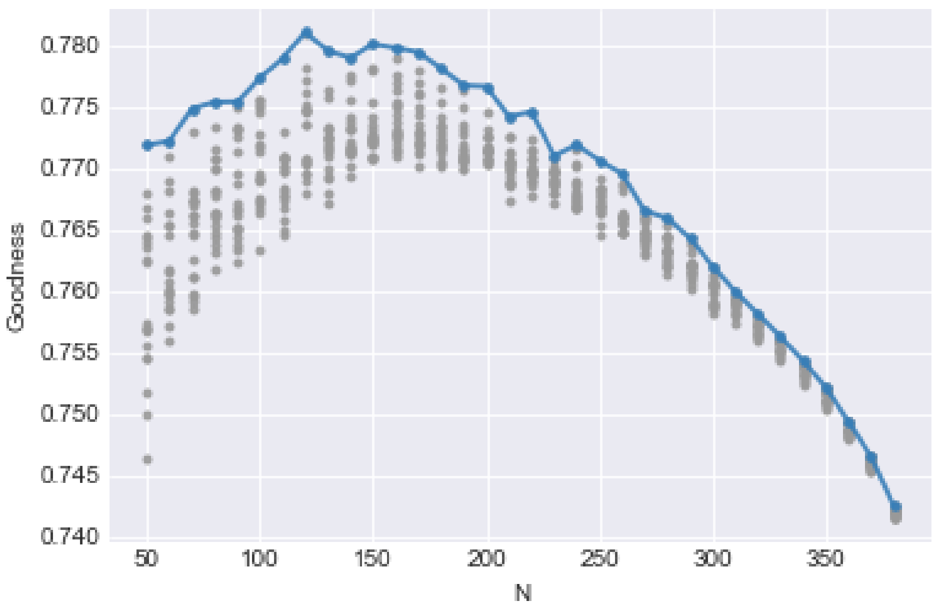


Figure 3.2: Goodness vs N, 25 iterations for the small training FMA dataset.

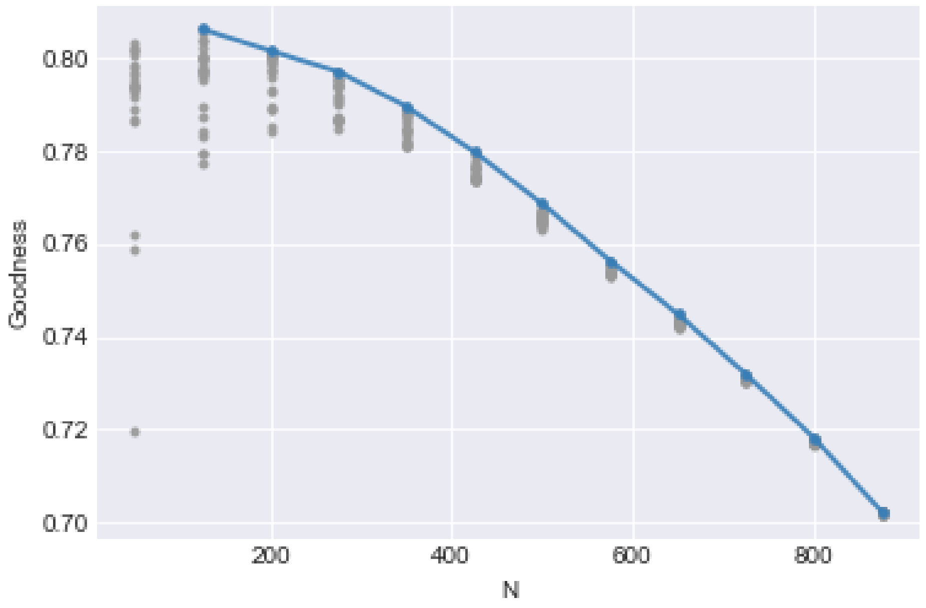
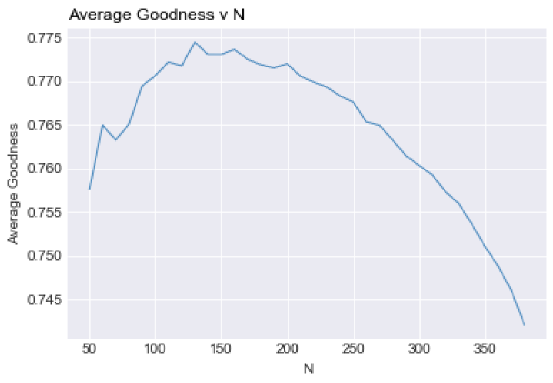
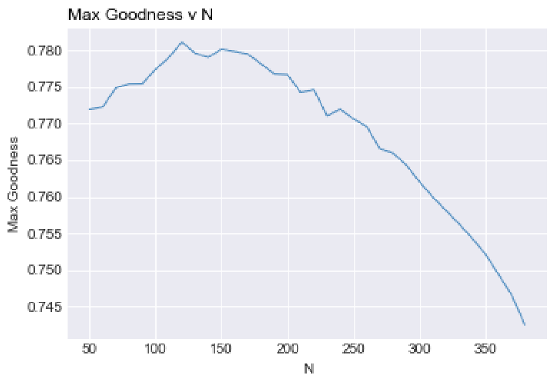


Figure 3.3: Goodness vs N, 20 iterations for the medium training FMA dataset.



(a) Average Goodness vs N.



(b) Maximum Goodness vs N.

Figure 3.4: (a). Maximum Goodness vs N, and (b) Average Goodness vs N, for 25 iterations for the small training FMA dataset.

3.4b. Average goodness values give a smoother curve and and can be used to better estimate the general trend of best goodness score for each N. Maximum Goodness represents only the best values. Therefore, to select the best value of N and the corresponding best set, we just choose the maximum value available to us in this graph, which is 120 for this dataset.

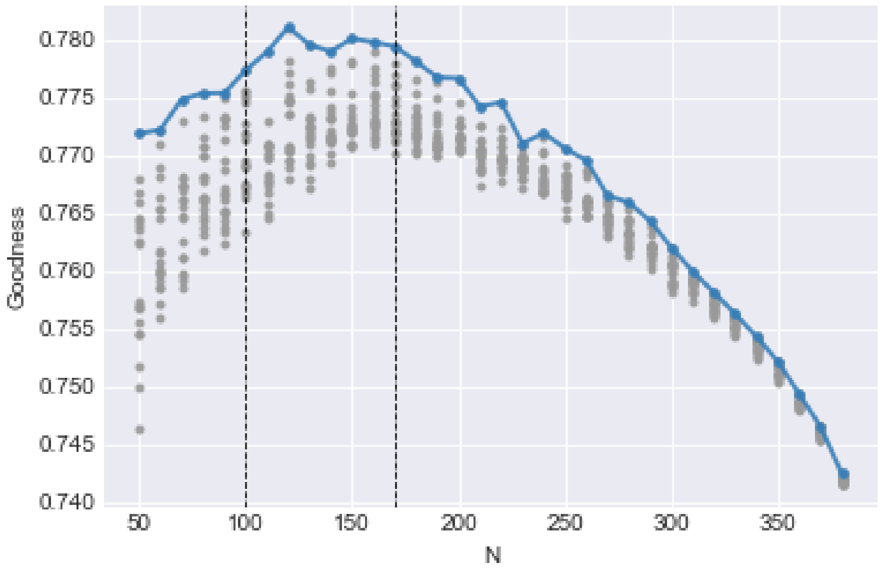


Figure 3.5: Maximum Goodness vs N, 25 iterations for the Small Training FMA dataset. The vertical lines enclose the region of best *N*s for any iteration.

3.2 Nature of the Optimization Function

In Figure 3.5, we see two vertical lines enclosing an area marked by different values of *N*. These vertical lines mark the range of best values of *N* observed in the 25 iterations of the optimization algorithm on the small training FMA dataset. These values are 100 and 170 respectively.

There are a couple of hypotheses we put forward to explain why there is so much noise and no clear peak value in the enclosed region.

(i) Consistency of convergence

(ii) Observed flatness in the enclosed region

3.2.1 Consistency of Convergence

First, our iterative optimization algorithm reaches pretty close to the global optimum. This is evident by the consistency in convergence observed in Figure 3.6 over the large number of iterations we ran for a given dataset. For example, in 25 runs to compute the best set (as shown in Figure 3.5), the resulting goodness values vary by only about 1%. This suggests that the algorithm always finds a set within about 1% of optimal. Of course, we do not know the true optimum goodness, but for the sake of argument, suppose the algorithm only finds goodness values around 90% optimal. If there are so many local optima at least 10% away from optimal, one would think that the algorithm would sometimes find much worse local optima, say at 80%. Since outliers like this are never seen, it seems most likely that the algorithm is converging close to optimal sets.

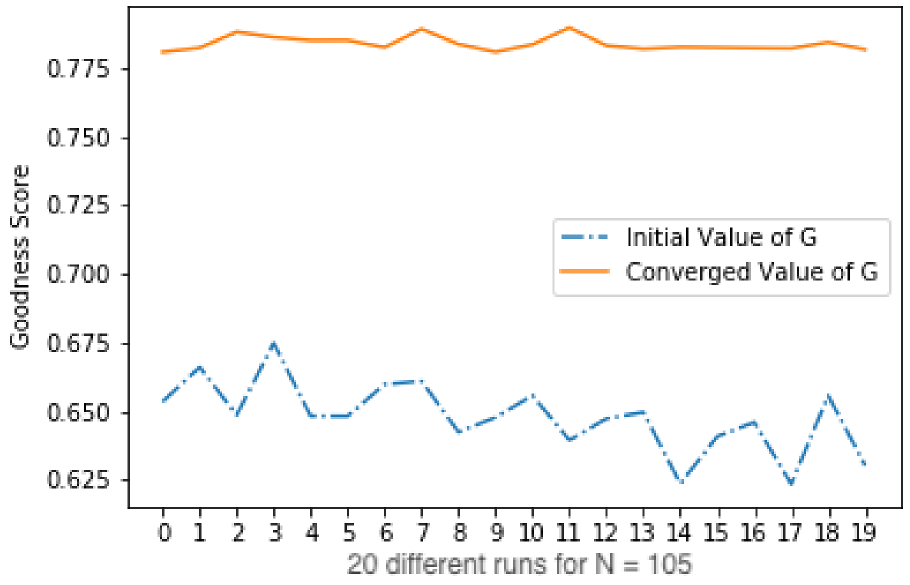


Figure 3.6: Starting G vs Converged G for 20 runs on the Small Training FMA dataset.

We fixed the value of N at 105, and ran our algorithm 20 different times on the FMA Small Training dataset. Figure 3.6 shows the starting values of G, which corresponds to the Goodness score for a randomly selected set of 105 artists out of a possible 394. The algorithm then iteratively improves the value of G based on the formulations in our algorithm, until it converges, depicted with a continuous line in the figure. Notice that regardless of the initial value of G, the algorithm finds a substantially better set with values of G all within a span of about 1% for the 20 times we ran our algorithm. This could of course be just a coincidence, but again, the most likely explanation is that the optimization is robust and consistently approaches close to the true optimum. Therefore, the consistency of the results support the conclusion that the algorithm finds sets with goodness scores withing about 1% of the optimal value.

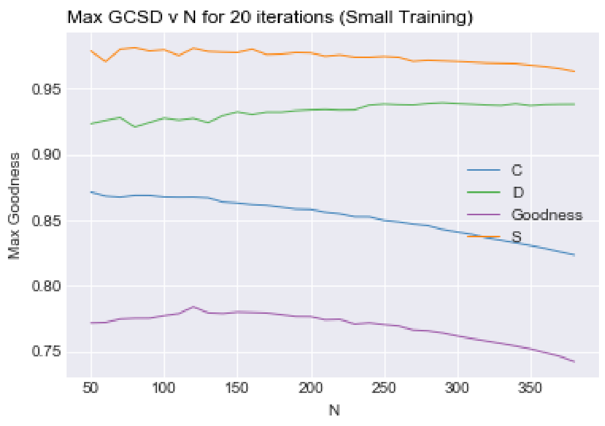
3.2.2 Observed Flatness in the Enclosed Region

Second, the true function is fairly flat in the enclosed region as observed in Figure 3.5. The range of converged values that we obtain, within 1% in the enclosed region, but not elsewhere, supports this point of consideration. Having performed multiple passes of calculation over the entire dataset and not finding any value of N with a clearly higher goodness, we believe the optimal goodness is fairly flat with respect to N, so a range of choices is reasonable. The three criteria that form the basis of our Goodness function - 1) An equitable distribution of songs among the artists in the final set, 2) Maximum genre coverage within the set with respect to the number of artists for each genre, and 3) more number of related artists, will hardly ever choose one particular value of N to be significantly better than at $N + 10$. All in all, it appears that a range of values of N will result in near-optimal goodness values, so we can consider other criteria in choosing N within this range. One possibility would be to choose N at the high end of the range, simply on the basis that more data is better. However, we decided to pick N that maximizes G. At least, this removes the somewhat subjective decision making. In the next section, we look at the S, C and D components of G since that reveals some of the trade-offs between different factors.

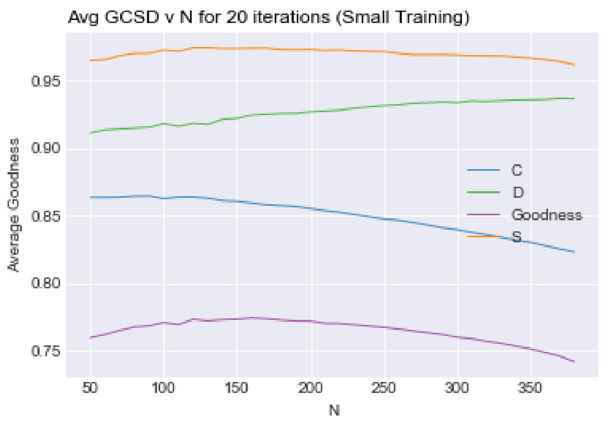
3.3 Individual Components of Goodness

Recall, that the Goodness function is based on the following,

1. similar artists per artist, hereafter referred as S in this section,
2. songs per artist, hereafter referred as D in this section
3. artists per genre, hereafter referred as C in this section



(a)



(b)

Figure 3.7: (a). Maximum Goodness, C, D, S vs N, and (b) Average Goodness, C, D, S vs N, for 20 iterations for the small training FMA dataset.

In Figure 3.7, we can observe how the values of these individual components of the goodness calculation vary with the overall goodness score for any size N. There are two representations, in (a), we plot the maximum goodness values and its corresponding C, S and D values, where as in (b) we plot the same values but for the average goodness scores observed. The patterns observed for all the individual scores are identical, and therefore can be assumed to be the same for any further analysis. Based on these individual values, we can choose our value of N.

It is interesting to note how the three values vary over N. We observe a general pattern of C going down. This is because as the number of artists increase in the set, the bias toward certain popular genres increases. The formulation brings down the score of equitable genre coverage as expected. D which corresponds to the equitable distribution of songs among artists, slightly increases over N. This is also as expected because as the number of artists increase, we get more artists with approximately similar number of songs in the best set. D plays a part in the goodness score and rewards this property, more number of artists with similar number of songs. Lastly, S increases and then decreases. For smaller values of N, we do not have enough number of artists to have enough associativity; however, as N increases we get more similar artists in set, but after a certain point, we start adding artists who may not be very similar to other artists. Essentially, we see that the amount of similarity in a set, peaks out and the starts decreasing with increasing N.

Computing G using a weighted combination of C, S and D

Going back to the example of the FMA small training dataset, as shown in Figure 3.5, the enclosed region lies within values of 100 and 170 of N.

We give equal weights to all the components of the formulation to come up with the Goodness score G in our algorithm described so far. However, it is possible to prioritize one or two components over the others. For example, we might try to prioritize S, which is the number of artists related to each other in the set, if we need a best set comprising of more related artists. Similarly, we can prioritize D, which is the equitable distribution of songs among artists, and/or C, which is the number of artists per genre and hence, genre coverage. This can be achieved by using exponents for these individual components when multiplying them to calculate G.

Therefore,

$$G = D^x \times S^y \times C^z \tag{3.1}$$

Here, x, y and z are the exponents used to provide weights to D, S and C respectively. If we just analyze the above equation taking D for example, when the value of x approaches 0, the value of D approaches 1, and hence minimizing its contribution to G. As x approaches 1, D's contribution reaches equal in weight to other components (keeping the values of y and z fixed at 1). Beyond that, as we increase the value of x

towards positive infinity, we increase the weight of D. Similarly, we could adjust the exponents for S and C.

3.4 Runtime Analysis of the Algorithm

For every value of N, the algorithm starts off with a randomly selected set and then iteratively improves it by maximizing the value of G for the set. Each artist in the chosen set is replaced with every artist not in the set, one by one, and the artists which improve the value of G for the new set, are retained. Therefore, the algorithm performs a total of $N \times (M - N)$ computations to calculate G, where M is the total number of artists in the full set. Every time we replace an artist in the selected set with an artist not in the set, we recalculate values of S, D and C, and hence G. For each of those individual components, we get the worst case time complexity of O(N). Hence, for each pass over the selected set of N artists, we get a time complexity of O($N^2 \times (M - N)$). However, the algorithm performs more than a single pass. After observing various iterations of running the algorithm, we found that to converge the value of G our algorithm took a maximum of three complete passes over the dataset. Therefore, there is no theoretical upper bound available but in practice we know the upper bound for a single pass and that the algorithm takes a maximum of three passes. The implementation uses optimized inner loops for fast computations through matrix multiplications. The data structures are implemented using numpy, a python library for fast matrix calculations. Currently, taking the example of the FMA Small Training Dataset, the maximum time taken for one iteration of the algorithm over the selected set, is for N = 180 (78.08 seconds), where M is 394. The runtime scales in proportion to $N \times (M - N) \times N$.

In Figure 3.8, our complexity formula for a single pass over the dataset, O($N^2 \times (M - N)$), is plotted along with the actual runtime. We scaled the complexity to approximately match the measured runtime. As the figure shows, the actual runtime is higher at the extreme values of N, possibly because there is additional work in the algorithm outside the innermost loop, e.g. loading data from disk, that becomes less important as the $N^2 \times M$ term begins to dominate.

By step size, we mean the granularity at which the best value of N was calculated. For example, for the FMA small training dataset, N was calculated at a granularity of 10 i.e., at N equaling 50, 60, 70, and so on up to 390, which is the maximum number of artists in the $full\ set$. In the future, it would be beneficial to calculate the best N at a smaller step size for all datasets.

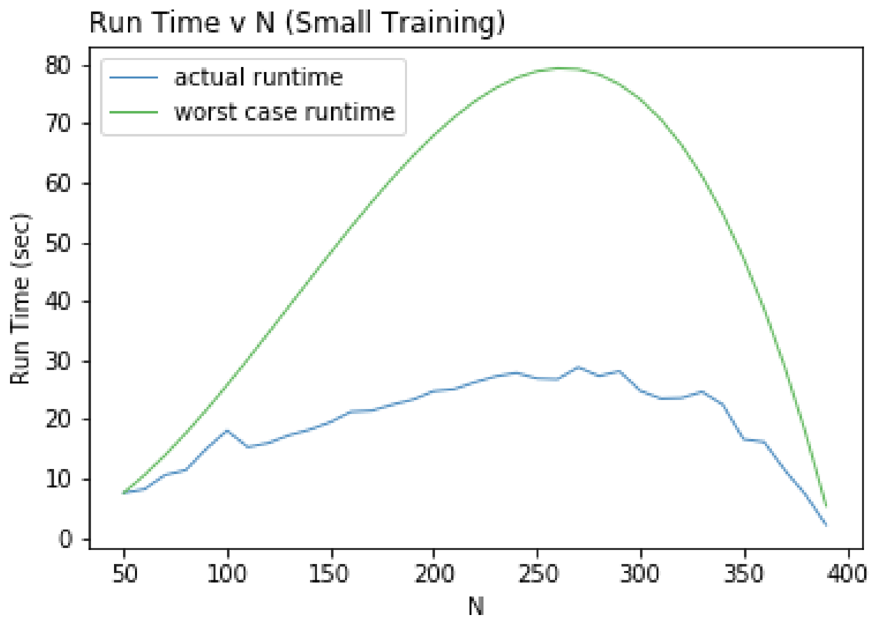


Figure 3.8: Actual Runtime, Worst Case Runtime vs N, for one pass over the FMA Small Training Dataset.

Estimating Runtime for the FMA Large Training Dataset

Overall, the algorithm takes about 42 minutes for one run over the dataset to calculate the best set, at a step size of 10, resulting in 34 values of N. We run the algorithm 25 times, thereby taking about 17.5 hours on the FMA Small Training dataset, on a single machine. It took us about 41 hours to run 20 iterations on the the FMA Medium Training dataset, with a step size of 75, resulting in 12 values of N. For the FMA Large Training dataset, we estimate the 5 iterations of the algorithm to take about 194 hours with a step size of 175, resulting in 13 values of N. The calculation of this number is specified below.

For $N = 50$, $M = 394$, the observed runtime is 8.6736 seconds. Let k represent the seconds per computation, then we have

$$N^2 \times (M - N) \times k = 8.6736 \tag{3.2}$$

By plugging in the values for N and M in the equation above we get $k = 1.008 \times 10^{-5}$

Similarly, for $N = 180$, while M remains to be 394, we get k to be 1.13×10^{-5}. We take the geometric mean of these values of k in order to normalize the range of k we are dealing with and not give too much weight to any one large value. This gives us the value of k to be 1.02×10^{-5}.

The FMA Large Dataset has a total of 2322 artists, and the value of N starts from 150, with a step size of 175, resulting in 13 values.

Using the obtained value of k, we calculate the runtime for calculating G for each value of N in seconds using the Equation 3.2, to get a total of 2329 minutes for a single run of the optimization algorithm (as described in Section 3.1.1). This is to estimate G for different set sizes and then to choose a set, but it would be even better to make at least several runs with each value of N to see if we get consistent convergence of G. To make 5 runs for each value of N, we would need about 200 hours of CPU time. Therefore, in the future, we plan to run the code on cloud computing services likes Amazon AWS or Google Compute Engine, or setting up another physical machine to run the algorithm.

Chapter 4

Evaluation Baseline and Framework

We envision that users of this dataset will construct, train, and evaluate artist similarity algorithms and systems. In order to compare algorithms, it will be useful to have standard evaluation functions to rate the quality of proposed systems. In addition, we provide an example, or ”baseline” artist similarity system and its evaluation. This can illustrate how to run our evaluation functions and also serve as a simple system against which improved systems can be compared.

In this chapter, Section 4.1 discusses the evaluation baseline used on the RASM dataset and the details of the implementation used. In the second half of the chapter, Section 4.2 looks at the evaluation framework that will be made available with the dataset to users of the dataset for comparing their own models against the ground truth.

4.1 Baseline for Finding Similar Artists

4.1.1 Approach

Measuring artist similarity algorithmically can be a complicated process at times, simply because of the different ways and perspectives which can be taken to approach it. Therefore, in our baseline we wanted to implement an intuitive approach (Figure 4.1), which would be straightforward, make sense and work well.

The objective of our artist similarity system is to account for each song of an artist toward building a global representation of the artist in terms of a feature vector in the given dataset. Intuitively, this representation should incorporate a certain underlying discriminative factor which makes two artists similar or dissimilar. These discriminative factors could relate to the musical style of an artist, general energy, type of melodic content, and so on. Therefore, for each track of the artist, certain features, as

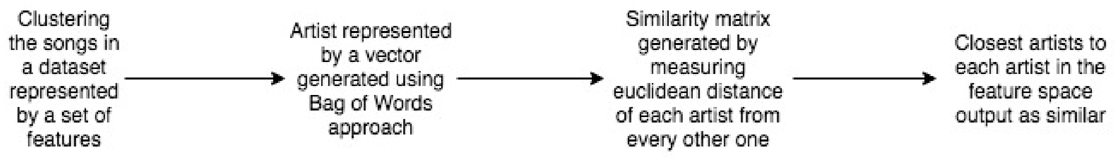

Clustering the songs in a dataset represented by a set of features
Artist represented by a vector generated using Bag of Words approach
Similarity matrix generated by measuring euclidean distance of each artist from every other one
Closest artists to each artist in the feature space output as similar

Figure 4.1: Evaluation baseline used by the algorithm.

described later, are used to represent it. For the small FMA dataset, about 5,000 tracks are used. The vector representing each track then undergoes clustering on the set of features. This approach is inspired by the bag of words model, and after the clustering step, a vector representation for each artist is generated, the length of which is equal to the number of clusters used. This vector representation is generated by counting the number of songs of the artist in each cluster, and then finally, the whole feature vector is normalized.

4.1.2 Features

Mel-Frequency Cepstral Coefficients

The first step of a music information retrieval model like ours is to extract features. The purpose of this step is to identify the components of the audio signal that accurately represent the sound of an artist and discard all the other information.

MFCCs are a feature widely used in ASR systems. They were introduced by Davis and Mermelstein in the 1980s and have been widely used ever since. Previously, linear prediction Coefficients (LPCs) were widely used, especially with Hidden Markov Model based classifiers. Although MFCCs gained popularity in realm of speech and speaker recognition research, this feature is extensively used in genre classification systems as well. It is typically believed to encode timbral information because it represents short duration musical textures. MFCCs are known to be not the most invariant feature for key and tempo, but it still performs consistently in any system which is related to musical content aspects like genre, mood, and style.

As shown in Figure 4.2, to get the MFCCs for an audio file we first frame the signal into short frames as the audio signal is constantly changing. The power spectrum is calculated for each of these frames, thus identifying the present frequencies. We then convert it to the mel-frequency spectrogram which closely emulates how humans tend to hear. The mel filterbank is applied on the power spectra, and the energy in each filter is summed. We then take the logarithm of all the filterbank energies. Because the used filterbanks are overlapping, the energies are quite correlated. Therefore, we take the Discreet Cosine

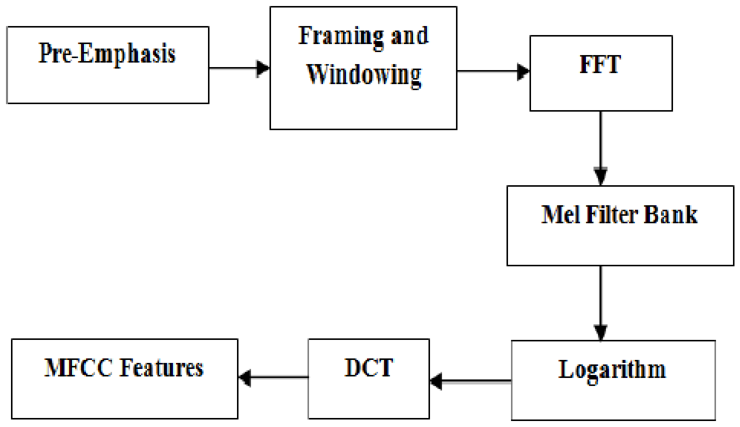


Figure 4.2: Algorithm to obtain MFCC features (source: ResearchGate).

Transform of it to decorrelate them.

The FMA dataset comes with MFCC features pre-calculated for each of the tracks present. Statistics of these MFCCs feature is calculated for every song leading to a fixed size vector output independent of the song length. The statistics provided by the FMA and used for the purposes of our baseline are kurtosis, maximum, median, mean, standard deviation, minimum, and skewness. There are 20 values for each corresponding to 20 coefficients.

Spectral Centroid

The spectral centroid is a measure used to represent the ”center of mass” or ”center of gravity” of a spectrum in digital signal processing. When taken in the context of a sound signal, it can also be thought of as a feeling of the ”brightness” of sound.

Mathematically, it is the weighted mean of the frequencies present in the signal output by the Fourier transform, with their magnitudes incorporated as weights. The individual centroid of a spectral frame is defined as the average frequency weighted by amplitudes, divided by the sum of amplitudes,

$$SpectralCentroid = \frac{\sum_{k=1}^{N} f[k]x[k]}{\sum_{k=1}^{N} x[k]} \tag{4.1}$$

where $x[k]$ corresponds to the magnitude of bin k, and $f[k]$ is the central frequency (in Hz) of bin k in DFT spectrum.

Some people use ”spectral centroid” to refer to the median of the spectrum, which however is quite different from the mean of it. Although there might be similarity of behavior, the performance obtained can be very different. The centroid is much higher that one might expect because there is much more energy above the lower fundamentals which contribute significantly to the average. Because the spectral centroid is an effective predictor of the ”brightness” of a sound, it will show some spectral components of music and is widely used in digital audio and music processing as an automatic measure of musical timbre.

We use the following spectral centroid statistics for each song: kurtosis, maximum, median, mean, standard deviation, minimum, and skewness.

Root Mean Square Energy

The root mean square energy is most often used to characterize a sound wave because it is directly related to the energy carried by the sound wave, which is called the intensity. In the context of music information retrieval, the intensity can be used as a discriminator between two very different genres, styles, or even artists. The intensity of a sound wave is the average amount of energy transmitted per unit time through a unit area in a specified direction. The intensity I of a sound wave is proportional to the average over time of the square of its pressure p.

We use the following root mean square energy statistics for each song in the FMA dataset: kurtosis, maximum, median, mean, standard deviation, minimum, and skewness.

Zero Crossing Rate

As per its definition on wikipedia, the zero-crossing rate is the rate of sign-changes along a signal (i.e., the rate at which the signal changes from positive to negative or back). Although this feature has been used heavily in speech recognition, it is also useful in music information retrieval for classification of percussive sounds.

One of the applications of zero-crossing rate is to identify to silence of unvoiced sections from the voiced ones [11]. For example, a song may have more quiet sections than others, and this can be indicated by a higher zero-crossing rate. In some cases, only the positively directed or negatively directed crossings are counted, rather than all the crossings, because logically, between a pair of adjacent positive zero-crossings, there must be one and only one negative zero-crossing.

For monophonic tonal signals, the zero-crossing rate can be used as a primitive pitch detection algorithm. Although it is highly unreliable, it can be used in conjunction with an energy representation for music information retrieval tasks as well.

We use the following zero-crossing rate statistics per song in the FMA dataset: kurtosis, maximum, median, mean, standard deviation, minimum, and skewness.

Using the features in the model

The above-mentioned features are stacked together for every track and used in the classification process. However, there are three different groups of features based on what the features represent, and how they are known to work together as per commonalities between what they represent and general usage in various research models in MIR. For example, spectral centroid has been known to work well when used with root mean square energy feature for music information retrieval tasks [12].

1. SCen + ZCR + RMSE : Spectral Centroid, Zero Crossing Rate, Root Mean Square Energy
2. SB + SCon + SR : Spectral Bandwidth, Spectral Contrast, Spectral Roll Off
3. MFCC : Mel-Frequency Cepstral Coefficients

These are different feature sets used for the baseline and their performance measure are presented later in this chapter.

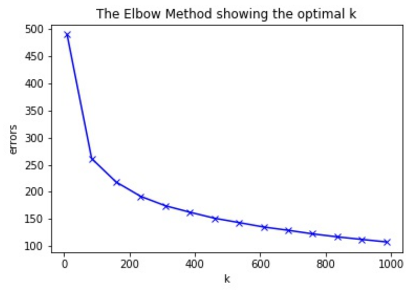
4.1.3 Unsupervised Clustering

K-Means

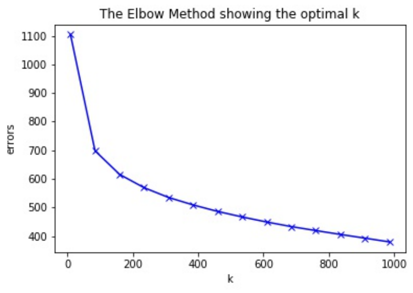
Unsupervised learning is used when we have unlabeled data (i.e., data without defined categories or groups) to learn an underlying distribution or classify data based on some emergent property. K-Means clustering is a type of an unsupervised learning algorithm, which can be used to find groups in the data based on similarity in the latent space of features of data, with the number of groups represented by, say variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm [13] are:

- The centroids of the K clusters, which can be used to label new data
- Labels for the training data

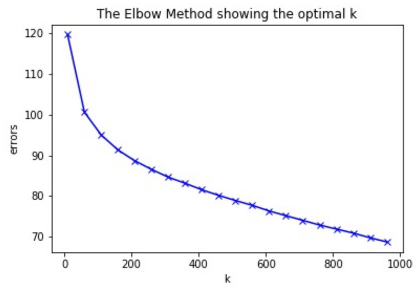
This method of unsupervised clustering allows us to find and analyze the groups that have formed organically based on the input features. However, the value of K needs to be input into the algorithm, and there are various ways to choose the best value. For our baseline implementation, we use the elbow method [14] to choose the number of clusters. We used three different sets of features, obtaining slightly different best value of K for each, as shown in Figure 4.3.



(a)



(b)



(c)

Figure 4.3: Elbow method results for choosing the best value of K. (a) SCen + ZCR + RMSE, K = 60, (b) SB + SCon + SR, K = 70, (c) MFCC, K = 65.

Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

4.1.4 Bag of Words Approach

In a bag of words model, a sentence or a document is considered to be a 'bag' that contains words. It will take into account the words and their frequency of occurrence in the sentence or the document disregarding semantic relationship in the sentences. In a practical setting, this model is mainly use for feature generation. In the field of Natural Language Processing, after transforming the text into a bag of words, the term frequency (many other characteristics can be used as well) (i.e., number of times a term appears in the text), is calculated. Each document is then represented by the list of counts, also known as histogram representation.

In the context of our evaluation baseline, we use this approach to count the number of tracks present for an artist in a specific cluster. This vector, the length of which equals the number of clusters, is then normalized by the total number of songs per artist. This normalized vector forms the feature representation of the artist which is then used for rest of the process.

4.1.5 Finding Similar Artists

Measuring similarity between two data objects is the very basic building block for activities such as recommendation engines, clustering, classification, and anomaly detection. The similarity measure is the measure of how much alike two data objects are, depending on how information is presented. Similarity measure in a data mining context is a distance with dimensions representing features of the objects. If this distance is small, then there will be a high degree of similarity where large distance will be the low degree of similarity.

Distance Metric

Euclidean distance is the distance between two points in any number of dimensions - the square root of the sum of the squares of the differences between the respective coordinates in each of the dimensions. It is the most commonly used distance metric for vector spaces.

Cosine similarity metric finds the normalized dot product of the two attributes. By determining the cosine similarity, we would effectively try to find the cosine of the angle between the two objects. According to wikipedia, the cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].

However, we use Euclidean distance between two artists for our purpose here as we found it to yield better results and was robust through the different feature sets. As an example, a comparison of results for the feature set of $SB + SCon + SR$ is showcased in Table 4.1. These vectors are a result of the bag of words approach i.e., a count of the clusters for each artist, which are then normalized.

Features	Prec.	F1 Score	Corr.
Cosine	0.2391	0.2706	0.2661
Euclidean	0.2820	0.2951	0.2892

Table 4.1: Resultant Values of Metrics for the Two Distance Metrics. Thresholding Method - Mean of the Distance.

Similarity between Artists

The similarity is subjective measure and is highly dependent on the domain, application and context. For example, two fruits are similar because of color or size or taste. Care should be taken when calculating distance across dimensions/features that are unrelated. The relative values of each element must be normalized, or one feature could end up dominating the distance calculation. Similarity is measured in the range 0 to 1 [15].

Similarity in the context of our baseline refers to how similar two artists are based on the distance between their vector representations. The method described until now in this chapter builds up on how the features are used to come up with clusters, and each artist is represented by a mix of different cluster present in their catalog of tracks in the dataset. We have the distance of each artist from every other artist. To categorize two artists as similar we need some type of a threshold to divide the artist set as similar versus not similar for each artist in the set. To achieve this, for every artist, we ordered every other artist based on decreasing proximity. Top k artists were then chosen as similar, and rest were regarded as dissimilar. This value of k come from the ground truth where we know how many artists are categorized as similar for every artist. Another approach was attempted where we chose the mean of the distances from an artist to every other connection as the threshold, and every artist which closer than the mean value was labeled as similar. This process was repeated for every other, thereby obtaining a similarity matrix. A comparison of these two approaches is discussed in Section 4.2.2.

4.2 Evaluation Framework for RASM

The RASM dataset comes with its own evaluation framework which can be used to compare a model's output to the ground truth. An evaluation framework is important because it serves as a standard way to compare and analyze the performance of a model. These metrics are discussed below.

4.2.1 Metrics Used in the Evaluation Framework

Root Mean Squared Error

The root mean square error (RMSE) is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed (i.e., ground truth), as per the definition seen in wikipedia. The RMSE represents the sample standard deviation of the differences between predicted values and observed values. Also, RMSE serves to aggregate the magnitudes of the errors in predictions for various times into a single measure of predictive power. This value is calculated per artist, and globally comprising of all the artists. RMSE can also be seen as a measure of accuracy, to compare forecasting errors of different models for a particular dataset and not between datasets, as it is scale-dependent.

As per wikipedia - RMSE is the square root of the average of squared errors i.e., errors of each prediction. The ground truth values are either 0 - two artists are dissimilar, or 1 - two artists are similar. The model output values are continuous, thereby encapsulating a measure of confidence in calculating similarity between artists. Therefore, RMSE makes it possible to compare two models based on confidence in a quantifiable manner. The effect of each error on RMSE is proportional to the size of the squared error. Thus, larger errors have a disproportionately large effect on RMSE. Consequently, RMSE is sensitive to outliers. In our context, the property of the RMSE is especially useful, as this serves as a good measure for training and improving a model iteratively.

In the RASM evaluation framework, we use the mean squared error for training and validation error. Choosing the best machine learning model for a problem at hand has been an area of research in the field of machine learning for a long time now. Using re-sampling methods such as cross validation, we can get an idea of how accurate a model may be on unseen data. RASM comes with nine different sub-divisions, divided based on the combination of size (small, medium, and large) and type (training, validation, and testing).

Any of the training datasets can be used for training a machine learning model to reduce training errors. The training error can be calculated in various ways when comparing the model output to the ground truth. The validation and test datasets will be used to compare a machine learning model's output after it has been trained on the training datasets. It is expected as is standard, that the validation error will reduce as well, just like the training error over the entire training period. A model's final output on the test dataset will used to assess its performance, and we use multiple metrics, as described below, to bring out how accurate a model's output is to the ground truth.

Other Output Metrics

Apart from the RMSE value, the evaluation framework outputs the following metrics [16] as well:

1. True Positives (TP) - This refers to the correctly predicted positive values (i.e., the value of both the actual class and predicted class is yes; e.g., when the actual class value indicates *Artist A* and *Artist B* as similar and predicted class depicts the same thing).

2. True Negatives (TN) - These are the correctly predicted negative values (i.e., the value of both actual class and predicted class is no; e.g., when the actual class indicates *Artist A* and *Artist B* as dissimilar and the predicted class depicts the same thing).

3. False Positives (FPs) - It indicates values where the actual class is no but the predicted class is yes (e.g., if actual class says *Artist A* and *Artist B* are dissimilar but predicted class showed that they are similar).

4. False Negatives (FNs) - It indicates values where the actual class is yes but predicted class in no (e.g., if actual class says that *Artist A* and *Artist B* are similar but predicted class showed that they aren't).

5. Accuracy - It is the measure of the effectiveness of the machine learning model

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

6. Precision - Precision is the ratio of correctly predicted positive values to the total predicted positive values. This metric highlights the correct positive predictions out of all the positive predictions. High precision indicates low false positive rate

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

7. Recall - The recall is the ratio of correctly predicted positive values to the actual positive values. The recall ratio highlights the sensitivity of the algorithm (i.e., how many of the actual positives were identified by the program).

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

8. F Score - (or the F1 score) It is the weighted average of precision and recall. At first glance, F1 might appear complicated. It is a much more sophisticated metric than accuracy because it takes both false positives and false negatives into account

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.5}$$

9. Correlation - The Pearson correlation coefficient is a statistical measure used for evaluating how linearly correlated two variables are. It has a value between +1 and -1 where +1 is total positive linear correlation and -1 is the opposite. The input artist similarity matrices are flattened and then the correlation coefficient is calculated.

These are the metrics that the evaluation framework outputs, in addition to the RMSE, that can be used to evaluate a model's performance against the ground truth. In Section 4.2.2, we discuss the value of these metrics when the evaluation baseline output is compared against the ground truth.

Using the Evaluation Framework

All the code for the RASM evaluation framework is publicly available, and hence there are a few different ways in which the framework can be utilized.

First, the *rasm_framework.py* file can be used as a script and run from the command line. It takes in multiple input, and the '-h' option will list out all the possible options and inputs, and how to use the script. However, the script essentially takes in two NumPy arrays, corresponding to the model's output similarity matrix, and the ground truth, and outputs a bunch of metrics as described in Section 4.2.

Second, the evaluation framework code is also available through Jupyter Notebooks, and hence, certain sections of the code can be incorporated in the user's model for calculating errors, and performance. At the same time, relevant functions can also be imported from the above-mentioned script *rasm_framework.py*.

4.2.2 Results of the Evaluation Baseline

We ran our baseline algorithm, as mentioned in Section 4.1, on the FMA small training set. The output of the baseline model was then fed into the evaluation framework, as described in Section 4.2 and results were obtained. We worked with all the artists in the FMA small training set to showcase the results on the $full\ set$ that we have used for all examples in this document so far, and below are the results obtained.

Table 4.2 showcases the results of the metrics obtained from the evaluation framework when the threshold for selecting similar artists in the evaluation baseline model is mean of the distance as the thresholding criteria for selecting similar artists to an artist. As we can see, the precision values are quite low and this is due to the large number of false positives. However, the true positive rate or recall is around 0.5 which indicates a low number of FNs. Therefore, in summary, the model does not miss out on many truly similar artists, as compared to the ground truth, but it does regard a lot more artists as similar than there are in the ground truth. Hence, the baseline model just needs to select fewer artists as similar, preferably the most similar artists. Therefore, the Top K thresholding approach was taken, as described in Section 4.1.5.

Table 4.3 showcases the results obtained when the thresholding method used was $TopK$. Here, we obtain the best set of precision, recall, and F1 scores for the feature set of $SB + SCon + SR$. We also use Pearson's correlation coefficient as a metric and even here, the previously mentioned feature set performs the best. Therefore, our final Evaluation Baseline uses the $SB + SCon + SR$ feature set and the $TopK$ method for thresholding. As evidenced by the scores, this thresholding method performs much better overall that previous one.

Although our evaluation framework also outputs accuracy, we consider this metric to be less important than others in our evaluation baseline because there is a large class imbalance in our ground truth here. The number of similar artists is quite low compared to the number of dissimilar ones in the ground truth. Therefore, even if the model outputs all artists as dissimilar, it will still obtain a high accuracy value.

Features	Clusters	TP	FP	TN	FN	Prec.	TP Rate	Accur.	F1 Score	Corr.
SCen + ZCR + RMSE	60	646	38513	115437	640	0.0163	0.5023	0.7545	0.0316	0.0519
SB + SCon + SR	70	647	36131	117819	639	0.0174	0.5031	0.7631	0.0337	0.0565
MFCC	65	629	34467	119583	657	0.0180	0.4891	0.7772	0.0347	0.0578

Table 4.2: Resultant Values of Metrics. Thresholding Method - Mean of the Distance.

Features	Clusters	TP	FP	TN	FN	Prec.	TP Rate	Accur.	F1 Score	Corr.
SCen + ZCR + RMSE	60	399	1017	152933	887	0.2812	0.3094	0.9933	0.2947	0.2888
SB + SCon + SR	70	404	995	152955	882	0.2873	0.3157	0.9934	0.3008	0.2950
MFCC	65	396	1007	152943	890	0.2819	0.3087	0.9934	0.2947	0.2888

Table 4.3: Resultant Values of Metrics. Thresholding Method - Top K.

Hence, other performance metrics, like precision, recall, and F1 score, become important.

Test of Statistical Significance for Selected Feature Set

As evident in Table 4.3, $SB + SCon + SR$ set of features outperforms other feature sets. As we cannot make any assumptions about the distribution of these set of metrics, we use a type of non-parametric statistical test called 'permutation test' [17] in order to prove the significance of our observed results. Our objective is to show that the nature of difference observed for each metric between the set of features remains so 95 percent of the times we re-sampled our observed output values.

Therefore, we shuffled the model's output for the three different input features 25000 times, and calculated the difference between three model outputs for each metric highlighted in Table 4.3. The idea is to show that margin by which second set of features performs better than first and third set is *not* a random occurrence, and the exact numbers presented below support this claim. Keeping 5 percent as our threshold, we found the second feature to be performing better than the other two in a statistically significant manner.

In Table 4.4, SZR refers to the numbers corresponding to the $SCen + ZCR + RMSE$ set of features, SSS refers to the $SB + SCon + SR$ numbers and $MFCC$ refers to the $MFCC$ numbers. For example, in the first column, $SSS - SZR$ value for Precision denotes the percentage of times the difference between SSS and SZR values of precision during the 25000 times we calculated it, was higher than $SSS - SZR$ from Table 4.3. In this particular example, it is 1.9%, which is less than our set threshold of 5%. Similar calculations were made for the second column. The last column is of most interest to us, as it signifies the number of times the difference between SSS and SZR, as well as SSS and MFCC values for the metric was greater than the values from Table 4.3. In this particular example, it is 0.9%, which is less than our set threshold of 5

Therefore, going by all the metrics covered in Table 4.3, SSS, which is $SB + SCon + SR$, perform better than other set of features in a statistically significant manner.

Metrics	SSS - SZR	SSS - MFCC	(SSS - SZR) and (SSS - MFCC)
Precision	0.01948	0.07692	0.00924
F1 Score	0.01952	0.05004	0.00732
TP Rate	0.01928	0.03008	0.00544
Corr	0.00218	0.00028	0.00004

Table 4.4: Statistical Significance Test Results. Thresholding Method - Top K.

Chapter 5

Conclusion

5.1 Summary

We began by introducing the problem of measuring artist similarity algorithmically by shedding some light on different aspects to be considered, prior work, and known challenges with estimating it. Thereafter we looked at how machine learning and deep learning are being used in the domain of audio related research. End-to-end deep learning models are getting more and more popular and perform better with more data at their disposal, something the FMA can provide with its full-length audio files. Finally, in Chapter 1 we looked at the related work that forms the bedrock of this thesis project: the FMA dataset and the Spotify Web API, through which we access Spotify's related artists set.

We established how it is essential to provide a dataset with a common ground truth and framework for different artist similarity models to test and compare performance, thereby promoting research on a topic which is quite subjective to begin with. We then outlined the details of the methodology used to build the dataset: data preprocessing, using the Spotify Web API to get the related artists, combining everything, and coming up with the most optimized set of artists in the dataset based on 1) number of similar artists in the set, 2) distribution of songs per artist and 3) number of genre covered in the dataset. Various experiments were conducted and the results obtained were presented and analyzed in Chapter 3. The optimization algorithm was presented and its performance was discussed alongside the nature of the optimization function itself. The formulation was discussed alongside the results obtained, broken down into its individual components.

It is important to provide an evaluation baseline and an evaluation framework with a dataset, and in Chapter 4, we discuss the baseline and framework associated with RASM. The various features used in the evaluation baseline were outlined and how they were used, followed by the unsupervised approach taken to calculate similarity between artists. Evaluation framework details were listed out including how the root mean squared error can be used for training, and other output metrics for comparing and evaluating performance against the ground truth. Finally, results of the evaluation baseline were presented as output by the evaluation framework, and the best method for the baseline was chosen.

5.2 Limitations

5.2.1 Not all artists in FMA could be included

As an example, the FMA large training dataset has about 16,000 distinct artists, out of which only 5,137 artists are present in the Spotify database. Therefore, many artists were discarded. There are various reasons for which these artists are not present in the Spotify database, including licensing issues on the streaming platform. However, it does reduce one of the main advantages of the dataset, which is that it is a massive dump of the FMA, a place where artists make their songs available freely or under certain licenses making it possible for researchers to use complete audio tracks.

Consequently, many various genres are under-represented as well. Although, 5,137 is still a big number for the count of artists in a dataset, there is a lost potential of using the entirety of FMA for artist similarity calculations and recommendation tasks.

5.2.2 Unable to obtain one true global optimum i.e., *Best N*

As discussed in Section 3.2, the nature of our optimization algorithm is that we get a range of best values of N instead of a clear winner. For the FMA small training dataset we get this range of 100 to 170 artists out of a possible 390. Although we did obtain a certain highest value at N equaling 120, but the goodness score for this value was only slightly better than others. Moreover, on multiple iterations of the optimization algorithm, we obtained different best values of N, albeit within the range of 100 and 170, thereby helping narrow it down to this range.

Although, one true value of *Best N* is desirable, obtaining a range of best values of N gives us the flexibility to choose the best value of N based on prioritizing of the individual components, like genre coverage, the distribution of songs, and the number of similar artists in set. For the purposes of this thesis project, we went ahead with the value corresponding to the highest goodness score obtained, which is 120 for the FMA small training dataset. We could have also just chosen the largest value of best N in the range, 170, in the interest of including more artists while still having a balanced dataset.

5.2.3 Results on the FMA Large Dataset

All the results, explanations, and visualizations that are presented in this thesis are based on the small and medium FMA datasets. This is due to the large computation time and memory requirements for the FMA Large dataset. There is no difference between the small, medium, and large datasets except for the number of artists, and hence, the number of songs and other data. The results obtained are consistent between the small and medium datasets, and therefore after running some preliminary runs of the optimization algorithm on the large datasets, we believe the consistency in results will scale to the large dataset as well.

5.3 Future Work

5.3.1 Deep Learning Baseline

As of now, the RASM dataset does not come with a deep learning baseline. However, we have been working on a deep learning feature embedding based approach for artist recommendation. In fact, the seed idea of RASM was motivated by earlier research on artist recommendation. Although we do have a very intuitive evaluation baseline available with code, we plan to publish a deep learning model as a baseline in the near future as well.

5.3.2 Computing on FMA Large Dataset, and at lower step size

We have already run preliminary tests on the FMA Large dataset, but due to limited computing resources and time limitations, the results are not verified and presented in this thesis. However, in the near future, we plan to publish all the results on the large FMA dataset. Please refer to Section 3.4 for more details on the runtime analysis of our algorithm and projected time for future computations.

5.3.3 Tutorials

Although all the code will be available online, is in the form of IPython notebooks and quite readable, we plan to provide certain tutorial notebooks. Most of the tutorials on using the FMA dataset are already available, we plan to add to those to use the related artists features.

In conclusion, we have attempted to build a dataset using the FMA Dataset and Spotify's Web API, bringing together key properties of both to promote research on measuring artist similarity algorithmically based on content. In a nutshell, this hard problem is equivalent to coming up with a dense graph from a sparse graph, but with a few added conditions like distribution of songs per artist and genre coverage within a set. We introduced a new mathematical formulation and an optimization algorithm based on those formulations to come up with a balanced subset of artists in the dataset. Furthermore, an evaluation baseline was implemented and its performance was tested using the evaluation framework that is published with the RASM dataset. Our hope is that RASM dataset serves as a great asset for the MIR community researching on content-based recommendation systems, providing a common ground on which people can build and compare models.

Chapter 6

References

[1] Daniel P. W. Ellis, Brian Whitman, Adam Berenzweig, and Steve Lawrence. The quest for ground truth in musical artist similarity. In *ISMIR*, 2002. vi, 2, 3, 4

[2] Brian Whitman and Steve Lawrence. Inferring descriptions and similarity for music from community metadata. In *In Proceedings of the 2002 International Computer Music Conference*, pages 591–598, 2002. 1

[3] Brian Mcfee and Gert Lanckriet. Heterogeneous embedding for subjective artist similarity. In *In Tenth International Symposium for Music Information Retrieval (ISMIR2009*, 2009. 2, 3

[4] Sergio Oramas, Mohamed Sordo, Luis Espinosa Anke, and Xavier Serra. A semantic-based approach for artist similarity. In *ISMIR*, pages 100–106, 2015. 2, 3

[5] Dan Ellis (LabRosa). Opennap (music collections) data - https://tinyurl.com/yd3lutzy. 3

[6] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron C. Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *CoRR*, abs/1612.07837, 2016. 5

[7] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016. 5

[8] Chris Donahue, Julian McAuley, and Miller Puckette. Synthesizing audio with generative adversarial networks. *CoRR*, abs/1802.04208, 2018. 5

[9] Kirell Benzi, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. FMA: A dataset for music analysis. *CoRR*, abs/1612.01840, 2016. 6

[10] E. Shakirova. Collaborative filtering for music recommender system. In *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 548–550, Feb 2017. 6

[11] Adapa B. Barkana B.D. Bachu R.G., Kopparthi S. Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal. pages 1–3, 10 2010. 26

[12] Unjung Nam (CCRMA). Spectral centroid - https://ccrma.stanford.edu/ unjung/air/areaexam.pdf. 27

[13] Andrea Trevino. Introduction to k-means clustering - https://www.datascience.com/blog/k-means-clustering. 27

[14] Scikit. Elbow method for selecting number of clusters - http://www.scikit-yb.org/en/latest/api/cluster/elbow.html. 27

[15] Saimadhu Polamuri. Five most popular similarity measures implementation in python - http://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/. 29

[16] Renuka Joshi. Accuracy, precision, recall and f1 score-interpretation of performance measures - http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/. 31

[17] Thomas J Leeper. Permutation tests - https://thomasleeper.com/rcourse/tutorials/permutationtests.html. 34

[18] Julian Urbano, Juan Llorens, Jorge Morato, and Sonia Sanchez-Cuadrado. Using the shape of music to compute the similarity between symbolic musical pieces. *7th International Symposium on Computer Music Modeling and Retrieval*, pages 385–396, 2010.

[19] Valerio Velardo, Mauro Vallati, and Steven Jan. Symbolic melodic similarity: State of the art and future challenges. *Computer Music Journal*, Volume 40, 2016.

[20] Yading Song, Simon Dixon, and Marcus Pearce. A survey of music recommendation systems and future perspectives. *International Symposium on Computer Music Modelling and Retrieval*, 9th:395–410, 2012.

[21] Michael D. Barone, Kurt Dacosta, Gabriel Vigliensoni, and Matthew H. Woolhouse. Grail: Database linking music metadata across artist, release, and track. pages 49–54, 10 2017.

[22] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, APPROX '00, pages 84–95, London, UK, UK, 2000. Springer-Verlag.

[23] Practical Cryptography. Mel frequency cepstral coefficient (mfcc) tutorial - http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

[24] Chris Emmery. Euclidean vs cosine distance - https://cmry.github.io/notes/euclidean-v-cosine.

[25] Keikichi Hirose Md. Khademul Islam Molla. On the effectiveness of mfccs and their statistical distribution properties in speaker identification. *VECIMS 2004 - E E E International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*, jul 2004.

[26] Antoni B. Chan Tom LH. Li. Genre classification and the invariance of mfcc features to key and tempo. *International Conference on MultiMedia Modeling*, pages 4–8, jan 2011.

[27] Daniel P.W. Ellis Michael I. Mandel. Song-level features and support vector machines for music classification. *ISMIR 2005: 6th International Conference on Music Information Retrieval: Proceedings: Variation 2: Queen Mary, University of London & Goldsmiths College, University of London, 11-15 September, 2005*, sep 2005.