Master thesis on Sound and Music Computing

Universitat Pompeu Fabra

Exploring music similarity with AcousticBrainz

Philip Tovstogan

Supervisor: Dmitry Bogdanov

Co-Supervisor: Alastair Porter

August 2018

Master thesis on Sound and Music Computing

Universitat Pompeu Fabra

Exploring music similarity with AcousticBrainz

Philip Tovstogan

Supervisor: Dmitry Bogdanov

Co-Supervisor: Alastair Porter

August 2018

Contents

Acknowledgement

I would like to express my sincere gratitude to my supervisors for guidance, my classmates for insights and communications, and my family and friends for always supporting me.

Abstract

Music similarity is used in many applications ranging from music recommendations to media retrieval systems. Most of music similarity metrics are computed as distances between songs on a certain multidimensional feature space. This feature space typically consists of audio descriptors that are extracted from raw audio (rhythm, tonality, spectral features). By using similarity distances we can perform a query in database, and it will return number of tracks with shortest similarity distances, thus the most similar ones.

Intrinsically similarity is a subjective concept, because different listeners perceive different things as similar. In the research of music similarity it had been defined in different ways and various similarity measures had been proposed. Evaluation of those metrics is a difficult task, and is usually done via objective metrics because it requires less resources.

Similarity can be measured either subjectively by using relative comparison by human subjects or objectively by using music metadata such as artist, album, genre and cultural context. Example of context usage is co-occurrence of the tracks on radio broadcasts or compilation CDs. However, subjective evaluation of most similarity metrics that are introduced in the research is usually performed with limited number of participants due to nature of experiments. In this thesis we address this problem and propose a system that can be used to perform subjective evaluation of different similarity metrics at large scale.

AcousticBrainz is an online platform that aggregates music descriptors extracted from audio with globally identifiable ID for each track. We use this platform for implementation of state-of-the art similarity metrics as well as crowd-sourcing evaluation capabilities. Using user feedback as the evaluation source we are able to compare performance of various algorithms at large scale. That is achieved by providing means of similarity computation between any of millions of songs in AcousticBrainz within acceptable time.

We also explore the possibility of allowing users to create their own hybrid similarity metrics and evaluate their performance in the system. The system allows to compare different metrics according to different similarity dimensions (timbre, rhythm) and calculate the performance based on subjective evaluation by users.

Our results confirm the relative performance of algorithms from the previous research based on subjective evaluation and the platform shows support for much larger scale experiments and potential of being used for various practical purposes.

Keywords: Audio; Music; Music Similarity; Music Information Processing; AcousticBrainz.

Chapter 1

Introduction

Music similarity is easily understood intuitively, however it is much more difficult to define it in the research context. While average music listener can say that track A is more similar to track B than track C, it is difficult to break down this decision to lower levels. There are multiple aspects of music that humans usually consider when performing similarity comparison: genre, instrumentation, melody, mood, vocal, tempo, rhythm, cultural context, lyrics (if present) and much more. And moreover, different people might perform this evaluation differently.

Historically, metadata has been used in research to objectively measure similarity (i.e. same album, artist, genre, country, year). Indeed, tracks from the same album usually are similar between each other in genre, instrumentation and overall feeling. However this method is not perfect because album composition inherently relies on diversity, and this type of similarity might be different from similarity that you would expect from different covers of the same song (melody, key, rhythm; however instrumentation and arrangement might be totally different).

Subjective similarity (relative) might be easy to measure, and it is typically similar between different people. However when talking about the fine details there might be differences in similarity perception between individuals. Reasons that pertain to this include some people giving higher weight to particular dimension of similarity, e.g. instrumentation, overall form and composition details, or using their gut feeling

or overall impression that is usually biased towards their personal experience.

A typical application of music similarity is the recommendation systems that are one of the selling points of different music streaming services. If the algorithm can provide good recommendations that the user enjoys, the engagement metrics will grow and the service will be naturally more successful. More often than not, collaborative filtering systems are used, which can be simply explained as: user is recommended tracks that other users with similar taste listen to (large number of common tracks listened), and which are missing from user's listening history. While this approach has proven successful, there are several problems with it:

- **Cold-start problem**, where if tracks don't have any listening data, they will not be recommended to anyone and thus have difficulties gaining more listening data. So this cycle is difficult to break unless the tracks explicitly will gain minimal amount of listening data to be recommended.
- **Isolation bubble**, where because users are recommended tracks based on listening history of users with similar taste, the algorithm is trying to make all users with similar taste to listen to similar music, exacerbating overall segregation and clustering in the userbase.

A typical answer to the cold-start problem is content-based recommendation systems that take advantage of the content instead of listening history and use whatever information that can be extracted from it to bootstrap the listening. Isolation bubble is more difficult to address, but a common approach is to add degree of randomness to the algorithm to occasionally recommend something totally different to increase diversity of recommendations.

Another application that emerged recently is automatic playlist generation (e.g. Spotify RecSys 2018 challenge[1]). While recommendation systems might be tuned for individual users, playlist generation is more general and context dependent (e.g. playlists for party or chill evenings). Although the nature of playlist generation is

[1] `https://recsys-challenge.spotify.com/`

similar to track recommendations, thus similar techniques are often used, there are slight differences that can be taken advantage of by algorithms to improve performance.

In this thesis we explore the audio content similarity between tracks with the focus on large-scale applications. Usually it is difficult to gather much data from subjective evaluation experiments due to human involvement and limited resources. We address this issue by proposing a platform that can be used for large-scale evaluation experiments.

This thesis is structured in the following way: the remainder of this chapter will introduce the motivation behind the research question and problem statement. In chapter 2 we will talk about state of the art and related research that had been already performed. In chapter 3 we talk about implementation options, what are pros and cons of each and how did we choose the implementation framework. Chapter 4 talks about the methodology, system design, experiment design and results. The implications of results, summary and future work are discussed in chapter 5.

1.1 Motivation

While there are number of commercial solutions and new research papers being published on the topic of music similarity, there is always a gap between industry and academia. Big companies can take advantage of large datasets of audio tracks that are available internally, while research datasets are difficult to open because of copyright issues. There are datasets available that can be used for music similarity, i.e. MagnaTagATune [1], around 2.5k excerpts of 30s with subjective similarity annotations. However it is difficult to compete with companies like Pandora or Spotify that have more than 30 million tracks available internally. Although similarity annotations are probably not available explicitly, they can be computed from listening history and other data.

AcousticBrainz [2] (AB) is a platform with the goal of gathering audio descriptor data in one place without storing audio itself, thus not being subject to limits of

copyright. Users that own music can use a tool that is based on the Essentia library [3] library that extracts the information from audio and submits it to AB. At the moment of writing this thesis AB contains approximately 10 million submissions for 3.6 million unique tracks. The information is extracted with collection of Essentia's music extractor algorithms that are state-of-the-art in Music Information Retrieval (MIR).

Although a number of attempts in recent research work with convolutional neural networks (CNNs) that use spectrograms as input [4], the downside is that you need actual audio or spectrogram excerpts of the tracks. While audio descriptors from AB contain less information overall, we can take advantage of the huge amount of data available on the platform.

1.2 Problem statement

The issue that we want to address, particularly in the area of music similarity, is that subjective evaluation is done on a relatively small scale due to experiment setup limitations and lack of resources. Thus using AB as the base for building and evaluating different similarity metrics provides the platform that can be used for large-scale evaluation. Moreover, with the number of different audio descriptors stored in AB, it is possible to build new metrics that can be used for similarity with potentially better performance.

Thus the problem that we address in this thesis is: how can we take advantage of large-scale data available in AB to perform an evaluation of state-of-the-art similarity algorithms and metrics? Moreover, can the evaluation platform based on AB help in developing new and better similarity algorithms and metrics?

Chapter 2

Background and related work

In research, music similarity is usually defined as distance between tracks on the multidimensional feature space. In this thesis we will focus on content-based features. This section will introduce several works that are relevant to this thesis as well as other state-of-the art approaches for computing music similarity.

2.1 Audio content descriptors

Essentia is a software library developed by Bogdanov et al. [3] that implements numerous different state-of-the-art algorithms that are used for music information retrieval (MIR), such as extraction of low-level audio features (temporal, timbre, tonal etc.) as well high-level descriptors (genre, mood, instruments etc.). Its MusicExtractor collection of algorithms encompassest the extraction of a multitude of MIR descriptors that are useful and descriptive for a music track.

The audio descriptors can be separated into several categories: timbral, rhythmic and tonal. Audio descriptors are computed on per-frame basis as well as their statistics over the whole track. While per-frame data is useful, it takes huge amount of space to be stored and time to be processed, thus is usually discarded in storage-sensitive applications. Moreover, by using a statistical summary of the frame-based features: mean (first moment), variance (second moment) as well as higher moments

it is much easier to compare different tracks using several numbers instead of using long vectors of frame-based features. This approach is preferred due to space and time efficiency, thus we mostly focus of statistical descriptors in this thesis.

AcousticBrainz [2] (AB) is a platform for aggregating audio descriptors extracted by Essentia's MusicExtractor in the online database. There are no actual audio files present in the database due to copyright issues. AB is part of the MetaBrainz foundation and is connected with the MusicBrainz [5] (MB) platform that provides the unique ID for every track and metadata including artist name, album, name, genre, etc. Anybody can make a submission if you have the audio data via provided software that performs extraction of descriptors and uploads them to AB. Same tracks can have multiple submissions that are independent and share only the reference track ID. Obviously, if the data between submissions is too distinct, that means that either the track identification might have failed, or the audio might be corrupted or modified.

2.2 Similarity metrics and algortihms

Music Information Retrieval Evaluation eXchange (MIREX) is an annual event that evaluates performance of submitted algorithms on multitude of MIR tasks. The Audio Music Similarity and Retrieval task[1] (ASM) has been run since 2006. As the name implies, the algorithms are expected to perform a query on the database of music tracks and find tracks that are considered most similar to the reference track. The evaluation is performed subjectively using 2 metrics: BROAD (3-point scale) and FINE (from 0 to 100). The latest instance was run in 2014 and the ASM task was not included in the subsequent iterations. We will take an overview of several of the high-performing algorithms that had been evaluated in the context of MIREX ASM as well as some meta studies.

There are several approaches that are typically used to tackle computation of music similarity. Most common one is representing tracks as points on the multidimensional space where distance metrics can be used to calculate similarity. Bogdanov

[1] http://www.music-ir.org/mirex/wiki/2016:Audio_Music_Similarity_and_Retrieval

et. al. [6] have done extensive research and introduced several baseline metrics as well as proposed some simple metrics and the combined hybrid metric. This paper is the basis for this research, as the audio descriptors for this research are the same ones that are stored in AB and most of the proposed metrics can be easily implemented in the context of AB. The brief overview of metrics introduced in the paper is presented in the following section (2.2.1). Moreover, we overview high performing algorithms that had been submitted to MIREX in section 2.2.2.

2.2.1 Hybrid content-based similarity

The first baseline metric is an euclidean distance based on principal component analysis (PCA) of number of timbral, temporal, and tonal descriptors (L_2-*PCA*). It follows work proposed by Cano et al. [7] and includes the manually selected subset of musical descriptors (201 in total). The pre-processing of the data includes normalization, and then principal component analysis (PCA) is performed to reduce the number of dimensions to 25 variables.

The second baseline metric is an euclidean distance based on relevant component analysis (RCA) of the L_2-*PCA* manually selected subset: L_2-*RCA-1*; as well as full set of descriptors: L_2-*RCA-2*. In both cases the number of output dimensions is chosen to be 25.

The third and last of the baseline metrics is based on the timbre modeling with Gaussian mixture models (GMM) of first 13 MFCC coefficients (*1G-MFCC*). In the paper authors used simplification of the timbre model using single Gaussian with full covariance matrix. The distance is measured as a closed form symmetric approximation of the Kullback-Leibler divergence:

$$\begin{aligned} d(X,Y) =& Tr(\Sigma_X^{-1}\Sigma_Y) + Tr(\Sigma_Y^{-1}\Sigma_X) + \\ & Tr((\Sigma_X^{-1} + \Sigma_Y^{-1})(\mu_X - \mu_Y)(\mu_X - \mu_Y)^T) - 2N_{MFCC} \end{aligned} \tag{2.1}$$

where μ is a vector of means, Σ is covariance matrix and N_{MFCC} is number of

dimensions.

The first proposed metric is a tempo-based distance (*TEMPO*) that compares number of beats per minute (BPM) and onset rate (OR) defined as number of onsets per second. It takes advantage of the assumption that BPM and OR values that are integer multiples of each other are more similar:

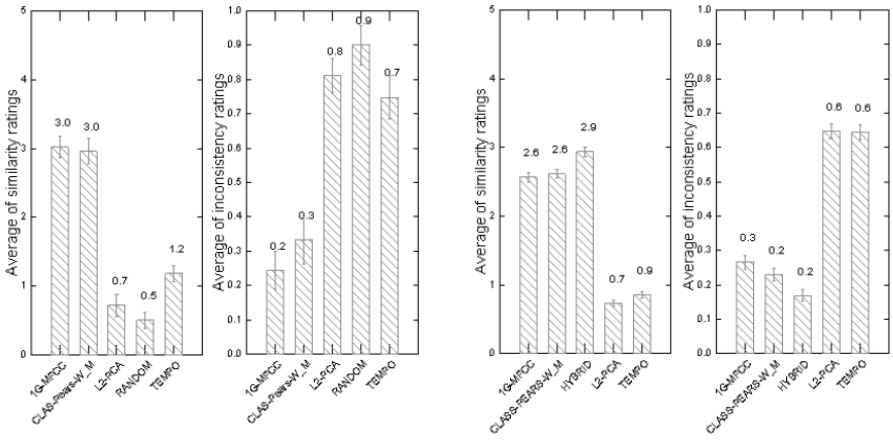
$$d_{metric}(X,Y) = \min_{i \in N}(\alpha_{metric}^{i-1} \left| \frac{\max(X_{metric}, Y_{metric})}{\min(X_{metric}, Y_{metric})} - i \right|) \tag{2.2}$$

where $metric \in \{OR, BPM\}$. The combined *TEMPO* metric is an equally weighted linear combination of OR and BPM.

The second approach is a classifier-based distance (*CLAS*) that is powered by 14 multi-class support vector machine (SVM) classifiers. They are split into 3 categories of "genre and culture" (GC, 4 classifiers), "moods and instruments" (MI, 8) and "rhythm and tempo" (RT, 2). Each classifier has its own number of output classes based on the dataset labels and training is performed on 14 ground-truth collections with input being zero mean unit variance normalized low-level audio descriptors. The output of a classifier is a probability vector of respective classes. Similarity is computed as a distance between each classifier's probability vectors. There are multiple distance metrics considered, but the best performing one is Pearson correlation distance manually weighted between 3 categories: 0.5 for GC, 0.3 for MI, 0.2 for RT and equal weighting inside the categories: *CLAS-Pears-*W_M.

The subjective evaluation of performance of the metrics is shown in Figure 1a. It was done on the 5-song playlists, where users rated the similarity rating on 0 to 5 scale and inconsistency on a binary scale.

The last approach proposed by the authors involves linearly combining previously introduced metrics into a hybrid one. Weights are manually assigned based on the performance of the algorithms (see Figure 1): 0.7 for L_2*-PCA*, 3.0 for *1G-MFCC*, 1.2 for *TEMPO* and 3.0 for *CLAS-Pears-*W_M. Final evaluation results of the *HYBRID* approach are included in Figure 1b.



(a) All metrics (b) *HYBRID* evaluation

Figure 1: Subjective evaluation of hybrid approach (taken from [6])

2.2.2 Other approaches (MIREX)

In this section we give an overview of several algorithms that have been submitted to MIREX ASM in the recent years.

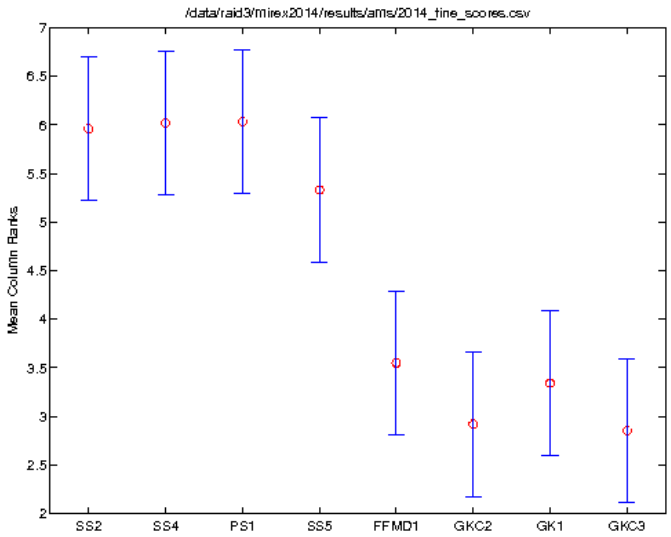


Figure 2: MIREX 2014 evaluation: FINE scores

Seyerlehner et al. [8] (SS on the figure 2) use auto-taggers as a intermediate step for similarity computation. Auto-taggers, as the name implies, automatically provide number of tags for the tracks. They are purely content-based and provide a transformation from audio descriptors to semantic space that is used later for computing

the similarity. Training is done on publicly available datasets: MagnaTagATune [1] and RadioTagged with tags retrieved from Last.fm.

Two sets of tag classifiers are trained: *low* and *high* quality. Both are based on *block-level* audio features [9] where the high quality includes two additional block-level features. Low quality auto-tagger uses PCM to reduce dimensionality and SVM for classifiers, while high quality one uses random forest (RF) directly on high-dimensional input data. Similarity is computed as a Manhattan distance between affinity vectors that contain tag classes.

Pohle et al. [10] (PS on figure 2) use the Fluctuation Patterns [11] (FP) audio descriptor to compute similarity. FPs measure periodicities of the loudness in various frequency bands, considering a number of psychoacoustic findings. Several extensions of FPs: Onset Patterns (OPs) and Onset Coefficients (OCs) are introduced as better metrics for rhythmic similarity. Moreover, the timbre information is added to FPs for it to be used for general music similarity.

Both of these approaches are very successful in the multiple MIREX ASM task submissions (see figure 2). Because block-level features as FPs are the definitive strengths of the algorithms they cannot be efficiently implemented in the current context of AcousticBrainz. However, the concepts introduced by the authors might be useful in designing new metrics that use AB statistical features.

Another approach proposed by Gkiokas et al. [12] (GK, GKC on figure 2) uses Deep Belief Networks (DBF) directly on audio descriptors as input. Their method consists of three main parts, with the first step being feature extraction, which involves the calculation of three feature sets that correspond to music timbre, rhythm and harmony. Next, for each feature set a DBF is trained without supervision on a large music collection. The respective distances of the output units of the Deep Belief Networks between two music excerpts are computed, normalized and finally combined to form the distance measure. The proposed method is also evaluated in 2014 MIREX ASM task, however its performance is among the lowest ones (see figure 2).

Sequential complexity, introduced by Foster et al. [13] (FFMD on figure 2) takes advantage of sequential frame data for a particular descriptor and tries to quantify its complexity with feature complexity descriptors (FCD). The system models audio as a track-wise summary of FCDs computed on frame-based features. Across considered audio features authors compute pairwise distance measures between FCDs and to predict musical similarity, pairwise distances are linearly combined and then normalized.

2.3 Issues with subjective evaluation of similarity

Subjective evaluation of music similarity is prevalent in MIREX ASM task, but the performance and accuracy of the evaluation process is also important to consider. Flexer et al. [14] investigates this question and shows that there are inherent problems with this evaluation method.

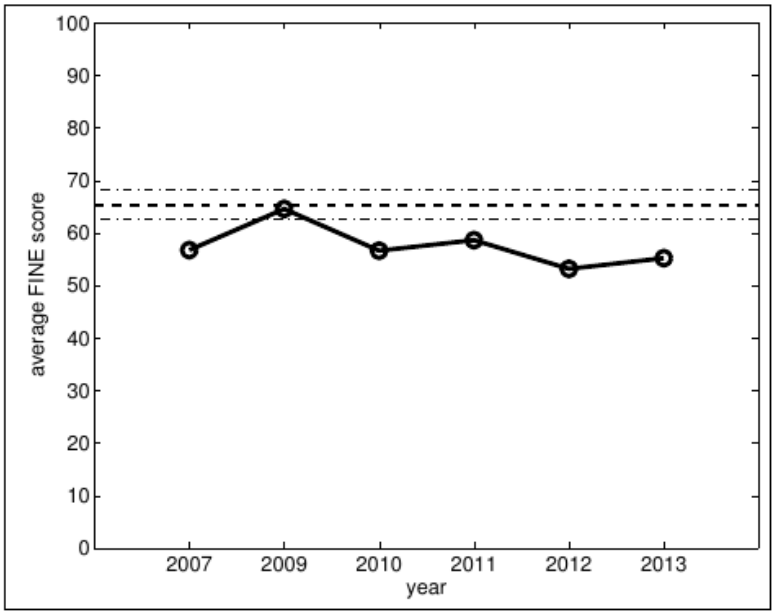


Figure 3: Best MIREX performances vs upper bound

The authors argue that there is a low inter-rater agreement due to the coarse concept of music similarity. One of the most important reasons is the vague definition of similarity for the MIREX graders that participate in the evaluation of task submissions. Thus each grader uses their own personal variation of music similarity

concept to evaluate submissions. As a direct result of this there exists an upper bound of the performance of the algorithms that compute music similarity. The upper bound is derived from data on inter-rater agreements and is shown in Figure 3 along with top performances. From the figure it is evident that this upper bound has already been achieved in 2009 and not been surpassed since.

One possible solution is to have more than one grader rate same query or candidate pairs of songs thus providing data to calculate inter-rater agreement and an upper bound that can be used for normalization. Moreover, each rater has different variance for scores, which also should be normalized on per-rater basis.

Several ideas from similarity evaluation of textural sounds can be applied to music similarity, specifically introduction of dimensions. For example, for textural sounds such qualities as “high-low”, “smooth-coarse”, “tonal-noisy” have been proven to be useful to discern different sounds. A similar approach can be used for music, where the inter-rater agreement can be increased by increasing dimensionality.

Last, but not least, is the importance of the context [15]. For example, similarity that is used for building a playlist might be different from similarity used to recommendation systems. Inclusion of the context will make the goal much clearer thus improving the consistency of subjective evaluation.

Chapter 3

Implementation frameworks overview

There are number of frameworks and libraries that provide the functionality that can be used for implementation of similarity subsystem for AB. In this chapter we provide an overview and go through pros and cons of each one.

3.1 Gaia

Gaia[1] is a general library to deal with multidimensional feature vectors built from audio descriptors. As it was developed directly with Essentia integration in mind, it is a first natural choice to be used for similarity computation. It is used for this purpose in Freesound [16] where there are approximately 380 000 sounds as of the moment of writing.

However one of the shortcomings of Gaia is that all of the audio descriptor data needs to be stored in memory, so while it is very powerful and easy to use, it falls short when dealing with large data. The number of tracks in AB is at least one order of magnitude larger than Freesound, and apart from low-level descriptors that are the direct output of MusicExtractor, there are also high-level descriptors available (SVM classifier outputs) that are useful for the computation of high-level similarity.

While not all of the descriptors are used in the metrics that are to be implemented

[1] `http://essentia.upf.edu/documentation/gaia/`

for similarity computation, only a small subset of them, eventually we want all the descriptors to be available for construction of new metrics. So while Gaia is good choice for implementation at the start, it might have some scalability issues further down the development pipeline.

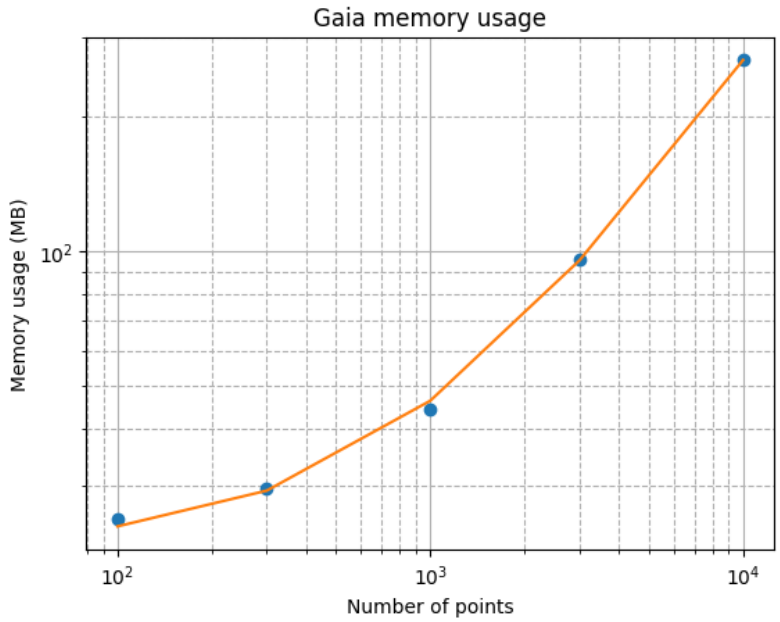


Figure 4: Gaia memory usage ($N = 2514$)

Figure 4 shows the memory usage of Gaia for the worst case where all descriptors available in AB (2514) are loaded per each data point. While obviously a worst-case scenario, it still shows the extent of scalability issues with Gaia. So for the custom metric functionality where we want to let users build new metrics we need to have all potential data available. The projected memory usage for dataset with size of 8 million points is more than 200 GB, and this is without distance metrics, just pure data.

Another downside of Gaia is that the project has stopped being developed and is currently outdated and no longer being improved. It requires number of updates both security-wise and feature-wise that will take substantial development time that is out of scope for this thesis. Thus in summary, there might be increasingly big technical debt that would be difficult to deal with several years into the future, unless the development and maintenance of Gaia will be addressed soon.

So, to summarize, we can see while Gaia can be very effective, there are number of possible issues with it, both in context of using it for such a large scale application and being outdated and not actively developed.

3.2 PostgreSQL Cube extension

The database engine that AB uses is PostgreSQL, so naturally some of engine's capabilities can potentially be used to implement similarity system. As of PostgreSQL version 9.6 (released in September 2016), its *Cube* extension that was built to store multidimensional cubes gained an functionality to perform k-nearest neighbours (KNN) search on its multidimensional space. It implements GiST index [17] thus achieving very good performance. This functionality is useful for performing similarity queries, while different metrics can be represented as different GiST indexes on the same table, and after initial testing proved to show adequate performance in our context.

Table 1 shows the results of initial evaluation of query times with 12-dimensional metric on the server with all AB data (NFS) and local machine (HDD). NFS server had more computing power, however storage access was slow due to data being hosted on a different virtual machine. HDD had better disk access and speeds, however it had only subset of the data due to storage limitations.

#rows	1k	10k	100k	1m	8m
NFS cold start	1.7	1.9	12.1	209	-
NFS repeated	0.02	0.03	0.16	1.1	2.9
HDD cold start	0.05	0.8	1.2	40.7	-
HDD repeated	0.05	0.07	0.1	0.3	-

Table 1: PostgreSQL Cube performance (s)

Because Cube is a PostgreSQL extension and all data in AB is already stored in the database, further data scaling is not a problem. If necessary, even advanced techniques with distributed database systems can be used to optimize the performance.

Limitations include a soft limit on 100 dimensions on the vector space, however that can be overridden if necessary. The distance functions that can be currently used

with GiST index are limited to Euclidean (L2), Manhattan (L1) and Chebyshev (L-inf). While there is not much work needed to introduce new metrics on already existing data, the index creation takes some time. Introduction of new metrics that operate on processed data might be costly due to computation of new vectors as well as index.

3.3 Apache Lucene: Solr and ElasticSearch

Apache Lucene[2] is a open-source project that develops Lucene Core subproject, that provides Java-based indexing and search technology. Solr[3] is a high-performance search server built upon Lucene Core and is designed to work with large-scale data. ElasticSearch (ES)[4] is another open-source product that is build upon the Lucene Core. The primary design principle of both Solr and ES is to provide effortless full-text search as well as data analytics with complex queries.

There are multiple differences between ES and Solr, but for our purposes important distinction is that ES typically works with JSON data, while Solr is more XML-based. One of the prominent features is TF/IDF-based full-text search, but it is not really usable for our purposes. Another built-in feature of distance measurement is tailored towards geographical applications. So for similarity computation between vectors we need to use custom distance-based implementation of ranking algorithm. As it had to be implemented using ES scripts, it was sub-optimal.

Another issue is the data storage. The only practical solution that allows the ES be used on top of PostgreSQL is an open-source project ZomboDB[5] which at the time of performing framework evaluation was only supporting PostgreSQL 9.5 and ElasticSearch 5.6 which were outdated versions. Due to several practical difficulties in utilizing ZomboDB for evaluation purposes, it was dropped out from consideration.

Thus if we would use ES directly, it would need the data to be stored in its own

[2] http://lucene.apache.org/
[3] http://lucene.apache.org/solr/
[4] https://www.elastic.co/products/elasticsearch
[5] https://github.com/zombodb/zombodb

indexes, so we would need to essentially have copy of all descriptors in the ES index in addition to ones already stored in database, which is a huge cost in space. Moreover, we would need to take care of synchronization and importing additional data from the database when it is available. All of this overhead is not feasible given that it proved to be difficult to utilize advantages of ES/Solr for feature vector similarity queries.

Thus ES/Solr have proven to be not the best solution for our problem given that PosgreSQL cube extension requires much less overhead for comparable performance, if not better one.

3.4 Google BigQuery

BigQuery[6] is designed to deal with big data and has its own indexing capabilities, but it is a commercial product and all the processing is done on the Google's cloud. This is the biggest issue of BigQuery - it is commercial. If we would to use this solution, to deploy instance of AB developers would need register a Google Cloud account and set up billing for BigQuery, thus having problems deploying AB as an open-source project. While it is possible to separate similarity functionality and make it optional for the AB project, it is generally not appropriate to require commercial products for even some functionality of an open-source project.

Having said that, BigQuery is still powerful tool to be used for search and ranking in such large-scale data as AB, and it is a good secondary optional candidate framework to be used for similarity queries.

3.5 Results

After careful evaluation and initial experiments, we chose PostgreSQL cube extension as the most appropriate framework for the implementation of similarity search in AB. Although BigQuery doesn't conform to open-source nature of AB, it is good secondary option, and while it is not used in the remainder of this thesis, it might

[6] `https://cloud.google.com/bigquery/`

be very useful for running other experiments on AB data.

Chapter 4

Design and implementation of similarity search

Given the advantages and limitations of the selected framework (PostgreSQL cube extension), in this chapter we present our overall approach to similarity implementation based on previous research. The main idea is to define and utilize *transformations* that take audio descriptors and transform them into vector space where simple distances work as well as complex distances on the original vector spaces. We also define *metrics* as the transformed feature vectors that define a new vector space where we can perform a similarity search using k-nearest neighbors algorithm.

In this thesis we work with statistical audio descriptors that are computed over all frames, thus discarding fine-grained details of temporal evolution of the descriptors over the track. The primary reason for that is that we don't have frame data available in AB. Although statistical descriptors are very useful and sufficient for our needs, as we mentioned in chapter 2, there are multiple algorithms that can take advantage of frame data that we are not discussing in this thesis as they are out of scope due to limitations of data available from AB.

4.1 Concept of transformations

As there are only several distance functions that are available for KNN search due to selected framework (Euclidean, Manhattan and Chebyshev are the distances available in PostgreSQL 10), we propose the idea of transformations that are used to transform the audio descriptors that are used in complex distance computations to vector space where the distance computation is simple. This will move the complexity from the distance computation to vector transformation that needs to be performed only once when the metric is created, thus simplifying and increasing performance of the similarity metric. It effectively takes advantage of GiST indexing and its fast KNN performance.

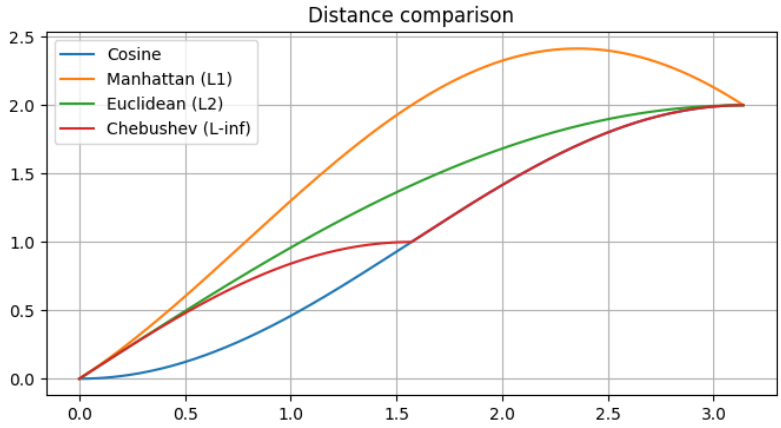


Figure 5: Distance comparison

Typical distances that are used in music similarity research are cosine distances. As it is not an available option for implementation, we aim to normalize audio descriptors to approach close to cosine distance (see Figure 5). The figure shows all available distance metrics in PostgreSQL 10 and how they behave for the unit vector rotating along the unit circle with its angle going from 0 to π. Obviously, all distances monotonically increase until π except Manhattan distance. Euclidean distance has advantage over Chebyshev by having much smoother and monotonic difference from cosine distance, and not having a sharp corner at $\pi/2$. Moreover, Euclidean distances are prevalent in MIR research along with cosine distances.

Thus important step is normalizing descriptors before them being used in similar-

ity calculation. Different normalization techniques should be applied depending on the nature of the descriptor, but in the end we want to have vectors with similar distributions. Thus we perform a feature standardization with zero mean and unit variance. For example in the MFCC case, it would make sense if each coefficient would be zero mean unit variance normalized individually, and then concatenated into a transformed MFCC vector. Thus the resulting MFCC vector would be approximately distributed around unit hypersphere.

Another limitation is related to queries based on several distances being linearly combined. Because we cannot aggregate distance metrics together to be used in queries due to PostgreSQL cube extension limitations, one of the possible ways to incorporate hybrid metric queries is to concatenate corresponding vectors. A downside to this method is increased dimensionality and thus increased complexity of queries. Although the concatenation approach is not ideal, it is the best solution that works with our limitations. Based on equation 4.1 we see that we are essentially replacing Manhattan distance for hybrid metric with Euclidean. While it has impact on the ordering of results being returned, on large scale it has minimal impact.

$$\begin{aligned} d_{HYBRID}(X,Y) &= w_1 d(x_1,y_1) + w_2 d(x_2,y_2) \\ d_{CONCAT}(X,Y) &= d((w_1x_1, w_2x_2), (w_1y_1, w_2y_2)) = \\ &= \sqrt{w_1^2 d(x_1,y_1)^2 + w_2^2 d(x_2,y_2)^2} \end{aligned} \tag{4.1}$$

To perform concatenation properly having equal importance in case of no weights, individual vectors should be normalized around the unit hypersphere as discussed above. Multiplying vectors that are being concatenated by weights works as expected with smaller weight corresponding to smaller importance.

Now we will introduce several transformations that are inspired by previous research. While complex metrics that have been introduced cannot be directly mapped to transformations, we take a trade-off between complexity of transformation and re-

sulting precision.

4.1.1 Circular transformation

In their previous work Bogdanov et al. [6] introduced *TEMPO* metric that takes advantage of the fact that for BPM and onset rate (OR) values that are integer multiples of each other are perceived as very similar. Moreover because of large weights being selected for integers that are powers of two, we assume that this relationship is exponential. Indeed, because of overwhelming presence on square hierarchical beat patterns humans are more likely to consider tempi similar if they differ by factor that is power of two than other integers.

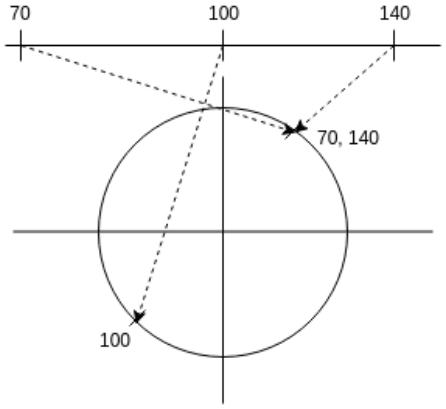


Figure 6: Circular transform: BPM

If we imagine a line with logarithmic scale with tempo values represented on it, we would try to put values that are equally spaced close to each other. We can do that by wrapping our tempo line onto the 2D circle with those points laying on top of each other (see figure 6), or effectively being separated by 2π after scaling the circle to be unit circle. Now we effectively have defined the transformation that takes BPM or onset rate and transforms the value into 2D vector.

$$T(x) = \begin{bmatrix} \cos(2\pi \log_2 x) \\ \sin(2\pi \log_2 x) \end{bmatrix} \tag{4.2}$$

So instead of calculating distance metric according to equation 2.2, we are calculating transformed values according to 4.2 and then calculating Euclidean distance between

them.

Some adjustments can be applied to this strategy if needed. If the multiples are considered to be not completely the same, but some distance apart, the transformation can have spring-like output wrapped around cylinder or sphere in 3D with subsequent strands of the spring separated apart by constant distance. However depending on the typical transformed values (e.g BPM 70-140), it should be scaled appropriately to match the unit sphere with average and deviation.

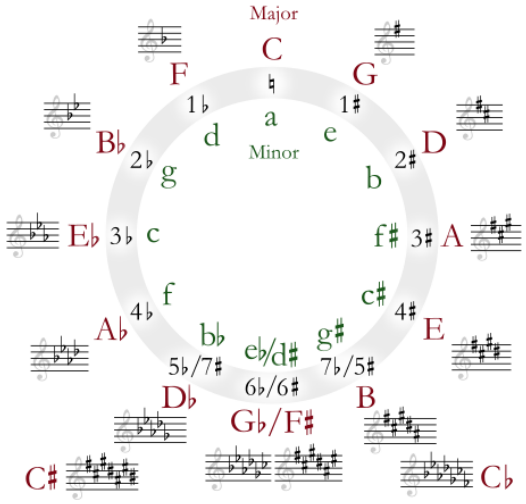


Figure 7: Circle of fifths
Just plain Bill / Wikimedia Commons / CC-BY-SA-3.0

While both BPM and OR can be used as individual metrics, we approximate the original *TEMPO* metric with concatenation of transformed BPM and OR.

A circular transformation can also be applied to tonal similarity, specifically key/scale taking advantage of musicological knowledge of how similar different keys are based on circle of fifths (figure 7). The simplest approach would be to map combinations of keys and scales to 2D vectors directly based on circle of fifths. A more advanced approach would be to also incorporate the pitch proximity thus increasing number of dimensions to properly represent both relations.

4.1.2 Probability vectors

One of more successful metrics *CLASS* incorporates outputs of high-level SVM classifiers. The same classifiers that had been used in the thesis are implemented in AB and stored in database as probability vectors. So to approximate *CLASS* we want to use those probability vectors. This data is considered high-level as opposed to raw extracted data that is named low-level.

As we discussed before, we need to normalize the vectors before using them for similarity computation. While it makes sense to use them as-is for simple similarity based only on high-level data, thus reducing amount of computations for preprocessing the data; to use them in hybrid metrics with low-level data we cannot avoid normalization.

If left as-is they will take non-negative values, thus being clustered in tiny $1/2^N$ part of hypersphere space, where N is the number of dimensions. Pearson correlation distance has been the best performing distance used successfully on probability vectors [6] for similarity evaluation. Thus we introduce "pearsonization" transformation (see equation 4.3) that after being applied to individual vectors reduces the Pearson correlation computation to Cosine distance, as well as taking care of normalization at the same time.

$$T_{PEARS}(x) = \frac{x - \bar{x}}{|x - \bar{x}|} \tag{4.3}$$

While resulting vectors are not actually zero mean unit variance normalized, it is due to the nature of probability vectors, where the number of degrees of freedom (DOF) is one less then number of its dimensions. Pearsonization brings both averare down and increases the variance, but the values will be spread on hyper-plane that crosses the hypersphere.

$$T_{BIN}(x) = 2x_1 - 1 \tag{4.4}$$

One limitation of pearsonization is that it is unusable for binary classifiers, because it pushes vectors to one of two extremes: (0,1) or (1,0). However as there is effectively only one DOF, we can extract it, normalize it (see equation 4.4) and concatenate with others to form a feature vector. As the "moods and instruments" category primarily consists of binary classifiers, this is the transformation that is used.

4.2 Similarity metrics

As work by Bogdanov et al. [6] is the base for this thesis, the initial implemented metrics are inspired by it and listed in Table 2. Concatenation of more than 2 metrics is indicated by | symbol.

Category	Descriptor	Dimensions
Rhythm	BPM	2
Rhythm	BPM | OnsetRate (OR)	4
Rhythm	BPM | OR | Key/Scale	6
Timbre	MFCCs	12
Timbre	MFCCs (weighted)	12
Timbre	GFCCs	12
Timbre	GFCCs (weighted)	12
High-level	Genre (Dortmund)	9
High-level	Genre (Rosamerica)	8
High-level	Moods	5
High-level	Instruments	3

Table 2: Base metrics

TEMPO metric is approximated by concatenation of BPM and onset rate (OR). Individual BPM metric is also considered to investigate effect of presence of OR. Moreover, while it is technically going outside of Rhythm category, we want to evaluate concatenation of BPM with OR and Key/Scale. The expected results of this similarity query are songs that might go well together in DJ mix, being similar both in tempo and compatible in key.

Timbre metrics are inspired by usage of MFCC in *1G-MFCC* metric. The 0th coeficient is discarded, because it represents average energy, which is not useful for timbre similarity. Individual coefficients are globally zero mean unit variance normalized according to previously discussed strategy.

We also consider GFCC coefficients that are based on ERBBands [18] instead of Mel-frequencies because they are known to exhibit better performance on the classification tasks of non-speech audio signals than MFCCs [19]. Moreover, the weighted versions of both of these metrics are considered, where coefficients with higher index are scaled down to reduce their importance (see equation 4.5). The reasons behind weighting is that typically higher-order MFCCs contain less useful information and are more noisy.

$$C_{i,MFCC_W} = \alpha^{i-1} C_{i,MFCC}, \text{ where } \alpha = 0.95, i \in [1, 12] \tag{4.5}$$

Genre metrics are composed of single classifier respectfully because of higher number of dimensions. *Moods* metric includes binary classifiers *happy*, *sad*, *aggressive*, *relaxed*, *party*. *Instruments* includes *acoustic*, *electronic*, *voice-instrumental*.

4.3 Implementation

As we decided to use PostgreSQL Cube extension, for evaluation purposes we recreated an instance of AB on a separate server with subset of 4.5 million tracks (approximately half of the original data) due to server space constraints. According to the metrics and transformations that have been described above, the transformed vectors were computed from raw JSON data and stored in separate table.

While low-level metrics can be defined purely as indices with custom functions on the table that contains all of low-level data, high-level data is stored separately, thus complicating things. Having all data in one table simplifies index definitions and while it is not the most space-efficient method, it is a good trade-off between space and complexity.

Another advantage of this approach is that we can perform transformations at the same time as we are moving the data. In the case of building indexes upon the unprocessed data the transformations should be defined as functions in the database engine using languages such as PL/pgSQL that are limited and clunky for imple-

mentation or PL/Python. Instead, we can use whatever language to transform data as it is being moved (Python in our case) and have freedom in transformation implementation.

Metric	Data size (GB)	Index size (GB)
BPM	0.16	0.7
Onset Rate	0.16	0.7
Key	0.16	-
BPM | OnsetRate	-	0.5
BPM | Onset Rate | Key	-	0.7
MFCCs	0.50	6.2
MFCCs (weighted)	0.50	3.8
GFCCs	0.50	3.1
GFCCs (weighted)	0.50	2.5
Genre (Dortmund)	0.40	1.7
Genre (Rosamerica)	0.36	0.8
Moods	0.26	2.9
Instruments	0.19	1.6

Table 3: Data and index size

Table 3 shows the size of the data and indexes. As it is evident, size of indices is larger then size of the actual data. All of the indexes are of GiST type (generalized search tree) [17] that allow for quick KNN lookup. Essentially the trees subdivide the vector space for quick search, thus the size of the index is different for the vector spaces with same number of dimensions, because it depends on the distribution of feature vectors.

4.4 Evaluation

To evaluate metrics that we have introduced in the previous chapter we are using the metric commonly used in MIREX: BROAD. It is a 3-point scale with values being: not similar (0), somewhat similar (1), and very similar (2). As we implement it as part of AcousticBrainz platform, it can be used for controlled experiments as well as for slow crowd-sourcing of data and feedback.

In the context of this thesis we have picked 6 tracks (see table 4) that represent different types of music and asked the users to evaluate how similar are the query

Artist Track	MBID
Queen - Bohemian Rhapsody	ebf79ba5-085e-48d2-9eb8-2d992fbf0f6d
Amorphis - Drifting Memories	8d5f76cf-0fa1-45a1-8464-68053d03b46b
Blind Willie McTell - Low Rider's Blues	919a494b-dccc-4801-8aff-779ba574afae
Skrillex - Scary Monsters and Nice Sprites	47974dfd-f37d-4f41-b952-18a86af009d2
The Beatles - Let It Be	0cdc9b5b-b16b-4ff1-9f16-5b4ba76f1c17
Howard Shore - The Three Hunters	b7ffa922-7bb8-4703-aa51-3bcc6d9cc364

Table 4: Evaluation tracks

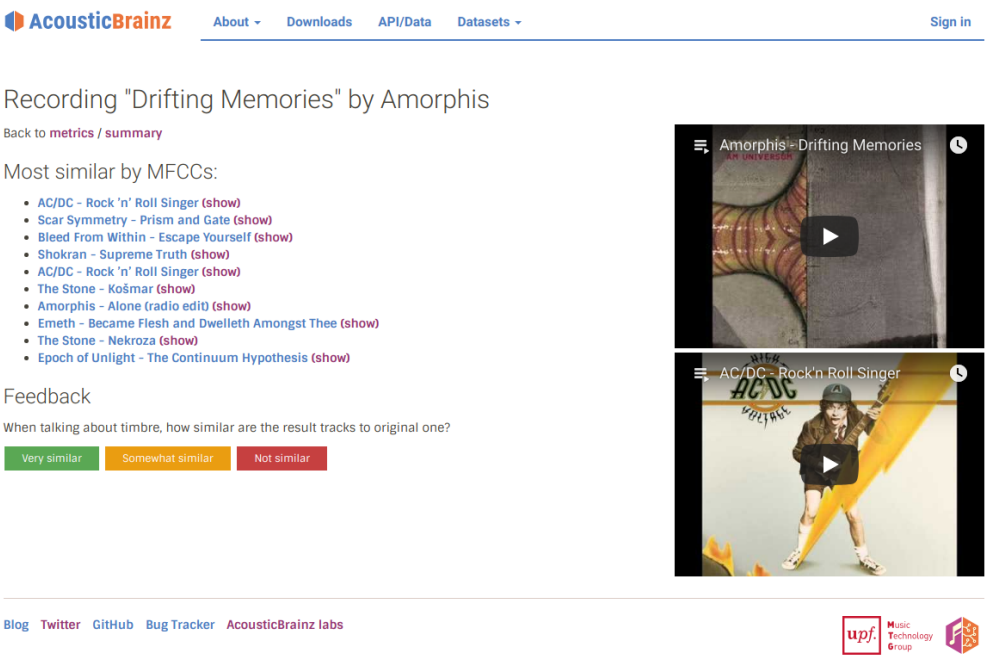


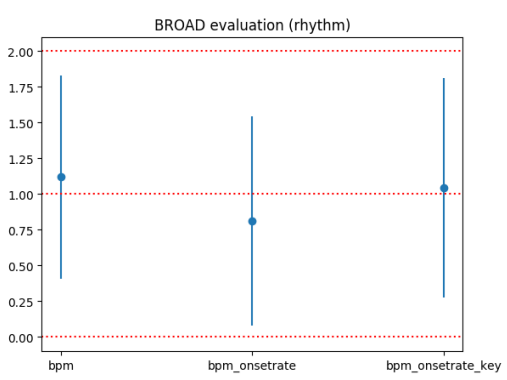
Figure 8: Evaluation interface

results made according to each of introduced metric (11 in total). The interface is shown in figure 8. We have presented links to similarity search results for each track with each metric (66 in total) to number of subjects for evaluation.

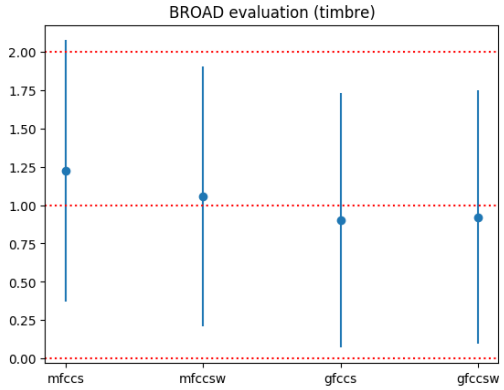
4.5 Results

As we mentioned before, the evaluation method was inspired by MIREX's BROAD metric. Number of participants varies from 5 to 10 depending on the track and the metric, but in total we gathered 218 evaluations for 11 implemented metrics over the 6 different tracks. Although subjects have been encouraged to evaluate similarity results of other tracks that they are familiar with, we haven't received

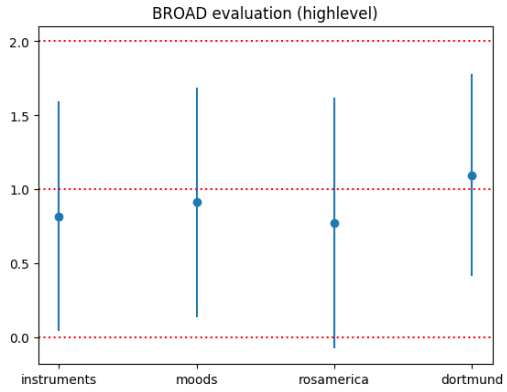
any evaluations for other tracks. In figure 9 you can see the average rating with the standard deviation for each metric, as well as the red dotted lines that indicate the values of BROAD scale.



(a) Rhythm



(b) Timbre



(c) High-level

Figure 9: BROAD evaluation of proposed metrics

Given the low amount of participation in this experiment, the performance of each metric is not significantly different (T-test with $p = 0.1$) from other ones. However we can still compare them based on the average rating.

It is very interesting seeing the BPM metric having the best performance in the rhythm category with BPM together with OR being the worst. That proves that people care more about tempo of the major beats as opposed to other onsets that might be happening in the track.

From the timbral perspective, although weighting was supposed to improve the similarity by putting more weight on lower-indexed coefficients, from the results of

our experiment it had degraded the performance slightly. Moreover, looking at all metrics across categories, MFCC metric had achieved the best performance overall with average BROAD score being 1.25.

For the high-level metrics, comparing two genre-based metrics, Rosamerica and Dortmund models, we see that performance of Dortmund in much higher than that of Rosamerica, which signifies the better quality of the model. Moods metric performed better than Rosamerica, and instruments' average is similar to Rosamerica, alhough as we mentioned before, there is no statistically significant difference between those.

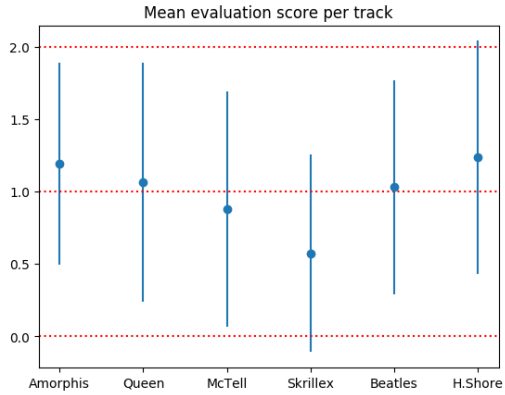


Figure 10: Track bias of BROAD evaluation

Another interesting and important thing to look at is the rating bias towards particular tracks. The average similarity rating for each of the 6 tracks is shown in figure 10. And from the figure we can see that the difference between average ratings of tracks is not smaller than differences between metrics in the same category, even across categories. So the performance of the metrics is significantly influenced by the query track.

That definitely has to do with amount of potentially similar tracks in the database. Particularly, the Skrillex track has the lowest average similarity, which might be attributed to lack of dubstep presence on the AB in comparison with other genres like rock or pop. The bias towards higher evaluation rankings is present in the metal track by Amorphis and the cinematic soundtrack from Howard Shore, which can be attributed to higher availability of potentially similar tracks in the database.

Chapter 5

Discussion and future work

One of the most important results of this thesis is the introduction and implementation of the platform that can be used for building different similarity metrics as well as large-scale evaluation. Although similarity metrics that are inspired by the latest state-of-the art algorithms have been implemented, the evaluation hasn't shown any significant results due to lack of user participation. Moreover, the evaluation data of proposed metrics that has been gathered is a good proof of the platform usability and scalability.

5.1 System performance

The performance of the system is evaluated based on the query time. Because of the way the database system handles indices, there are two states that have vastly different performance: cold - when the index hasn't been used in a while the database unloads it from memory; and hot - when the index has been used recently, thus it is partially loaded into the memory. In the cold state queries take additional time for the relevant part of index to be loaded into memory, that is directly influenced by reading speed of storage medium being used (HDD vs SSD). The amount of available memory is also important as the indexes that are used will push out indexes that are used less, however the OS caching mechanism speeds up this process comparing to direct reading from the disk, which proves that keeping the database in hot state

improves performance more than just memory vs storage read speed.

When the amount of requests sent to the server is not enough, the database will spend most of the time in cold state, thus having much longer queue response time. However, if the server is much more active with requests being much more regular, performance will improve significantly with query time being much faster.

Because of query times being too long for the cold state of the database (more than several minutes for bigger feature vectors such as MFCCs), we artificially added more requests to simulate the database being in hot state for performance evaluation being closer to real-world scenario.

In the hot state, the query time for most descriptors is less than 1 second, with an exception of timbral (MFCC, GFCC) and genre (Rosamerica and Dortmund) descriptors that sometimes take more, but still less than 10 sec. In the cold state the loading of heavy indexes, such as MFCC or GFCC usually takes around 5 minutes with occasional outliers. Indexes with smaller dimensionality take comparatively short time to be loaded from cold state - generally less than 20 seconds, however this performance is still not acceptable for a live system. Thus as we mentioned before - it is important to keep the database in the hot state.

Query times are noticeably larger for vector spaces with large number of dimensions, so our proposal of building hybrid metrics based on concatenation doesn't work very well, thus other options need to be explored, e.g. rank fusion or cascading. Rank fusion would still take advantage of efficient KNN look-up, but would require large number of returned tracks to effectively take advantage of intersection between the sets. Cascading suffers from inner queries having to run sequential search instead of taking advantage of indexes, thus its usefulness depends on its comparative performance to high-dimensional hybrid queries.

5.2 Summary

To summarize, we can definitely say that the proposed system achieves acceptable performance, which can be further improved by using faster storage drives as well as

overall power of the server. The evaluations for the introduced metrics are promising and prove the usability of the system, however the significance of our evaluation results is not big enough due to lack of user participation.

5.3 Future work

The current usability of AcousticBrainz platform is limited for a casual user, as it lacks functionality to get to track that you have in mind quickly. Currently the only way to do this is to use MusicBrainz to find the MBID of the sought track, and use it to access AB page of that recording. Thus we had no evaluations of tracks, other that have been provided (6), because users had troubles accessing them on AB. To address this issue, AB needs UI and UX improvements, primarily for browsing experience of different recordings based on artist, album and track name.

In current state, similarity results can be only accessed and evaluated on the AB page of that track. Similarity functionality that is implemented in open-source project such as AB might be of great use to multiple third-party apps and websites. A natural extension of implemented functionality is to make it available via the AcousticBrainz API. As it will provide access to many more users, the issue with keeping the database in hot state will be partially addressed.

Moreover, as per the initial proposal, once we have the baseline metrics implemented, it is not difficult to build hybrid metrics from baseline blocks. We can allow users to combine already transformed feature vectors into new hybrid similarity metrics with subsequent evaluation to find new high-performing metrics. There is great potential in such functionality to even find new approaches to compute similarity efficiently and precisely.

At the time of evaluating frameworks for the thesis, there had been a framework that had been overlooked due to its novelty and alpha availability - Annoy[1] by Spotify. As it evolved overtime, now it is a flexible alternative to PostgreSQL cube extension. However, due to the timing constraints it hasn't been evaluated and tested in the

[1] `https://github.com/spotify/annoy`

scope of this thesis. For the future work it can definitely be considered and its performance compared with PostgreSQL.

List of Figures

List of Tables

Appendix A

Code

All of the code used in this paper is open-source and available on GitHub:

- AcousticBrainz: `https://github.com/metabrainz/acousticbrainz-server/`
- AcousticBrainz fork with similarity implementation: `https://github.com/philtgun/acousticbrainz-server/`
- Various scripts to retrieve data from database, process it and plot figures that are used in the paper: `https://github.com/philtgun/smc-thesis/`

Bibliography

[1] Law, E., West, K., Mandel, M. I., Bay, M. & Downie, J. S. Evaluation of algorithms using games: The case of music tagging. In *ISMIR*, 387–392 (2009).

[2] Porter, A., Bogdanov, D., Kaye, R., Tsukanov, R. & Serra, X. Acousticbrainz: a community platform for gathering music information obtained from audio. In *16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, 786–792 (Malaga, Spain, 2015). URL `http://dblp.org/rec/html/conf/ismir/PorterBKTS15`.

[3] Bogdanov, D. *et al.* Essentia: an audio analysis library for music information retrieval. In *International Society for Music Information Retrieval Conference (ISMIR'13)*, 493–498 (2013).

[4] Kim, J., Won, M., Serra, X. & Liem, C. Transfer learning of artist group factors to musical genre classification. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, 1929–1934 (International World Wide Web Conferences Steering Committee, 2018).

[5] Swartz, A. Musicbrainz: a semantic web service. *IEEE Intelligent Systems* **17**, 76–77 (2002).

[6] Bogdanov, D., Serra, J., Wack, N., Herrera, P. & Serra, X. Unifying low-level and high-level music similarity measures. *IEEE Transactions on Multimedia* **13**, 687–701 (2011).

[7] Cano, P., Koppenberger, M. & Wack, N. Content-based music audio recommendation. In *Proceedings of the 13th annual ACM international conference on Multimedia*, 211–212 (ACM, 2005).

[8] Seyerlehner, K., Schedl, M., Sonnleitner, R., Hauger, D. & Ionescu, B. From improved auto-taggers to improved music similarity measures. In *International Workshop on Adaptive Multimedia Retrieval*, 193–202 (Springer, 2012).

[9] Seyerlehner, K., Widmer, G., Schedl, M. & Knees, P. Automatic music tag classification based on block-level features. *Proceedings of Sound and Music Computing 2010* (2010).

[10] Pohle, T., Schnitzer, D., Schedl, M., Knees, P. & Widmer, G. On rhythm and general music similarity. In *ISMIR*, 525–530 (2009).

[11] Pampalk, E. Computational models of music similarity and their application in music information retrieval. *PhD thesis, Technischen Universitat Wien* (2006).

[12] Gkiokas, A., Katsouros, V. & Carayannis, G. Deploying deep belief nets for content based audio music similarity. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*, 180–185 (2014).

[13] Foster, P., Mauch, M. & Dixon, S. Sequential complexity as a descriptor for musical similarity. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **22**, 1965–1977 (2014).

[14] Flexer, A. On inter-rater agreement in audio music similarity. In *ISMIR*, 245–250 (Citeseer, 2014).

[15] Schedl, M., Flexer, A. & Urbano, J. The neglected user in music information retrieval research. *Journal of Intelligent Information Systems* **41**, 523–539 (2013). URL `https://doi.org/10.1007/s10844-013-0247-6`.

[16] Font, F., Roma, G. & Serra, X. Freesound technical demo. In *Proceedings of the 21st ACM international conference on Multimedia*, 411–412 (ACM, 2013).

[17] Hellerstein, J. Gist: A generalized search tree for database systems. *UC Berkeley* 1–28 (1996).

[18] Shao, Y., Jin, Z., Wang, D. & Srinivasan, S. An auditory-based feature for robust speech recognition. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, 4625–4628 (IEEE, 2009).

[19] Valero, X. & Alias, F. Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification. *IEEE Transactions on Multimedia* **14**, 1684–1689 (2012).