SN Computer Science (2024) 5:78
https://doi.org/10.1007/s42979-023-02367-y

ORIGINAL RESEARCH

Parametrized Optimization Based on an Investigation of Musical Similarities Using SPARK and Hadoop

Savita Chaudhary[1] · V. Karthik[2] · R. Shankar[3] · Ayesha Taranum[4] · E. Naresh[5]

Received: 13 June 2023 / Accepted: 26 September 2023

Abstract

The big data processing framework Spark is used to power a parameterizable recommender system that can make recommendations for music based on a user's individual tastes and take into account a variety of musical tonal qualities. The system as it is presently built is completely scalable, which means that additional songs can be contributed to the data, the cluster size could be increased, and new types of audio information, in addition to more cutting-edge similarity evaluations, may be included. Another issue discussed in this research paper is the parallel collection of required audio characteristics on a computer cluster. Song recommendations for a dataset including more than 114,000 songs may be created on a Spark cluster with 16 nodes in under 12 s by integrating eight distinct audio feature types and similarity assessments. After the features have been retrieved, they are sent to the Spark-based recommender system to be processed. The calculated distance was displayed, examined, and graphically depicted. By computing the distance depending on the melody, rhythmic, and timbral components of the music, the final software controls song suggestion.

Keywords Big data · Hadoop · Music information retrieval · Optimization · Spark

Introduction

The primary goal of this work is to aid consumers in locating and obtaining music. There are currently numerous music services that provide this functionality. Apple&#39, iPod and the associated iTunes Music Store [1], which provides consumers with several ways to discover music, are particularly well-known. The portal recently sold its one billionth song [2]. Because of Amazon, iTunes, and other online stores, the way music is disseminated online is changing. Unlimited shelf space, highly effective suggestions based on consumer profiles, and 24-h availability render physical businesses uncompetitive [3, 4].

The purpose of this work is to suggest a transparent method for matching music that is based on a number of weighted characteristics rather than a predetermined combination. Applying different weights to various features enables similarity retrieval techniques to look for various similarities, allowing the user to select the features that are most significant to them and providing music recommendations

This article is part of the topical collection "Diverse Applications in Computing, Analytics and Networks" guest edited by Archana Mantri and Sagar Juneja.

✉ E. Naresh
naresh.e@manipal.edu

Savita Chaudhary
savitha_cs@sirmvit.edu

V. Karthik
karthikv.v1@gmail.com

R. Shankar
shankar@bmsit.in

Ayesha Taranum
dr.ayesha.tara@gmail.com

[1] Department of Computer Science and Engineering, Sir M Visvesvaraya Institute of Technology Bengaluru, Bengaluru, India

[2] Department of Information Science and Engineering, Ramaiah Institute of Tech, B'lore, India

[3] Department of Computer Science and Engineering, BMS Institute of Technology and Engineering, Bengaluru, India

[4] Department of Computer Science and Engineering, Vidyavardhaka College of Engineering, Mysuru, India

[5] Department of Information Technology, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal, India

Published online: 08 December 2023

based on their tastes. For instance, weighting a song's pace and beat more heavily than its melody allows for the development of playlists for exercise and sports, while melodic/timbral/other commonalities enable the search for songs from related musical subgenres [5].

The development of a parameterized similarity definition is made possible by the use of a big data framework like Spark. The musical distance between two pieces could be determined by combining and taking into account a variety of musical elements. This provides a more varied music recommendation system than those that are presently in place. In order to do this, a variety of features must first be retrieved from the audio data. In order to speed up operations, content (such as audio features) and context (such as listener behavior) data can then be supplied into a big data framework [6, 7]. But for this thesis, content-based information is the only thing that matters.

There has already been an extensive study on context-based collaborative filtering methods that employ big data frameworks and take into account other users' listening habits. However, the purpose of this thesis is to exclusively rely on the musical qualities of the songs in order to suggest a user-centered recommendation engine. There won't be any prejudice based on the musicians' popularity if the criteria are entirely based on the musical aspects of the songs [8].

Related Work

Based on how useful each piece is to a specific query, large collections of music or musical information are looked for it and structured using a process known as music information retrieval (MIR). This is especially important given the vast volumes of musical data accessible in digital format as well as the prevalence of music-related digital services. A growing community of multi-disciplinary scientists, such as library staff, software engineers, musicians, digital engineers, and music critics, among others, have come together, thanks to the Conference on Music Information Retrieval (ISMIR), a substantial conference series that was launched over the past four years. The majority of video streaming producers and sellers (like Philips, Sony, and Apple) also are fully engaged in the course's research, thanks to its evident commercial appeal, and many institutions are attempting to integrate MIR aid in some form in their digital services [9, 10].

In response to a textual query submitted by the user, like "David Bowie Heroes," simple MIR algorithms retrieve data. The technology basically becomes equivalent to any text-based web browser in certain situations by comparing the text with the textual data associated to albums and tracks (e.g., Google, Yahoo). Nevertheless, given the characteristics of the material be retrieved, there's a requirement for systems that really can accept "musical" queries, such as scores, sang tunes (query by hum), or audio input snippets (query by example). The latter scenario is the topic of my proposition [11].

The goal of search by instance is to identify musical compositions that are comparable to a particular audio document within a vast collection of digital music resources. The capability for investigation by example is a crucial prerequisite for MIR systems. It has a variety of challenges, including concerns with computation and complexity, test-bed design, and the requirement to select an appropriate format for the audio inside the query or music collection [80, 81]. The audio representation that is used affects the connections that the system can identify. Current representation selection techniques may be broadly divided into those that aim to evaluate high-level (like note, rhythm, etc.) or low-level (acoustic) similarities [12].

Low-Level Resemblance

Low noise similarity-based algorithms are frequently created to recognize a particular record even now in noisy settings and with significant signal degradation. These sound fingerprintings convert audio from a set of basic selected features into a more compact form with the help of a categorization system: Halker and Kalker [13] For logarithmically spaced sub-bands, Allamanche et al. use quantization and Fourier coefficients. Battle and Cano employ MPEG-7 low-level spectral properties, although [6, 14] use Mel Frequency Cepstrum Coefficients (MFCC), followed by decoding Hidden Markov Models, to get the appropriate labeling (HMMs). It is possible to extract these features from the signal using frame-by-frame analysis, which needs little to no musical theory or expertise.

This method has shown exceptional ability to identify a flawless one-to-one connection between both the audio query as well as a record in the database, regardless of whether the audio query has indeed been compromised by compression or background noise [15]. It has been employed commercially for music recognition for end users and radio broadcast monitoring [16]. Yet, acoustic similarity studies exclude any relationship to the sound's musical characteristics. As a result, even if two recordings of the same song employ the same singer and equipment, if they are different from one another, they cannot be near-matched in a similarity-ranked list. Low-level similarity assessments make it challenging to locate musically pertinent close matches [17].

High-Level Resemblance

As an alternate to limited similarity for music retrieval, we may aim to develop high-level representation from sound that highlights musical similarities across recordings.

Our strategy in the OMRAS project, which used high-level representations, showed some effectiveness in finding musical similarities [18]. It was based on the contrast between the representation of the audio inquiry and a collection of symbol information (polyphonic music scores). The approach was based on automated transcribing of music, which involved the conversion of aural information into a powerful symbol approximating a score. As this conversion is usually inaccurate, a one-to-one matching of transcription or database events is ineffective for retrieval purposes [19]. Harmonic distributions were generated from the transcribed or stored data, as similarity was assessed by measuring the difference between the two distribution within a single event space. OMRAS [90] was the first system to be able to successfully extract polyphony score from polyphonic audio requests. Moreover, OMRAS provided meaningful near-matches with high similarity scores ("similar" songs).

Automatic transcription and harmonic modeling techniques are constrained by the sorts of instruments and music that may be analyzed. While not usually connected to the sonic quality of recorded music, it has a lot to do with the way formal music notation is used [20]. The range of workable musical queries is constrained when an extremely high level of information is used for retrieval that is musically acceptable. This is especially important since a musical piece is better represented in a performance than it is in a score (e.g., pop music as opposed to classical music). In order to accurately identify musical similarities, we need a different representation from the low-level data that is not important to music and the imperfect and constrained high-level music theory. We propose mid-level representations as this alternative [21].

Mid-level descriptions of music are measures that result from the transformation of an audio signal into a significantly subsampled function that describes the properties of musical constructs in the original signal. This strategy is important to a vast class of musical methods for processing signals (e.g., onset and pitch detection, tempo and chord estimation). These approaches were developed after a thorough examination of musical knowledge and human perception [22].

Without being bound by the constraints put by the laws of music notation, mid-level representation can effectively describe the rhythm framework of a piece and achieve higher levels of semantic richness than low-level elements. These capabilities effectively represent a broad variety of musical signals, including numerous musical styles and genres, according to our work on onset detection [23].

Initial attempts to offer additional relevant representation for retrieving included the use of spectral envelopes to identify timbral similarities [24, 25] and attempts to record the beat of a song utilizing periodicity histograms [26] or temporal sequences [27]. However, the chosen feature sets' simplicity—which may still be classified as low-level—limited their popularity. Recent advancements in semantics of music, which include ours in our opinion, make it possible to construct a set of features which is more artistically pertinent.

We categorize mid-level portrayals into two groups: segment-based mid-level representations, which also categorize characteristics of lengthier melodic segments like melodic line, tranquilly, chorus, etc., as well as event-based mid-level representations, which also classify characteristics of particular musical occurrences like note onsets and pitch detection. Moreover, we suggest that the former could be thought of as a language that creates the latter. This is demonstrated by our work on tempo estimate, which successfully uses onset detection methods to create tempo contours for a range of input.

Our organization is also looking into tick forecasts for beat tracking and pitch contour lines in monophonic as well as polyphonic environments, furthermore to onset detection systems utilizing high-frequency components, spectroscopic difference, complicated spectral distinction, step deviation, wavelet familiarity modulus, as well as great shock reinforcement learning. Moreover, we characterize music structure using segment-based mid-level representations like tempo outlines, harmonic curves for tonal assessment, or texture scores for long-term structure segments.

Methods

Dataset

Archive of Free Music

The largest dataset is the Free Music Archive (FMA), which has 117,123 distinct tracks comprising over one terabyte of music data from various music genres. For the majority of the songs, there is also a wealth of metadata, such as genre tags [27].

Private Music Library

Metal music makes up the majority of the private music collection utilized in this work. All rights to the music, which were properly acquired, belong to their respective owners. This dataset cannot be made public in association with this research project as a result. On the other hand, the personal music library has been accurately categorized,

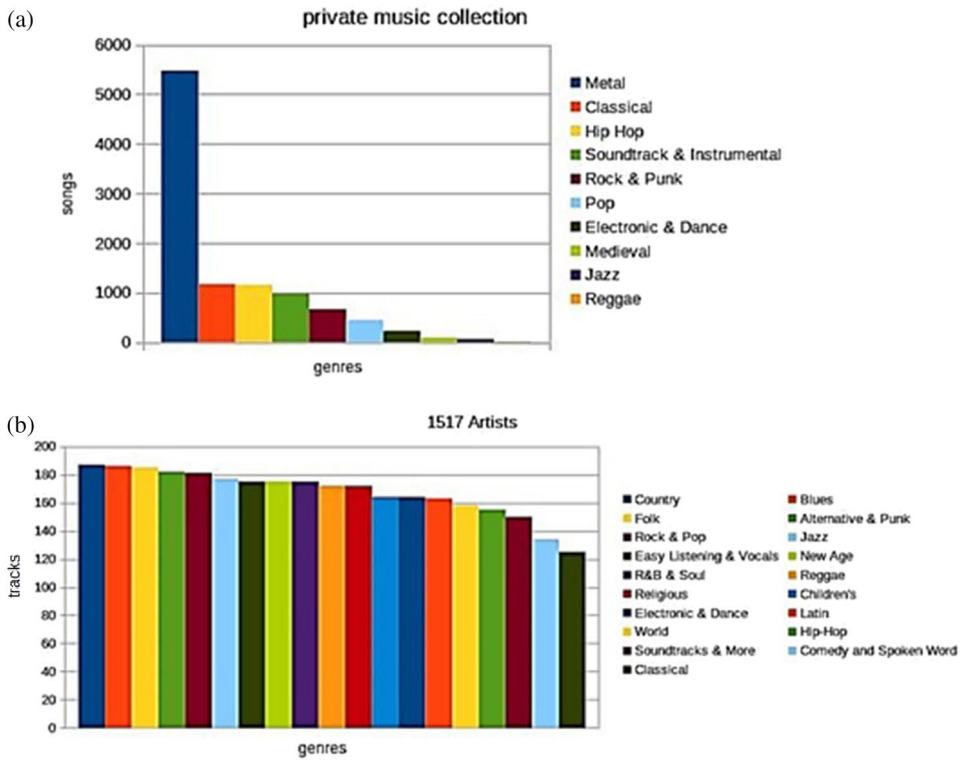

Fig. 1 **a**, **b** Musicnet and 1622-Artists [29]

and the accompanying PDF file is attached as one of the appendices. Shows how many songs in each genre there are in this sample (b). Additionally, a private recording dataset with original music and ambient recordings was used. You may find the majority of these files on SoundCloud [28]. The private music collection must be included in order to allow a subjective assessment of the results from the built recommendation engine because music suggestions are continually based on personal taste and judgments of the quality of the results may vary (Fig. 1).

The 1622-Artists dataset as well as the Musicnet dataset are two alternative sources of music. The 1622-Artists dataset has 3318 songs from diverse genres, while the Musicnet dataset contains 332 items of classical annotated with harmonic progression values. Since MedleyDB pitch estimation may be performed by device or instrument, multitrack data could be useful for melodic or pitch-based similarity.

Covers

The covers dataset, which includes 80 original songs, mostly from the rock and pop musical genres, as well as 80 cover versions, is made available for cover song identification analysis. In regard to musical style, rhythm, and timbre, these cover versions are vastly different from the original.

Spotify API

The Spotify API is another resource for locating audio characteristics, information, and music samples. The downside of utilizing the Spotify API is that no fully loaded, prepared test sample with all the essential characteristics is available. As a result, in the interest of conducting studies, a test dataset must first be produced. The relevant data may be obtained easily via a simple Python tool called Spotify [30]. A simple script is included during experimentation that can be used to collect all audio characteristics and analysis data from specified songs in a playlist that contain a sample URL for a 30-s audio excerpt. JSON files are used to contain data about the following audio attributes and analyses: danceability, acousticness, instrumentality, liveness, intensity, speechiness, polarity, projected keys, tempo, and volume.

Figure 2a shows the returning chroma aspects of Beethoven's "Fu r Elise," while Fig. 2 (b) shows the song's introduction in more depth with green dots that approximate estimated bar markers. The note values for an octave are shown by blue dots. These may approximate a number between zero and 10, where 0 represents the key C, and 11 represents a B, according to this. A segment is a collection of samples with a consistent timbre and harmony. For every one of the semi-tones per segment, the Spotify API actually returns a chroma feature value. The plots, however, only show the key that is mostly prevalent inside each segment to emphasize the main melody line [30].

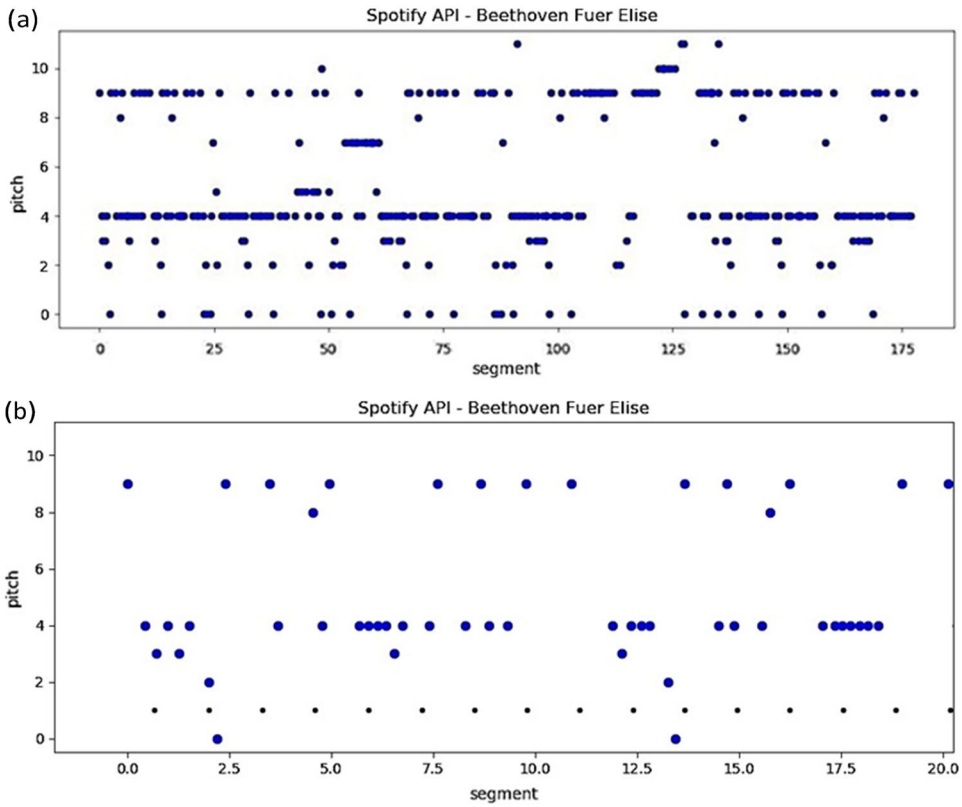

Fig. 2 Spotify APIs

Spotify could offer the all data required to create a sizable dataset for MIR, in addition to the 30-s audio samples that other features like MFCCs may be gathered. Crawling the Spotify services is clearly forbidden by the terms and conditions.

So, utilizing the Spotify API to increase the dataset is not possible without running the risk of breaking the law. Data mining and information crawling can differ from one another, and these limitations might not be applicable for tiny datasets. If used commercially, creating an algorithmically created playlist identical to the "Discover Weekly" playlist could result in legal issues, as per Spotify. Nonetheless, it does not forbid use in non-commercial settings [26].

The Spotify API will not be used in this study to produce a test dataset because the Spotify API developer team did not respond to an early enquiry. Without additional contact with Spotify, it is not possible to build a test dataset using the Spotify API.

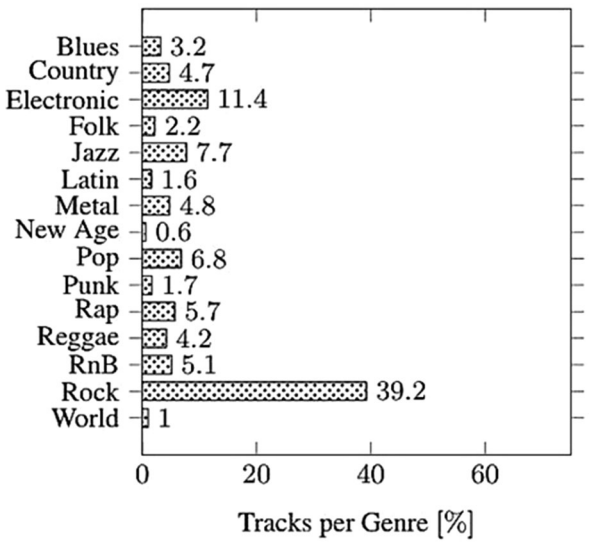

Fig. 3 Million Song Dataset with genre

Dataset of a Million Songs

Another exceptional and huge collection is the Million Song Dataset. It also incorporates additional auxiliary datasets, such the Last.fm dataset and the tantrum genre annotation, in addition to a significant quantity of metadata for each music. Pitch, loudness, vitality, and danceability are just a handful of the additional audio properties that are part of the Echo Nest API collection. The SecondHandSongs Data, containing an inventory of covers discovered in the Million Song Dataset, is another contribution (Fig. 3).

Because of Spotify API includes audio elements from the Echo Nest, the MSD in a big data scenario would provide a simulation of dealing with Spotify information without the need to mining the actual data. The MSD has

previously been utilized with big data systems to retrieve music similarity according to metadata and user input. Although the MSD's original absence of audio recordings, 30-s samples may be produced once the data was made accessible using simple programs from 7digital.com. However, users can no more install the 28-s attached file from 7digital, rendering this dataset useless for this research endeavor because omitted audio properties like MFCCs cannot be deduced from the sound recordings themselves [31].

Big Data

Following an evaluation of various sources of data as well as the presentation of many strategies for extracting and processing various audio aspects [32], the next part offers analysis of data using big data processing platforms, such as Apache Spark and Hadoop. Most of the Spark and Hadoop basic knowledge in the following sections is drawn from Jeffrey Aven's book, "Data Analytics with Spark Using Python," which provides a clear and informative introduction of the field of big data processing utilizing PySpark.

Hadoop

The need for toolkits and efficient algorithms to manage the large volumes of high-dimensional data that are becoming more and more accessible has grown over the past few years. One method for resolving big data difficulties is to use parallelism.

Early in 2001, search engine providers like Google and Yahoo first ran into the problem of using "internet-scale" data as they struggled to store and handle the ever-growing volume of indexes from internet-based publications. In 2002, Google released a white paper titled "The Google File System." Google created the programming paradigm known as MapReduce as a response to the problem of internet-scale data, which was first detailed in the paper "MapReduce: Simplified Data Processing on Large Clusters" in 2006[33].

Mike Cafarella and Doug Cutting were working on the "Nutch" web search project at the time. Motivated by the two publications, Cutting combined Google's processing and storage techniques to build Hadoop. In 2006, Hadoop became a member of the Apache Software Foundation. Hadoop's basic concept is the MapReduce programming model for data processing. With big computer clusters, a scalable application known as Hadoop may be used. It does not require a supercomputer environment and may be operated on commodity hardware clusters. To store data tediously on other nodes, a variable replication factor is employed, and it regulates how so many copies of each data block are kept there. As a consequence, errors can be handled simply by restarting.

MapReduce

The incoming data are divided into chunks and disseminated among cluster nodes in the first step. This is often maintained by a file system with distributed storage such as HDFS. All information chunk identifiers are maintained on a single master node. The data is subsequently sent to the mappers, who process it before converting it to key–value tuples. Before being sent to the reducers, the key–value pairs are usually aggregated in an intermediate step by their keys. Every tuple that uses the same key is subjected to a unique procedure by the reducers.

The "replication rate" is defined as the total amount of key–value pairs produced by all map makers divided by the entire number of data input (r). The largest number of values for a single key entered into a reducer is denoted by the letter "q." (reducer size). There are usually trade-offs between a high replication rate (r) and a small reducer size (q), both of which are highly parallel and generate more network activity (less network traffic, but reduced parallelism owing to a smaller total reducer count).

Spark

When compared to more recent options like Spark, utilizing Hadoop as a platform for big data processing has some disadvantages. The 2008-launched Spark project was a part of the Mesos research project. In place of Hadoop's MapReduce implementation, it was developed. Spark supports native support in addition to being created in the Java Virtual Machine (JVM)-compatible programming language Scala.

Using Big Data Frameworks to Compare Music

One-to-many-item similarities are useful to estimate the similarity. Because of this, only one song may have all of the commonalities to that song between other songs computed concurrently. This was the method of inquiry employed in this study. Making a detailed similarity matrix beforehand is an alternative (all-pairs similarity). Nevertheless, it would take a while to use vast databases including millions of songs. Merging the two techniques includes calculating the similarities for each song demand individually and keeping the outcomes in a sparse similarity matrix to expedite subsequent requests for the same songs. Yet this is outside the purview of the ongoing investigation. The choice to utilize Spark is explained in the short overview to big data framework that follows.

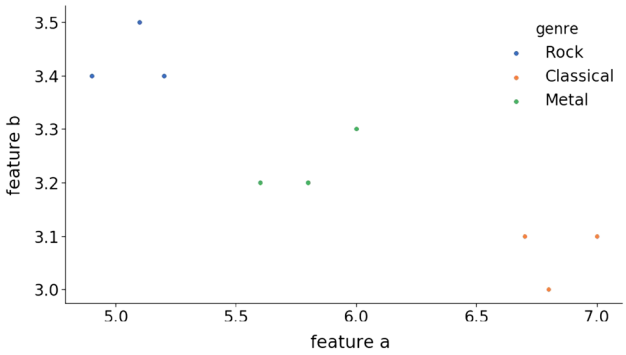

Fig. 4 Feature space illustration

The shared nothing method of Spark is used to calculate the "one-to-many-item" similarity. Concurrent distance computation is made possible by the fact that the qualities of different songs are wholly different from one another. The greatest and the lowest values must total in order to scale the outcome. In order to return the top results, a sorting technique must be utilized. Contrary to conventional methods, which require data shuffles, all the characteristics may be dispersed throughout a cluster, and the data localization principle can be used to independently assess how similar two songs are. With regard to very big datasets, this provides a completely scalable solution [34]. The audio characteristic data may also be properly cached in main memory using Spark. Interactive sequential song requests could be fulfilled without constantly reading feature data from hard drive if all of the characteristics from across all songs can fit within the cluster's main memory. One restriction is that Spark itself is unable to handle or read audio files. Just the recovered features must be delivered after the feature extraction procedure, which must be done independently.

Results and Discussion

Correlation of Features and Distance Distribution

This section analyzes the similarity analysis results to identify how the ranges from various feature sets connect to one another as well as how they are spread from across unit interval [0, 1]. To evaluate this, a test dataset was generated from the distances the Spark program returned. 97 songs were chosen at random from the 1619-Artists dataset, and the locations of each song relative to the others were calculated. The sample consists of 3219 songs that are uniformly distributed among more than 15 different genres. It is imperative to sample distances from different genres when examining the distribution of distances. Different distances and distributions apply depending on where the real music is in the feature space. Songs chosen from the feature space distribution's periphery will have different distances from songs chosen from its center.

The framework demonstrated in [35] offers a straightforward example to demonstrate this. In this scenario, the distances between a song from the "Classical" category and the "Rock" or "Metal" tracks are different, although the distances between songs with the "Metal" tag and those with the "Rock" and "Classical" tags are almost the same. Rock music and classical songs are separated by twice as much as metal songs are [104]. The connection between the lengths from different feature categories is depicted in Fig. 4. The eight different lengths for each song pair are totaled up using the conventional equation, with all weights set to 1. This results in a single new combined distance. This entire distance is shown by the letter "agg" in the following charts. The multiple rhythmic characteristics and the JS and SKL

	rp	rh	bh	js	skl	mfcc	chroma	notes	agg
rp	1	0.918345	0.258626	-0.0131253	0.0357719	0.105182	0.0455418	0.00375641	0.752988
rh	0.918345	1	0.192452	0.0207377	0.0443187	0.150032	0.0396717	-0.00201152	0.7558
bh	0.258626	0.192452	1	-0.203041	-0.160113	-0.0695903	0.0286554	-0.00464233	0.323581
js	-0.0131253	0.0207377	-0.203041	1	0.747947	0.0894321	-0.021468	-0.00046403	0.435151
skl	0.0357719	0.0443187	-0.160113	0.747947	1	0.0580153	-0.0458679	0.0222944	0.461898
mfcc	0.105182	0.150032	-0.0695903	0.0894321	0.0580153	1	0.047422	0.0705918	0.378666
chroma	0.0455418	0.0396717	0.0286554	-0.021468	-0.0458679	0.047422	1	0.169881	0.142827
notes	0.00375641	-0.00201152	-0.00464233	-0.00046403	0.0222944	0.0705918	0.169881	1	0.25369
agg	0.752988	0.7558	0.323581	0.435151	0.461898	0.378666	0.142827	0.25369	1

Fig. 5 Correlation matrix

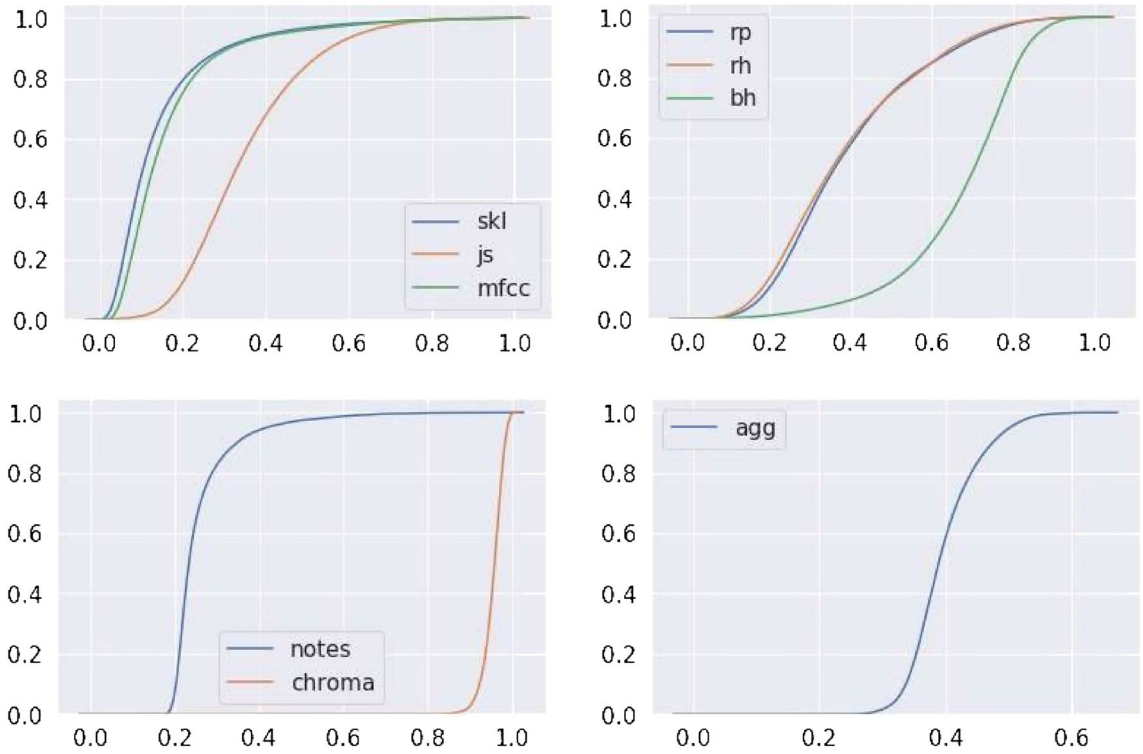

Fig. 6 Cumulative histogram

components all easily complement one another. Only a tenuous relationship exists between the musical components.

Figure 4 provides a simple illustration to illustrate this description. While the distances between songs with the "Metal" tag and those with the "Rock" and "Classical" tags are roughly the same, in this case, the distances between a song from the "Classical" genre and the "Rock" or "Metal" tracks are different. The distance between the Rock songs and the Classical songs is double that of the Metal songs.

The connection between a feature type and the total distance from the weighted sum shows how much of an impact that feature type has (Fig. 5). The total distances are altered differently based on the type of feature even though distances are not all spread equally over the unit interval. Figure 6 uses

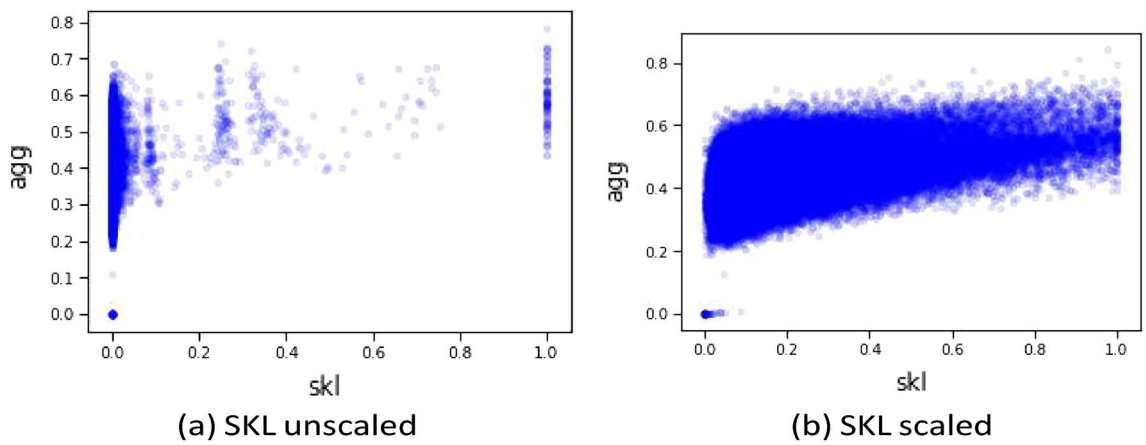

Fig. 7 shows how the weighted total is affected by SKL scaling

the cumulative histograms across the unit interval to show how the distances are dispersed. It is particularly clear that cross-correlation lengths really aren't evenly distributed.

In Fig. 7a and b, intriguingly, there is a declining association between the combined distance ("agg") and the Jensen–Shannon-like divergence. One explanation is that although the SKL and JS distances are strongly connected, the inadequate scaling prevents the SKL from having any bearing on the overall distance. The total weighted sum of the JS divergence results cannot be affected by the results from the JS divergence alone.

Using the entire scatter plot matrices of the varying distance for the 95 song samples from diverse genres, Figs. 8, 9 show the correlation and dispersion of the distances. The histograms of the separations from the relevant special feature sets are shown on the principal diagonal. It reveals that every sort of feature, excluding chroma properties, substantially correlates with the weighted sum of all features. The scatter plots make it easy to see the symmetrical Kullback–Leibler diverge, the Jensen–Shannon-like divergence, as well as the significant association between both rhythmic patterns and rhythm histograms.

Identification of a Cover Song

Only MFCC-based recommendation system can identify cover tunes. Algorithms for comparing melodies, The Scorpions' rendition of Knightsbridge's song "Rock you like a Hurricane" were found by the Spark implementation to be the top suggestion when using the cross-correlation during the initial testing on the whole dataset, which comprised 121,722 songs. Out of more than 121,722 tracks, the program was nevertheless able to find a top recommendation for a distinct version of the music "Fu r Elise" cover.

1. Mozart's "Turkischer Marsch" for piano solo (private collection)
2. RONDO ALLA TURCA KV 331 by FRITZ STEINEGGER (1517-Artists)
3. 136071-2Kutup, "We Shall Cuddle Up And Sleep" (FMA dataset)

Fourth, Sean Bennett's Iterations of the Turkish March (1517-Artists)

5. Mozart's D-minor Fantasie (1517-Artists)

Although there are two more variations of this song in the private music collection, a second test used simply the JavaScript and chroma properties without the rhythm patterns. Below is another list of the top six suggestions:

1. Song request: Alla Turca (Allegretto) (Private Collection), Mozart, 100 Meisterwerke der Klassik, JS + CHROMA
2. Mozart Collection CD31, KV331-3, Alla turca allegretto (private collection) Piano Collection / CD25 - Mozart - Alla Turca Allegretto (private collection)
3. Tu rkischer Marsch by Mozart, played on the piano (private collection) FRITZ Steinegger's Rondo Alla Turca KV 331 is number four (1517-Artists) Fifth, 136071 (2Kutup - We Shall Cuddle Up And Sleep) (FMA dataset) Variations on the Turkish March by Sean Bennett, No. 6 (1517-Artists)

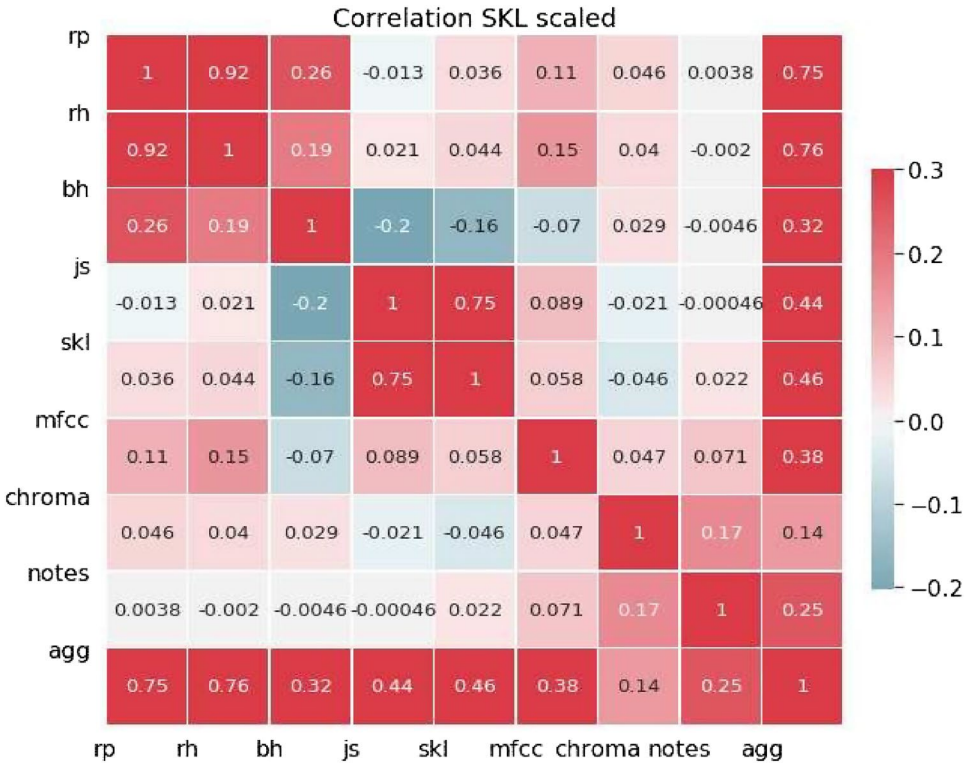

Fig. 8 illustrates the correlation of features based on SKL scaling

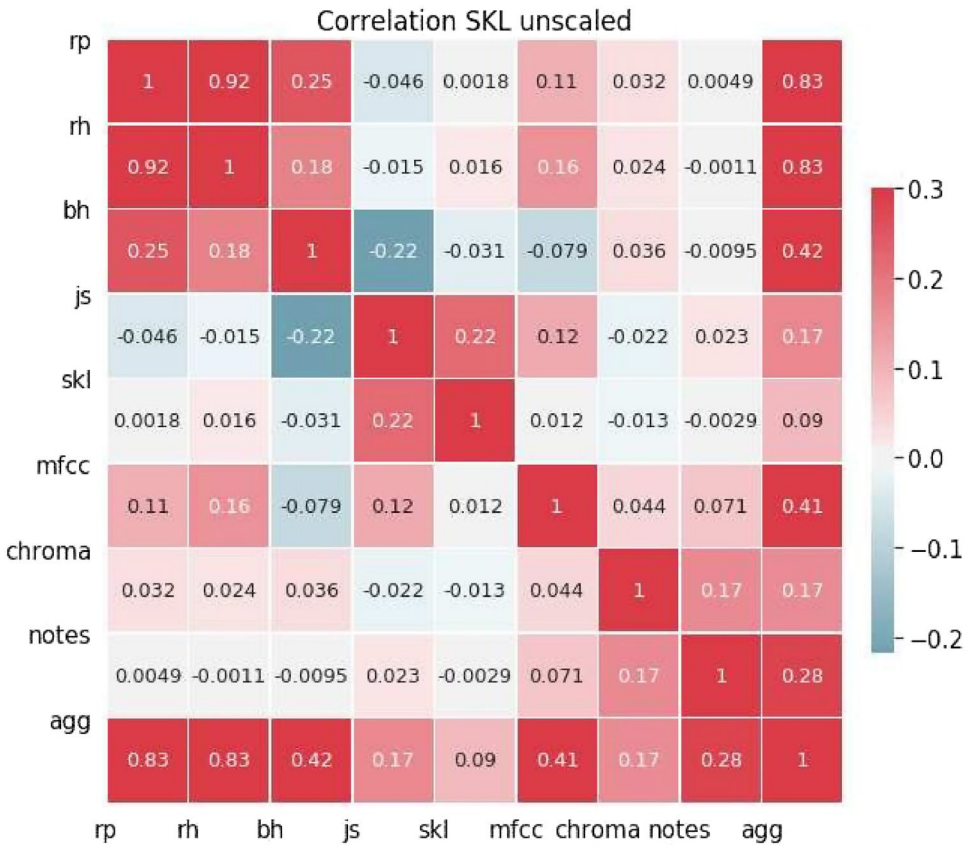

Fig. 9 illustrates the correlation of features based on SKL Unscaling

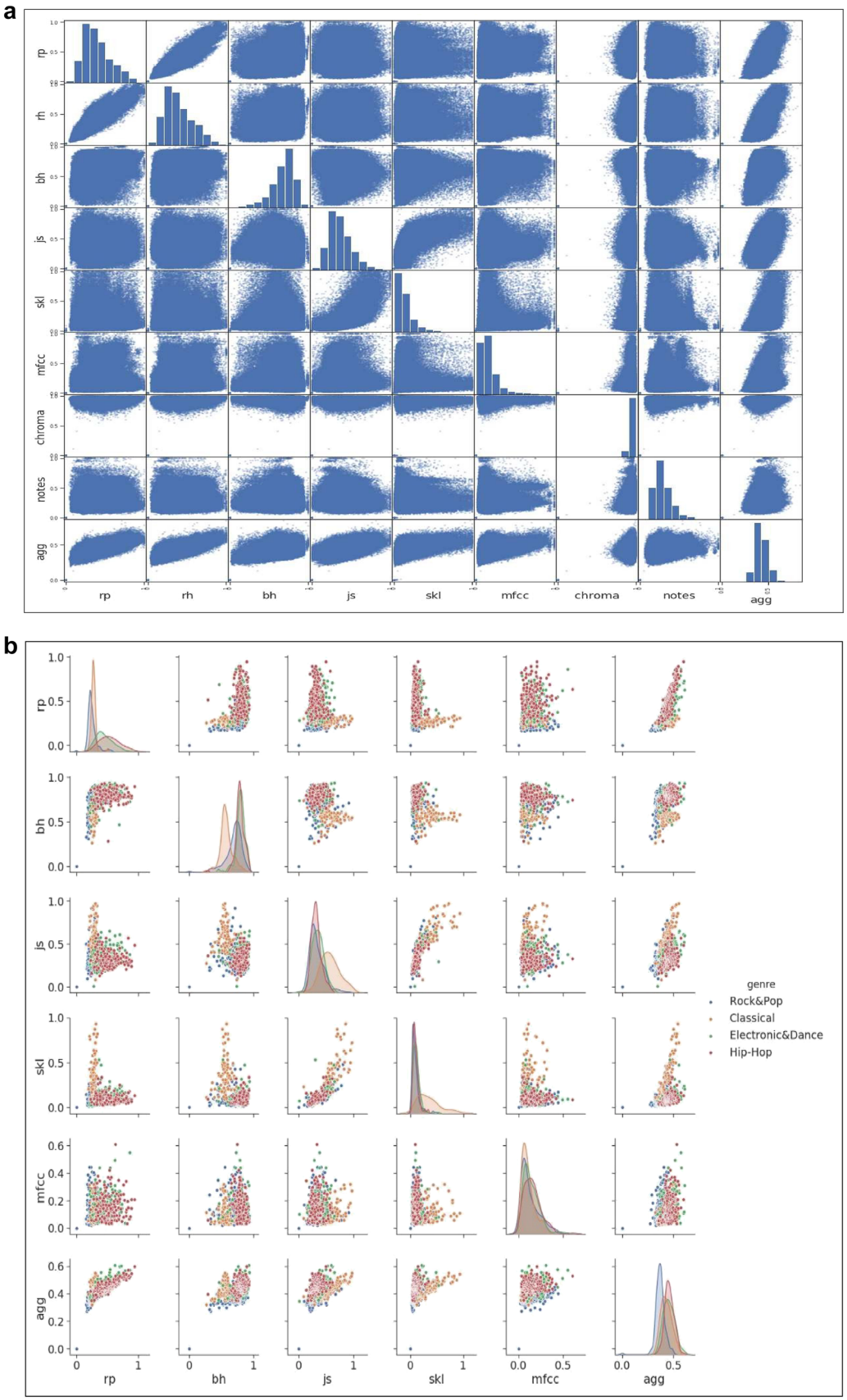

Fig. 10 **a** 97 tracks, more than 15 genre), 1619 artists are represented Scatter matrix, correlation. **b** Representation of track for genres

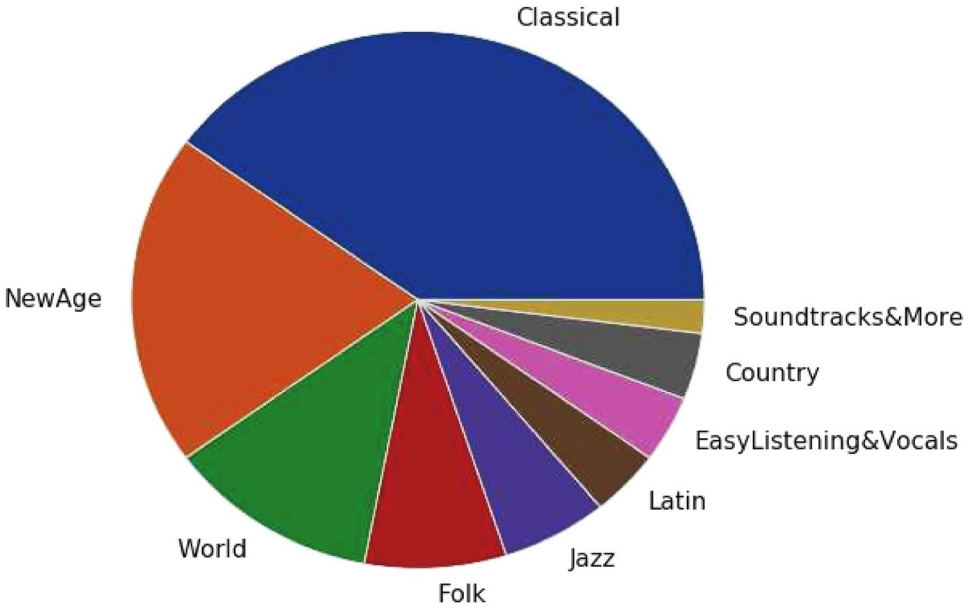

Fig. 11 Scatter matrix, distances for 1517 artists, 1 random Rock & Pop song, and 4 genres

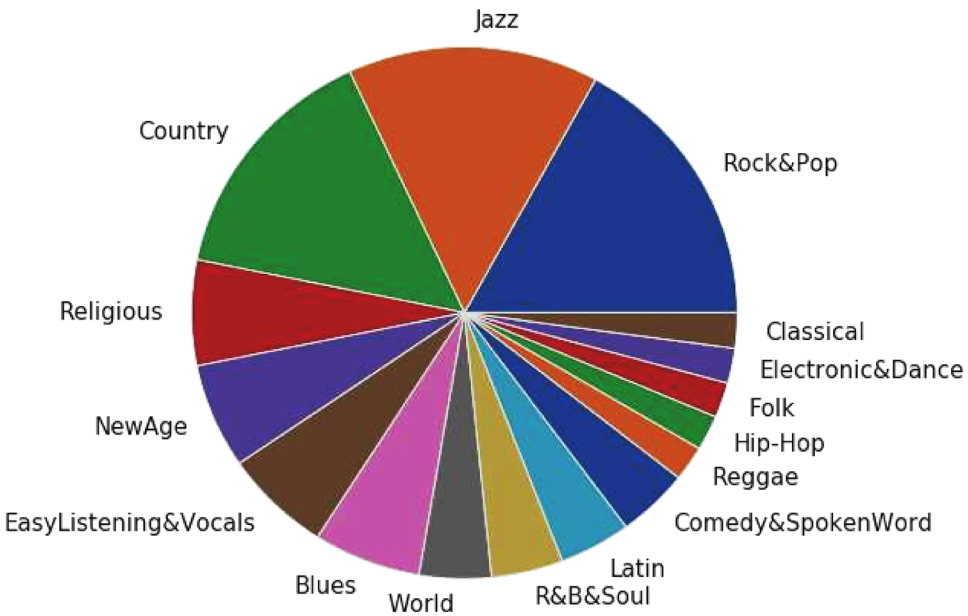

Fig. 12 Scatter matrix, distances for 1517 artists

The initial alternate record showed up as the 13th suggestion in a third inquiry whenever the other recordings could only be found using the Jensen–Shannon-like divergence. This supported the hypothesis that timbral features, the Jensen–Shannon-like difference, as well as the symmetrical Kullback–Leibler deviation are helpful for identifying cover tunes.

The cross-correlation, however, is unable to detect the cover song for some song requests. After the positive results of the initial testing, the cluster was loaded with the covers dataset specified in cover songs. The music requests for the 80 "A-versions" songs were given to the Spark program, and the resulting closest neighbors were looked at.

Although 28 from out 78 identified songs don't always appear to be a remarkably high strike rate as well as isn't quite as excellent as the original paper's results, it should be noted that most of the cover songs in the cover80 set of data markedly deviate from original tracks in regard to musical fashion, measuring instruments, rhythm, or even genre. As a result, cross-correlation is much more exact, yet more computationally efficient.

Genre Relativeness

Occasionally, a single song from a more "contemporary" genre—such as Hip Hop, Rock & Pop, Electronic & Dance, or Reggae—will play. Similarly, five songs from the Rock & Pop genre were put to the test. The results from the 1517-Artists dataset are dispersed throughout 17 of the 20 major genres as shown in Fig. 10a. This may be due to the fact that the songs in this dataset that have "Rock & Pop" annotations originate from a wider range of subgenres. If you look closely at the dataset, you can see that, for instance, Metal tracks are also classified as Rock & Pop as shown in Fig. 10b.

Another experiment was conducted to examine the effects of various feature types on the overall recommendations and to demonstrate how the distribution of distances differed by genre. A part of the 1629-Artists dataset that contained songs from the "Classical," "Hip Hop," "Electronic & Dance," and "Rock & Pop" categories was used to compute all distances to the songs for requests for a single song. Figure 11 shows the scatter matrices for all distances from a single Rock & Pop music request. The genre of the suggested music affects the recommendations' various distances. The calculated kernel density for the pertinent feature type is shown diagonally. The JS distances can tell classical music apart from the other types of music. However, it's interesting to notice that it cannot tell Rock/Pop songs from Hip Hop songs. Nevertheless, rhythmic patterning cannot distinguish musical instruments from pop and rock music on their own. Nonetheless, all three genres may be distinguished when all feature kinds are taken into account. Three sets of songs from various genres are shown by the scatter diagram of the rhythm pattern variations and Jensen–Shannon-like divergence combined. Regardless of the feature set employed, the fourth genre, "Electronic & Dance," can't be distinguished from hip hop music. But it's crucial to remember that all of these gaps resulted from a Rock/Pop song request.

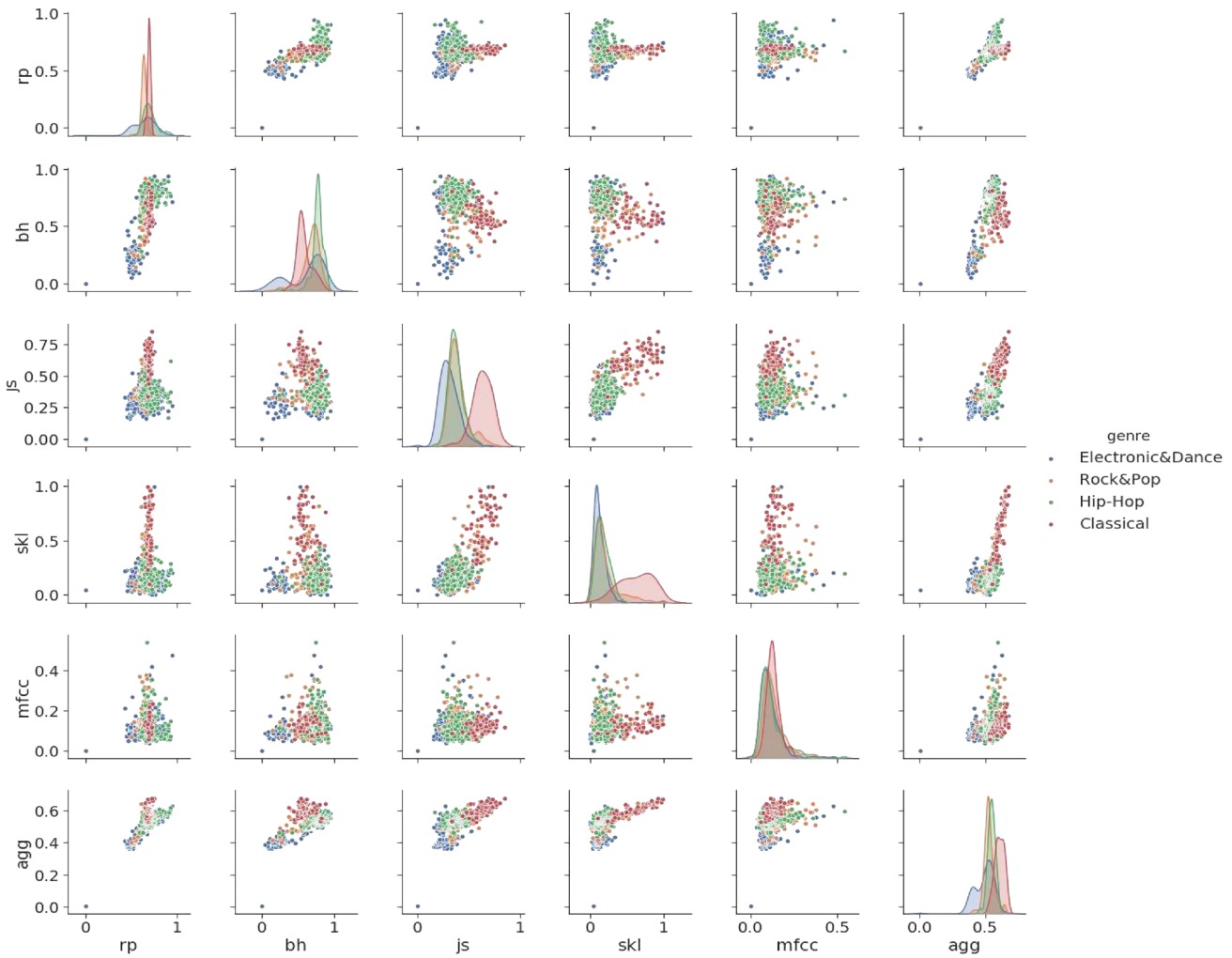

Fig. 13 Randomly selected Electronic song

The arrangement of the distances varies based on which part of the feature set the song request is situated. It appears that there is a similar distance between the desired Rock/Pop song and music in the Hip Hop and Electronic/Dance genres. When distributing the distances, a song from the Electronic/Dance genre sounds entirely different. The weighted total of all characteristics, includes cross-correlation and Levenshtein distance that are not depicted in the plots, and is represented by the "agg" graphs. In Figs. 12 and 13, the aggregate findings of all feature classes primarily propose further Rock & Pop songs.

The Spark recommender system would have no way to distinguish between all four of the various genres when employing a single feature type A list of suggestions that is overall satisfying may only be recovered when various rhythmic and timbral features are merged.

Features of Rhythm

The availability of music that is played at nearly the same pace is another essential requirement for a recommendations engine. The distances between the results of two song request that have been made to assess the effectiveness of the rhythm characteristics are displayed in Fig. 14 for the dataset 1517-Artists. The scatter plots demonstrate how close the beat histogram and rhythmic patterns connects to the music's overall BPM. As each of the eight different feature types has a "agg" value, it is feasible to see how the rhythm features have affected the suggestions overall (the weighted sum). Overall, the weighted sum of rhythmic elements makes it more likely that the Spark recommendation engine would propose songs with comparable BPM. Further evidence that the BPM is not the only important element affecting the distances is shown by the request for a classical song in Fig. 14d.

Conclusion

The proposed study offered an insight into the subject of music information retrieval. There were also added multiple high- and low-level audio aspects that were described in numerous methods for assessing the similarities of audio clips depending on audio features. A brief review of big data technologies, such as Apache Spark and Hadoop, also was provided, and several resources of acoustic data were obtained. From audio recordings, timbre, rhythm, and melodic components could be retrieved and processed

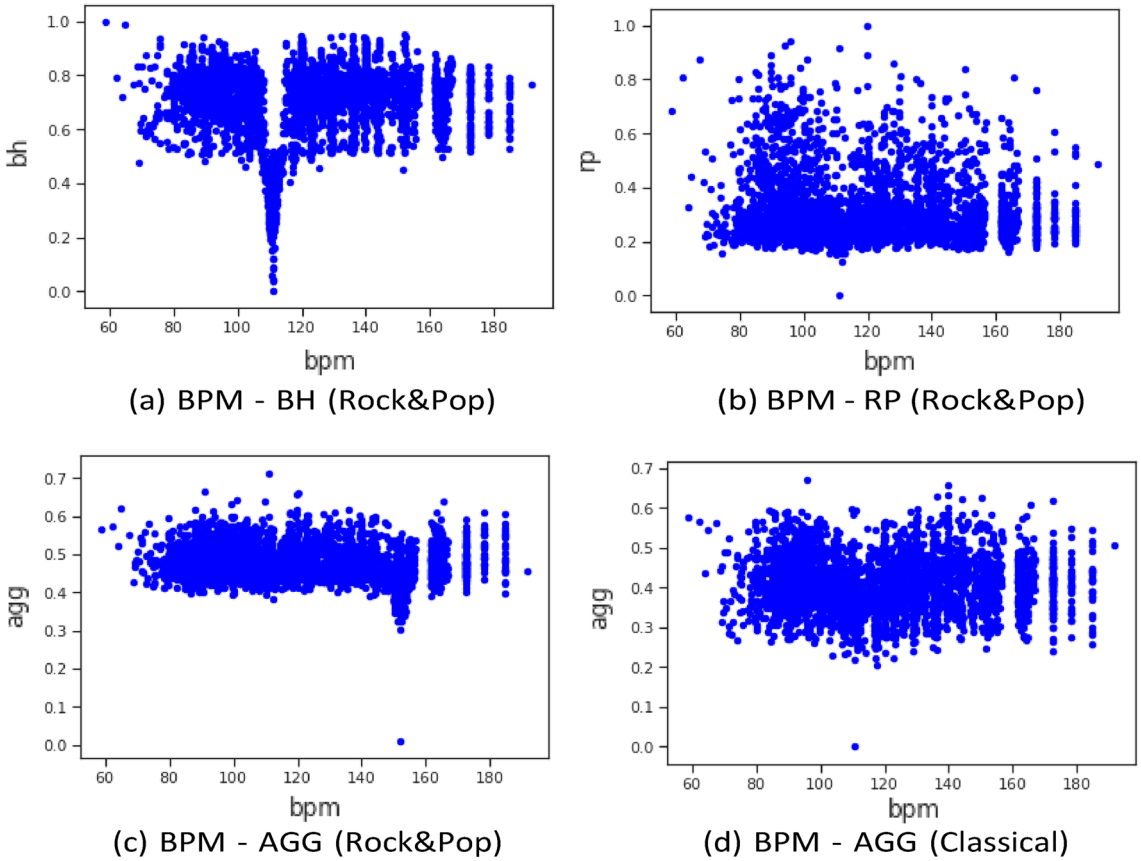

Fig. 14 Random Rock & Pop and Classical tracks

beforehand. Several alternative methods were used to calculate the separations between the features that were collected. The implementation can be prepared using the theoretical understanding from the preceding chapters. Around 1 TB of music tracks with 123,767 different songs was combined from collected data.

To prepare the data to be utilized by the big data processing platform Spark, the relevant audio characteristics were extracted and pre-processed in parallel utilizing MPI on a computer cluster. As an illustration, the melody and chroma information was separated. The Spark framework was used to generate, test, analyze, and improve numerous similarity measures, and the features were added to a cluster's HDFS. Many methods (RDD, Filter, DataSet, Refined, and Cluster Configurations) were analyzed using Spark, as well as the runtime was assessed. The calculated distance was displayed, examined, and graphically depicted. By computing the distance depending on the melody, rhythmic, and timbral components of the music, the final software controls song suggestion. The user could prioritize various features of the music using the customizable suggestions. The system is scalable. The cluster size can be increased, new songs can be added, different audio aspects can be included, and more advanced similarity tests can also be applied.

Funding Open access funding provided by Manipal Academy of Higher Education, Manipal.

Data Availability The sample data set information is included in the article that support the findings of this research.

Declarations

Conflict of Interest The authors do not have any conflicts of interest in this article.

References

1. Gupta A, Thakur HK, Shrivastava R, Kumar P, Nag S. A big data analysis framework using apache spark and deep learning. In: 2017 IEEE international conference on data mining workshops (ICDMW). IEEE. 2017. pp. 9–16
2. Assefi M, Behravesh E, Liu G, Tafti AP. Big data machine learning using apache spark MLlib. In: 2017 IEEE international conference on big data (big data). IEEE. 2017. pp. 3492–3498
3. Fu J, Sun J, Wang K. Spark–a big data processing platform for machine learning. In: 2016 international conference on industrial informatics-computing technology, intelligent technology, industrial information integration (ICIICII). IEEE. 2016. pp. 48–51
4. Ghasemi E, Chow P. Accelerating apache spark big data analysis with fpgas. In: 2016 Intl IEEE conferences on ubiquitous intelligence & computing, advanced and trusted computing, scalable computing and communications, cloud and big data computing, internet of people, and smart world congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). IEEE. pp. 737–744.
5. Han Z, Zhang Y. Spark: a big data processing platform based on memory computing. In: 2015 seventh international symposium on parallel architectures, algorithms and programming (PAAP). IEEE. 2015. pp. 172–176
6. Maheshwar RC, Haritha D. Survey on high performance analytics of bigdata with apache spark. In: 2016 international conference on advanced communication control and computing technologies (ICACCCT). IEEE. 2016. pp. 721–725
7. Hazarika AV, Ram GJSR, Jain E. Performance comparision of Hadoop and spark engine. In: 2017 international conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC). IEEE. 2017. pp. 671–674
8. Wang G, Xu J, He B. A novel method for tuning configuration parameters of spark based on machine learning. In 2016 IEEE 18th international conference on high performance computing and communications; IEEE 14th international conference on smart city; IEEE 2nd International conference on data science and systems (HPCC/SmartCity/DSS). IEEE. 2016. pp. 586–593
9. Gu L, Li H. Memory or time: performance evaluation for iterative operation on hadoop and spark. In: 2013 IEEE 10th international conference on high performance computing and communications & 2013 IEEE international conference on embedded and ubiquitous computing. IEEE. 2013. pp. 721–727
10. Samadi Y, Zbakh M, Tadonki C. Comparative study between Hadoop and Spark based on hibench benchmarks. In: 2016 2nd international conference on cloud computing technologies and applications (CloudTech). IEEE. 20167. pp. 267–275
11. Dev D, Patgiri R. Performance evaluation of HDFS in big data management. In 2014 international conference on high performance computing and applications (ICHPCA). IEEE. 2014. pp 1–7
12. Veiga J, Expósito RR, Pardo XC, Taboada GL, Tourifio J. Performance evaluation of big data frameworks for large-scale data analytics. In: 2016 IEEE international conference on big data (big data). IEEE. pp. 424–431
13. Luo N, Yu Z, Bei Z, Xu C, Jiang C, Lin L. Performance modeling for spark using svm. In: 2016 7th international conference on cloud computing and big data (CCBD). IEEE. 2016. pp. 127–131
14. Han S, Choi W, Muwafiq R, Nah Y. Impact of memory size on bigdata processing based on hadoop and spark. In: Proceedings of the international conference on research in adaptive and convergent systems. 2017. pp. 275–280
15. Verma A, Mansuri AH, Jain N. Big data management processing with Hadoop MapReduce and spark technology: a comparison. In: 2016 symposium on colossal data analysis and networking (CDAN). IEEE. 2016. pp. 1–4
16. Gao H, Yang Z, Bhimani J, Wang T, Wang J, Sheng B, Mi N. AutoPath: harnessing parallel execution paths for efficient resource allocation in multi-stage big data frameworks. In 2017 26th international conference on computer communication and networks (ICCCN). IEEE. 2017. pp. 1–9
17. Wakde A, Shende P, Waydande S, Uttarwar S, Deshmukh G. Comparative analysis of hadoop tools and spark technology. In:

2018 fourth international conference on computing communication control and automation (ICCUBEA). 2018. IEEE. pp. 1–4
18. Lee J, Bryan NJ, Salamon J, Jin Z, Nam J. Disentangled multidimensional metric learning for music similarity. In: ICASSP 2020–2020 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE. 2020. pp. 6–10
19. Lu R, Wu K, Duan Z, Zhang C. Deep ranking: triplet MatchNet for music metric learning. In: 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE. 2017. pp. 121–125
20. Wolff D, Weyde T. Learning music similarity from relative user ratings. Inf Retr. 2014;17:109–36.
21. Chen K, Liang B, Ma X, Gu M. Learning audio embeddings with user listening data for content-based music recommendation. In: ICASSP 2021–2021 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE. 2021. pp. 3015–3019
22. Jiang C, Yang D, Chen X. Similarity learning for cover song identification using cross-similarity matrices of multi-level deep sequences. In: ICASSP 2020–2020 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE. 2020. pp. 26–30
23. West K, Cox S, Lamere P. Incorporating machine-learning into music similarity estimation. In: Proceedings of the 1st ACM workshop on audio and music computing multimedia. 2006. pp. 89–96
24. Logan B, Ellis DP, Berenzweig A. Toward evaluation techniques for music similarity. 2003
25. Wang L. Collaborative filtering recommendation of music MOOC resources based on spark architecture. Comput Intell Neurosci. 2022. https://doi.org/10.1155/2022/2117081.
26. Kumar L, Mitra A, Mittal M, Sanghvi V, Roy S, Setua SK. Music tagging and similarity analysis for recommendation system. In: Computational intelligence pattern recognition: proceedings of CIPR 2019. Springer Singapore; 2020. p. 477–85.
27. Song M, Jia L. Big data mining method of thermal power based on spark and optimization guidance. In 2018 IEEE 7th data driven control and learning systems conference (DDCLS). IEEE. 2018. pp. 514–520
28. Wang B, Qin X, Wang C, Huang W, Song Y, Cui X. A distributed exam item bank system based on hadoop ecosystem. In: 2020 IEEE 2nd international conference on computer science and educational informatization (CSEI). IEEE. 2020. pp. 9–12.
29. Ghimire S. A comparative analysis of cloud based recommendation system on mapreduce and spark (Doctoral dissertation, Pulchowk Campus). 2017
30. Ameer S, Shah MA, Khan A, Song H, Maple C, Islam SU, Asghar MN. Comparative analysis of machine learning techniques for predicting air quality in smart cities. IEEE Access. 2019;7:128325–38.
31. Fan J, Han F, Liu H. Challenges of big data analysis. Natl Sci Rev. 2014;1(2):293–314.
32. Marx V. The big challenges of big data. Nature. 2013;498(7453):255–60.
33. Bourne PE, Lorsch JR, Green ED. Perspective: sustaining the big-data ecosystem. Nature. 2015;527(7576):S16–7.
34. Frankel F, Reid R. Big data: distilling meaning from data. Nature. 2008. https://doi.org/10.1038/455030a.
35. Mattmann CA. A vision for data science. Nature. 2013;493(7433):473–5.