Feature-based User-centric Music Recommendation

Arush Srivastava

Abstract

This paper presents a novel approach to music recommendation that leverages intrinsic musical properties to address the limitations of traditional collaborative filtering methods, particularly the cold-start problem. Unlike existing systems that rely heavily on user behavior and historical data, our proposed method utilizes a neural network trained on fundamental attributes of music—such as tempo, key, mode, duration, and loudness—to generate recommendations. We detail the process of extracting these musical properties using tools like the librosa library and a custom function, followed by training a neural network to predict user preferences based solely on these features. The system's performance was evaluated using the Million Song Dataset, achieving an accuracy of 76.75%. Despite its potential, the method faces challenges related to computational efficiency and the handling of diverse musical genres. Future work includes exploring hybrid models combining collaborative filtering with feature-based recommendations and testing alternative algorithms to enhance performance and applicability.

Introduction

Modern music streaming services such as Spotify and Apple Music have come a long way in terms of music recommendation. These services predominantly use collaborative filtering and user-based filtering to recommend music. In a nutshell, collaborative filtering groups songs that are commonly listened to together by users, and recommends other songs in a group to users that listened to one song in that group. This is combined with user-based filtering, which uses a user's listening history and playlists to locate similar users and recommend music they listen to. These methods have proved effective, except for one glaring problem: a cold-start problem. If a song is new and doesn't have many listeners, it won't be recommended, no matter how good it is. This forces artists to resort to other means of marketing to grow new songs, which is difficult for small artists. This deprives small talented artists of the recognition they deserve as well as listeners of good quality music they have been missing out on. To allay this problem, a novel method of recommendation using a neural network trained on the intrinsic properties of the music a certain user listens to, such as tempo, key, mode, and loudness, may be more suitable. Other researchers in the past have set some foundation for this, notably van den Oord et al., 2013, in their paper "Deep content-based music recommendation," where they combined intrinsic properties with listening statistics in a hybrid approach. This paper aims to create and test the efficacy of a novel music recommendation system that purely views musical properties and ignores all other skewing factors. This system will follow the following steps to recommend music: extract the musical properties of every song in a database and store it away, train a neural network based on the user's listening history and the music they enjoyed, take a random sample of songs, then use the model to see which song in the sample gains the highest score and thus is worth recommending.

The Recommendation System

Step 1: The Properties of the Music

In this method of music recommendation, a new neural network will be trained based on each listener's listening history. Once the listening history is obtained, the next step is to have a way to obtain the properties of each song. I selected tempo, key, duration, mode, and loudness as my properties due to their logical effect on one's enjoyment of a song. Tempo refers to how "fast" a song is. For example, a waltz would be a traditionally "slow" song, whereas heavy metal tends to be traditionally "fast." This is measured in beats per minute (bpm). Many people have a preference as to whether they like fast or slow songs; thus, tempo is a viable factor for this recommendation network. Key and mode are music theory terms regarding the base note and style of the song. For example, a song in the key "F" uses the note F as its base note, and a song in a "minor" mode sounds melancholic. There are various modes in traditional music theory; however, for the sake of simplicity and due to the rarity of modes separate from major and minor in popular music, all music will be grouped into one of these modes. The base note and the style of the music, in the form of key and mode, can be assumed to be a factor in a user's musical enjoyment due to their stature as the keystone properties of any song in modern music theory. Lastly, loudness refers to how "loud" a song is, measured in the industry-standard measure, LUFS (Loudness Units Full Scale). People tend to have preferences as to how loud or soft their music is; for example, those who prefer heavy metal prefer louder music, whereas those who prefer jazz prefer softer music. Therefore, loudness is a key component in musical enjoyment and, consequently, the decision of whether or not to recommend a song to a user. To extract these properties, I used the librosa Python library as well as a method devised by GitHub user "jackmcarthur" in his repository titled "musical-key-finder." Using these, I created a function to deduce the mode, loudness, duration, note of key, and tempo of a song.

Step 2: The Neural Network

The key of this recommendation system is the structure of the neural network. The network should be able to take listening data and relatively accurately predict whether or not the user will like the song. To test the neural network, we require data of a listener, songs, and whether or not the listener liked the songs. In this project, the trained model with weights doesn’t matter; once a structure is found that can take these data and successfully train to recommend music, it can be trained on any user to create their own personal trained recommendation model.

```
def analyze(name):
    audio_file = librosa.load(f’{name}.wav’)
    y, sr = audio_file
    tempo, beat_frames = librosa.beat.beat_track(y=y, sr=sr)
    duration = librosa.get_duration(y=y, sr=sr)
    y_harmonic, y_percussive = librosa.effects.hpss(y)
    tonef = Tonal_Fragment(y_harmonic, sr)
    fullKey, keyConfidence = tonef.print_key()
    keyNote = fullKey[0:2].strip()
    mode = fullKey[fullKey.lower().index("m"):]
    mode = 1 if mode=="minor" else 0
    loudness = 10 * np.log10(np.vdot(y, y) / len(y))
    return [mode, loudness, duration, keyNote, tempo]
```

Figure 1. analyze function to deduce the properties of music automatically

Data Acquisition and Preprocessing

To train the neural network, I decided to use the popular open-source music dataset "Million Song Dataset" (MSD). The MSD has all of the predetermined properties of the songs, as well as a metric called “hotness,” on a scale of 0 to 1, which represents the popularity of the song in the public. To create my model, I simulated the public as the "user" to be recommended to. As such, I aimed to create a neural network that can take the data of songs and predict, in this case, whether or not the public liked the song. For the purpose of this project, I decided that a popularity metric of over 0.5 will signify that the public liked the song, and a metric of under 0.5 will signify that the public didn’t like the song. From the MSD, I selected a representative subset of 10,000 songs to train on due to computational limitations. Next, I removed all unnecessary columns from the data. The MSD has a variety of data for each song, including non-musical attributes. I removed all non-musical attributes except for the 5 predetermined in order to ensure that the music is being recommended purely based on the intrinsic properties of the music. After this stage, I removed all rows with empty or missing pieces of data. In datasets with missing or empty data, there are 2 options: fill the empty cells with some predetermined value (usually the average), or remove the rows entirely. I opted to remove the rows entirely to ensure no one property gets weighted too much due to the undercoverage of the other properties. This left 5,648 rows remaining in the set. In the next stage of preprocessing the data, I one-hot encoded the "note of key" property. The key note of a song is a categorical variable signifying the note of the key of the song out of the 17 possible notes. Enharmonic notes and keys are treated separately for the purposes of this project. I opted for one-hot encoding over label encoding due to the possible biases that come with label encoding. Label encoding consists of assigning a distinct number to each possible categorical value. However, values that have higher assigned numbers may be weighted more, skewing the model. Thus, I selected to one-hot encode the key note of the songs. Lastly, I feature-scaled all of the variables to ensure no unintended mathematical biases appear in the network due to one attribute having higher numerical values than another.

The Design of the Network

After thoroughly testing many combinations, activation functions, layer counts, and other factors of the neural network, I ended with a neural network with 3 hidden layers. Each hidden layer used a leaky ReLU activation function, with 20, 15, and 10 nodes respectively. The output layer used a sigmoid activation layer to determine whether or not the song should be recommended. The network achieved a remarkable accuracy of 76.75%. Thus, this network has the potential to be an industry-standard recommendation model.

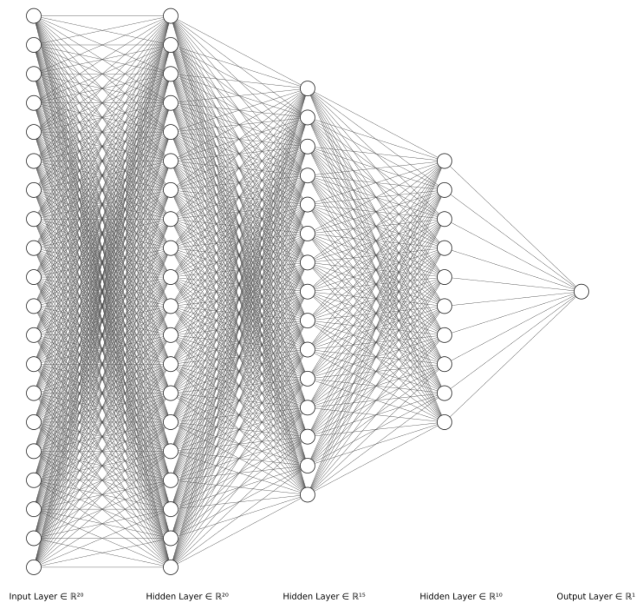
Input Layer $\in \mathbb{R}^{20}$
Hidden Layer $\in \mathbb{R}^{20}$
Hidden Layer $\in \mathbb{R}^{15}$
Hidden Layer $\in \mathbb{R}^{10}$
Output Layer $\in \mathbb{R}^{1}$

Figure 2. Neural Network Structure. Note that there is a dropout layer between the first and second hidden layers

Step 3: Recommending the song to the user

Lastly, to recommend music, the newly trained neural network must be used to recommend music to users. The model outputs a score from 0 to 1 of how likely it is that the user will like a song. In a perfect world, this model would predict the score for every song in the database, then recommend the song with the highest score and thus the highest probability of the user liking it. However, this is implausible. In most popular music databases, there are over 100 million and growing songs. Running the model on the entire database for every song recommended is simply implausible as it would take too much time, ruining the user experience. Instead, this system uses a completely random 1000-song sample of the database to run the model on and recommend a song based on the song with the highest score. The randomness of the sample ensures that every song has an equal chance of being chosen and recommended, ensuring that recommendations come completely from the intrinsic properties of the music.

Limitations

Time Taken to Analyze Music

A limitation of this method is the time it takes to analyze the music. The data in Figure 3 suggests that there is a roughly linear relationship between the number of songs being analyzed and the time spent analyzing the songs for their properties. The time itself can be reduced from roughly 14 seconds per song when using better computers, as most companies that would implement this system, such as Spotify, have vast access to such technology. My research is limited by my access to strong computing power. However, since the speed of the process does not affect the linearity of the data, the linearity of time per song will remain intact.
In order to use these properties to recommend music, all music in a database such as Spotify must be analyzed and its data then stored away. Most large services' computing access will allow them to complete this analysis in less time than 14 seconds per song. However, for the sake of this research, we will speak under the assumption that the roughly 14-second per-song analysis rate will stand for any large service that chooses to use this method. To obtain an estimate for how long it would take for a large music streaming service to gather the data needed for this recommendation system, let's use Spotify's music base as an example. Spotify boasts about 100 million songs on its platform. At 14 seconds per song, it would 1.4 billion seconds, or roughly 44 years, to analyze all the music. However, this number can be quickly whittled down by simultaneously running multiple computers at once. Using only 20 computers, a small number for a large service, would allow this entire database to be converted in about 2 years. 100 computers working at once would finish this in a little over 5 months. Thus, gaining the data for this method is plausible but will require a high short-term one-time computing investment. After this investment, every new song can be analyzed as it is uploaded, thus not requiring this process again in bulk.

Table 1. Table showing the average time taken to analyze n number of songs over 100 tests on a 2019 Macbook Air running Python 3.10.4

Number of Songs	Average Time in Seconds
1	14.778052958701737
2	27.75967443625443

Computational Power To Train the Network

Another limitation is the high computational need to train the network on each user. Neural networks take significant computational power to train, especially on large datasets such as one with over 500 songs. Furthermore, the model must be retrained every time a new song is liked by the user. As such, there is an upper limit to the number of songs that can be used as data by the system to remain within computational limits. However, enforcing this limit comes with the issue of losing old data and sacrificing accuracy, and not enforcing this limit will be implausible from a logistic standpoint. Combined with the computational need born of constant retraining, this method appears computationally implausible without sacrificing accuracy.

Random Sample May Not Include Optimal Songs

In this method, it is necessary to use a random sample of the music database to remain within the time limit for the recommendation. However, the random sample of the main database may not include any songs worth recommending to the user. In such a small sample of such a large database, it is naive to believe that the sample will indeed have songs that can even breach a score of 0.5 to indicate that they are even somewhat worth recommending. For example, for the sake of simplicity, let's discuss a user who likes exclusively hip-hop music. It is extremely likely that a 1000-song sample of such a large music base will not have a single instance of hip-hop music. Herein lies the biggest limitation of this method: on its own, the model will take too long to test every song; thus, a random sample must be used. However, using a random sample inherently destroys the accuracy of the method.

Cold-start with low listening data for the user

This method suffers from a separate cold-start problem than the one it aimed to fix. If the user is new and doesn't have much listening data, then this model fails. Neural networks require moderately high amounts of data to make accurate predictions; if a user only has 10 or 20 songs that they like it becomes difficult for the network to make an accurate prediction. On the other hand, methods like collaborative filtering can begin working from the first songs the user likes.

Inaccuracy of the analysis with non-Western or non-mainstream music

This model uses the basic music theory concepts of mode and key to recommend music. It attempts to sort every song into one of 17 key notes(separating enharmonic notes) and one of 2 modes: major or minor. However, a large amount of songs that are either non-Western or non-mainstream do not fit neatly into these buckets. For example, jazz does not have a true mode: it uses modern chords and improvisations rather than classical chords. As such, it is nearly impossible to sort them into a key as well. As such, for a user who likes jazz, this model will simply fail. Furthermore, there are more than 2 modes. In classical music theory, there are 7 modes, Although a lot of popular music can be sorted into one of the 2 modes used, there is a plethora of music that uses other modes that are far more difficult to identify automatically in a song. Lastly, modern music is straying quickly from the classical music theory concepts that are used in this model. As more artists experiment with their music, the differences between songs grow to the point where it is difficult to sort the songs through their musical properties without continually making new categories. Truly analyzing modern music trends is far beyond the current capability of machines. However, without doing so, there is a very high probability of inaccuracy when analyzing these songs and using the data to recommend new music.

Neural Network Structure May Not Be Accurate for Everyday Users

This project used data from the Million Song Dataset to design the structure of the neural network. The MSD had a similar data structure to what the network would use in a production environment; thus, it is logical to project that the neural network should be able to train and recommend successfully with real-world data. However, this cannot be tested yet as it is past the limitations of this project to acquire such data from real users. Thus, without proper experimentation, it would be hasty to assume that the projection made earlier will be true. In the case of human error in building this project where a factor has been left unaccounted for, this structure may not be able to be generalized and fall short in a production environment.

Time Taken To Train Network

This method of recommendation requires the network to be trained on the listening history of the user every time the user adds a song. However, this is a relatively minor limitation given that the number of songs trained remains under a generous limit. The data in Figures 4 and 5 suggests that there is a moderately strong linear relationship between the number of songs and the average time to train with a 0.9314 r-squared value. The best-fit slope suggests a mere 0.3150 increase in time to train per 1000 songs with a batch size of 64 and 10 epochs. As the training occurs in the background, 10 seconds to train can be set as the upper limit for a reasonable amount of time for the network to train on listening history. The model won’t need to recommend immediately, and if it does, it can use the old model for the remainder of the 10 seconds.

However, even if we take the 10-second training time, according to the data in Figures 4 and 5, the model can train on 5000+(10-4.125571271)/0.314 = about 23,708 songs before reaching that time limit.

Table 2. Table showing the average time taken to train the neural network on n number of songs over 100 trials in Google Colab

Number of Songs	Average Time in Seconds
1000	2.787152027
2000	2.993417148
3000	3.25339209
4000	3.466497451
5000	4.125571271

Table 3. The first 5 training times in seconds out of 100 for n number of songs

5000 Songs	4000 Songs	3000 Songs	2000 Songs	1000 Songs
2.955354794	4.079738054	2.876206835	2.965850049	3.80871767
3.107171452	2.811119763	2.880175064	2.713614427	2.403067449
4.066576206	2.833131294	3.286390991	2.370987529	2.549905548
3.34033478	2.871938845	3.164299164	2.678126667	3.295721305
3.073019369	3.919512971	2.72118862	3.371931355	2.777508057

Listening history of songs the user liked will very rarely reach that number, if at all. As such, the time it takes to train the recommendation network is a limitation, but the limit is so large it does not pose as an issue in utilising this method in a production environment.

Future Steps

Test Other Models

Naive Bayes Classifier

In order to reduce the time and computational weight of the recommendation system, other models aside from a neural network may be better suited. Neural networks excel most in the field of accuracy at the expense of time and computational power. An alternate model that would be plausible to use here would be a Naive Bayes Classifier. These models are very time and computationally efficient, at the expense of some accuracy. This is worth testing as a good middle ground between accuracy and time, given that one is adamant about using this method of only using the intrinsic properties of music. Furthermore, Naive Bayes Classifiers can function on little data fairly well. This would solve the cold-start problem of struggling under low listening data this method suffers from. However, this would come with a significant drop in accuracy.

Continuous Learning Models

Continuous learning models represent an exciting frontier in the field of recommendation systems, offering dynamic and adaptable approaches that contrast with traditional static models. Unlike conventional methods that rely on fixed datasets, continuous learning models are designed to incrementally update and refine their understanding based on new data as it becomes available. This characteristic makes them particularly well-suited for environments with rapidly changing content, such as recommending music in this novel method. By continuously integrating new music that the user listens to with continuously retraining the entire model, these innovative models may be able to circumvent the time and computational power-related limitations that this recommendation method suffers from.
However, the application of continuous learning models is still in its nascent stages, and several challenges need to be addressed. Firstly, these models require sophisticated mechanisms to manage and integrate continuous data streams without overwhelming the system's computational resources. The real-time updating and learning processes necessitate substantial computational power and can introduce complexities related to model stability and consistency. Additionally, the technology is evolving, and current continuous learning models are often tested on limited datasets or in controlled environments, which may not fully capture the nuances of real-world applications. As such, more research is needed to refine these models, improve their scalability, and ensure their effectiveness across diverse and large-scale systems, before introducing them in such an environment.

A Hybrid Approach

Due to the limitations listed above, utilizing a recommendation system that relies solely on the intrinsic properties of music is unrealistic. However, a hybrid approach between it and collaborative filtering may be able to encapsulate the best of both approaches.

A Transition Approach

Transitioning from collaborative filtering to a feature-based music recommendation system is a promising approach to mitigating the cold-start problem while leveraging the strengths of both methods. This hybrid approach effectively addresses the limitations inherent in each individual system, providing a more robust solution for diverse user scenarios.

In this hybrid recommendation system, collaborative filtering will be initially employed to cater to new users or situations where there is insufficient listening data to allow the feature-based method to be accurate. Collaborative filtering excels in leveraging user behavior and preferences by analyzing similarities between users and their interactions with the music. This approach is beneficial in the early stages when the listening history is limited, as it can still generate recommendations based on aggregate user data and general preferences.

As the user accumulates more listening data, the system gradually transitions to the feature-based model, which relies on intrinsic musical properties as previously discussed. This model is particularly useful for users with a substantial history, as it leverages detailed music attributes to refine recommendations. The transition can be managed through a phased approach, where the system begins to weigh feature-based recommendations more heavily as the user's listening history grows, eventually relying solely on this method when sufficient data is available.

Some benefits include enhanced personalization. Initially, collaborative filtering will help provide relevant recommendations based on similar user behavior, which is beneficial for new users. As more data is gathered, the feature-based system can deliver highly personalized recommendations by focusing on the unique attributes of the music the user enjoys. As mentioned, this approach also addresses the cold-start problem faced by the feature-based method effectively. Collaborative filtering provides a starting point for new users or new content, while the feature-based model takes over as more data becomes available, ensuring that recommendations are based on user preferences and musical attributes. To recap, collaborative filtering also experiences a cold-start problem with its inability to recommend songs with low or no listening data, since it relies on songs' listening data to find similarities between users and recommend music to them. With songs that have little to no listening data, the model ignores them. The feature-based method introduced here aimed to solve this problem. Thus, this transition approach solves both method's cold start problems as it uses both methods.

However, this transition approach comes with other disadvantages. For one, it is incredibly complex to implement. Implementing a hybrid system requires careful integration of both collaborative filtering and feature-based models. Managing the transition between these methods can be technically challenging and may involve complex algorithms to balance and merge the outputs of both models effectively. Plus, there may be a significant computational overhead. Initially, collaborative filtering requires substantial computational resources to analyze user interactions and similarities. As the system transitions to the feature-based model, additional computational resources are needed to process, analyze, and train on musical properties, potentially increasing the overall computational load. Furthermore, Maintaining and updating the system to smoothly transition between models can be complex. The system must track user data effectively and ensure that the transition from collaborative filtering to feature-based recommendations does not disrupt the user experience or degrade recommendation quality.

Transitioning from collaborative filtering to a feature-based recommendation system offers a nuanced solution to the cold-start problem, combining the strengths of both methods. While the approach provides enhanced personalization and mitigates cold-start issues, it also introduces complexities in implementation and management. Continued research and development are essential to optimize this hybrid model and address potential challenges.

Use Collaborative Filtering For the Random Sample

Another hybrid approach that could be used is to use collaborative filtering for the random sample that the model will be run on. Collaborative filtering would be used to create a sample of music that the user is likely to enjoy. Then, the model would be run on the sample to find the song that the user is most likely to enjoy with high accuracy. In order to ensure that the cold-start problem that occurs with collaborative filtering does not return, there will be a predetermined probability of the algorithm picking a completely random sample rather than a sample picked by collaborative filtering. This will allow all songs to get an even chance of being tested and recommended, while also ensuring more accurate recommendations through random samples selected by collaborative filtering. However, this comes with some issues. Using both the property-based recommendation method and collaborative filtering simultaneously will come with increased computational overhead as well as time loss. Furthermore, implementing both of these models in conjunction will likely come with complexities to find a balance between both models. Thus, this is a viable option for a hybrid approach as long as the implementation is done bearing the computational challenge in mind. More research is required for this to be viable.

Conclusion and Findings

In this paper, I proposed a novel solely feature-based music recommendation system aimed at addressing the cold-start problem inherent in collaborative filtering methods. My approach utilized intrinsic musical properties—tempo, key, mode, duration, and loudness—to generate recommendations, thereby circumventing issues related to insufficient listening data for new or obscure songs. However, due to limitations such as time, computational power, and accuracy, a music recommendation system that recommends purely based on intrinsic properties appears unfeasible currently.

The feature-based system demonstrated its potential to offer relevant recommendations based on the intrinsic characteristics of music. By focusing solely on attributes such as tempo and key, the system avoided the common limitations of collaborative filtering, particularly in scenarios involving new or underrepresented songs. The neural network model, designed to process these musical properties, achieved an accuracy rate of 76.75%, indicating its capability to provide meaningful recommendations based on music attributes alone. The feature-based approach also successfully mitigates the cold-start problem for new songs by relying on their musical properties rather than their listening history. This provides a significant advantage for both emerging artists and listeners seeking fresh music, as recommendations are not solely dependent on historical data.

However, the study identified several computational challenges, including the time required for analyzing musical properties and the computational resources needed for training the neural network. While analyzing a large database of songs is feasible with sufficient computing power, it remains a substantial initial investment. Additionally, the model's need for retraining with each new user interaction presents practical limitations. The current model also faces challenges with non-Western and non-mainstream music genres that do not fit neatly into the predefined musical properties. Further refinement of the feature set to accommodate a broader range of musical styles could improve the model's versatility. However, with the fast growth of modern music and the advent of new and innovative music styles by the day, this becomes difficult and illogical.

As opposed to a purely property-based recommendation system, the research highlighted the potential benefits of hybrid recommendation systems that combine collaborative filtering with feature-based methods. Specifically, transitioning from collaborative filtering to a feature-based model as user data accumulates offers a balanced solution to both cold-start issues and recommendation accuracy. Future research should explore the development and optimization of such hybrid approaches, as well as alternative models like Naive Bayes Classifiers and continuous learning systems, to further enhance recommendation performance while managing computational efficiency. However, implementing a hybrid recommendation system or integrating collaborative filtering with the feature-based approach introduces additional complexity and computational overhead. Effective strategies for managing these complexities and optimizing performance are essential for practical deployment.

In conclusion, this research contributes to the advancement of music recommendation systems by demonstrating the advantages and disadvantages of a feature-based approach. While there are notable limitations and challenges, the findings provide a solid foundation for future exploration and development of more sophisticated and adaptive recommendation models.

References

[1] B. Ma, T. Greer, M. Sachs, A. Habibi, J. Kaplan, and S. Narayanan, "Predicting Human-Reported Enjoyment Responses in Happy and Sad Music," 2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII), Cambridge, UK, 2019, pp. 607-613, doi: 10.1109/ACII.2019.8925463.

[2] Aaron van den Oord, Sander Dieleman, Benjamin Schrauwen. Deep content-based music recommendation. NeurIPS
Boenn, G., Brain, M., De Vos, M., & Ffitch, J. (2021). Computational Music Theory. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 8(4), 27-34. https://doi.org/10.1609/aiide.v8i4.12559

[3] Burgoyne, J.A., Wild, J., Fujinaga, I. (2013). Compositional Data Analysis of Harmonic Structures in Popular Music. In: Yust, J., Wild, J., Burgoyne, J.A. (eds) Mathematics and Computation in Music. MCM 2013. Lecture Notes in Computer Science(), vol 7937. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-39357-0_4

[4] North, A. C., Krause, A. E., Sheridan, L. P., & Ritchie, D. (2020). Comparison of popular music in the United States and the United Kingdom: Computerized analysis of 42,714 pieces. Psychology of Music, 48(6), 846–860. https://doi.org/10.1177/0305735619830185

[5] Kai R. Fricke, David M. Greenberg, Peter J. Rentfrow, Philipp Yorck Herzberg, Computer-based music feature analysis mirrors human perception and can be used to measure individual music preference, Journal of Research in Personality, Volume 75, 2018, Pages 94-102, ISSN 0092-6566, https://doi.org/10.1016/j.jrp.2018.06.004.