PlayMyMood: Personalized Hybrid Music Recommendation Engine based on Body Monitoring Parameters

Aditya Vinod Kumar, Pranav Kelkar, Arpit Agarwal, V.R. Badri Prasad
Computer Science Department
PES University
{ads.vinodk, pskelkar74, arpit64agarwal}@gmail.com
badriprasad@pes.edu

***Abstract*—In this paper, we aim to generate music recommendations for a user based on real-time body parameters, listening history, and the interest of similar users. We have gone with an unsupervised approach to this problem by gathering data from a Fitbit device and the user's Spotify account. We then pass the merged data to our novel hybrid recommendation system, which uses a nested clustering model to provide content-based and user-based collaborative filtering. Each user's listening history is recorded in their library, along with songs listened to by similar users. Their listening data is also stored in their clustering model with timestamp, heart rate, and song metadata in that hierarchy. Our paper brings together an approach that tries to model the behavior of a music taste of a user beyond the conventional music recommendation systems.**

I. Introduction

In the last few years, music streaming services have changed the industry. Each streaming service boasts several curated playlists based on genres and moods. Some of these playlists are created based on the content and are not personalised for a user. But the increase in popularity and demand of these services have led them to create innovative algorithms that can generate playlists based on a user's listening history. These work by considering the user's liked songs, favourite artists, and most played/recently played music. However, we realised that these had two problems. Firstly, these playlists do not change in real-time and are updated only every week. Second, from research, we realised that listening history is not sufficient [1] for recommendations and that a user's current activity state acts as a great auxiliary input for generating recommendations. One source that can help gather this activity information is smart devices, particularly smartwatches. Fitbit smartwatch has a populous user base, and they have heart rate sensors which can be a pretty good indication of a user's mental and activity state. With this paper, we aim to integrate Fitbit's heart rate sensor and Spotify's music streaming service and devise an algorithm that can recommend music based on the user's current state. Our core purpose is not to decipher the activity of the user but to recommend the best music after finding patterns in the user's music listening taste as well as the time and activity during the listening time.

II. Previous Work

The idea of building music recommendation engines that use body monitoring data isn't new and has been published several times. Ivana Andjelkovic *et al* [1] made a hybrid music recommendation system that integrates content and mood-based filtering in an interactive visual interface. It precomputes artist profiles based on audio and mood similarity of their tracks. The user provides as input an initial list of favoured artists using which the songs of artists nearest to the input artists' centroid are found. Xinxi Wang *et al* [2] described the use of the rich sensory data existing in mobile phones for providing short term recommendations using supervised probabilistic modelling. Starting with a set of labelled activities, the algorithm first classifies the activity and uses a probabilistic approach to find the song best matched for the activity. We draw inspiration from the idea of dividing the test users into different groups, performing blind tests and computing the quality of recommendations from the average grade provided by the test users. H. Kim *et al* [3] built a neural network that classifies the activity from the accelerometer and recommends songs from a pool of songs that are mapped to the activity type.

Several papers have outlined the use of smartwatch and mobile phones for understanding the mood and body characteristics of the user [4,5,6]. We take inspiration from these papers and build our system with the idea that body characteristics which we perceive through sensory analysis provide context for the user's mood. Since the total pool of songs is very large and assuming many users would have listened to just a small portion of the existing songs, we do not use the approach of item-to-item based collaborative filtering [7,8]. Instead, we understand that similar users can be found based on the genre of music the users listen to. We also understand different clustering algorithms and find that DBSCAN is the best algorithm to use for our requirements since there is no requirement for pre-defining the number of clusters [9,10].

The approaches mentioned above assume that there are a fixed set of activities or require manual input from the user for their mood preference. Our design approach was to have

the most seamless experience for the user without requiring any manual input. This helps a new user easily onboard to our platform. We have also decided to not use a supervised approach and instead build a personalized unsupervised system.

III. Problem Formulation

We have identified that users generally prefer to listen to songs based on their current activity and mood. Currently, the only way a user can achieve this is by manually choosing songs from a predefined list of songs in a playlist according to their mood and hit play. We want to make this whole process hassle-free and provide a seamless user experience by detecting a change in activities and moods. We pose this problem as aiming to draw a parallel between the activity and the music listening patterns of people. Provided that activity and body parameters have a strong correlation, we presume that the user's current body parameter data is a good fingerprint of the user's current activity. We aim to build a queue of song recommendations that are computed using the current body parameter data, providing a more personalized experience to an avid listener of music.

The problem involves understanding the music listening patterns of people and how we can correlate the listening hours, body monitoring parameters and song metadata into a single model. To put it in logical terms, the problem can be stated as follows - how can we recommend the best music for that particular time, the user's current activity and the user's current listening mood?

IV. Methodology

A. Understanding Personalisation

- The heart of the personalisation model is a user's music library which consists of all the songs that will be used for recommendation. There are 3 ways in which songs can be part of a particular user's music library -
  - When a user signs up, the user's most frequently played tracks from Spotify are added.
  - Every new song that the user listens to during the app's running time is added.
  - New songs that are being listened to by similar users are added with a certain probability.
- There are 3 types of input data that our recommendation engine consumes - current timestamp, current heart rate and current song's metadata vector. Our model assumes that users generally listen to music during specific intervals of the day and different patterns can be identified for the kind of music listened to during these times of the day. We further assume that a strong correlation exists between a certain activity performed by the user and the time of day. Since activity and heart rate have a strong correlation, we model heart rate as an indication of the intensity of the activity. Since most users have different preferences of music during different activities, we assume that song patterns are found under the user's activity.

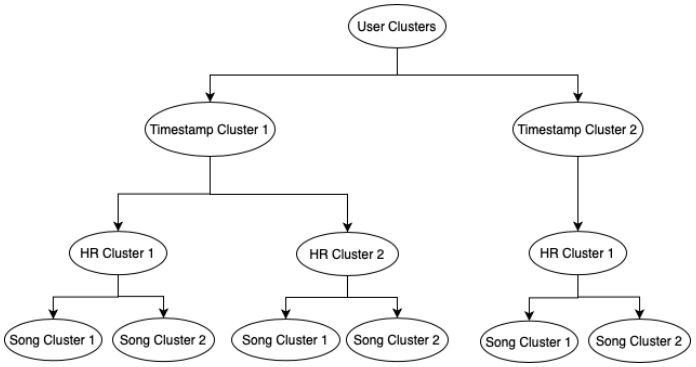

Fig. 1. Personalized Hierarchical Clusters for a User

For each user we build a nested clustering model - we model the timestamp clusters as the topmost level of the hierarchy. Each timestamp cluster has several heart rate clusters within it. Each heart rate cluster in turn has a number of song clusters within it. In this manner, we are able to infer the different music listening patterns of the user and understand their correlation with heart rate and time.

B. Personalized Hybrid Recommendation Model

Initially, when the user starts listening to music, we create the initial hierarchical clusters of the model using the user's existing data points. Each data point is a tuple of 3 values - (timestamp, heart rate, song metadata vector). The nested cluster model can be viewed as a tree data structure as seen in Fig. 1. where each node represents the cluster details such as centroid and underlying points. The immediate children of the root node are the nodes representing the different timestamp clusters. The children of the timestamp cluster nodes are the heart rate cluster nodes. The children of the heart rate cluster nodes are the song cluster nodes. We shall use this data structure to deliver Content-based recommendations for the user.

Each level of the tree consists of the clusters of a single parameter from the data points. We use DBSCAN for our clustering algorithm setting the values of minimum points and epsilon distance for the parameter.

Since the smartwatch sends data every few seconds, the server would be combining the multiple data sources into data points continuously. Since the instantaneous recommendations would be based off this data structure, the structure must always consist of all the data points of the user. Performing a re-cluster every few seconds is a very costly operation and will cause a considerable delay to return the song recommendations. To solve this problem, we propose a novel algorithm that uses the underlying concepts of DBSCAN to add the data point to the different cluster nodes of the hierarchical cluster tree without performing a full re-cluster.

In essence, the algorithm first performs a DBSCAN on the new data point and the existing outliers of the parameter clusters. If the datapoint forms a cluster along with some of the

existing outliers, those outliers would also need to be updated. It then checks if any existing cluster has at least MIN_PTS exist at a distance less than ϵ. If there exists at least one such cluster, we add all the points that are to be updated to the cluster and recursively call the same algorithm for the next parameter. If no such cluster exists depending on the count of points to update we either form a new cluster for the points or mark the data point as an outlier. This flow ensures that the least amount of operation is done while ensuring that optimal nested clusters are formed. After a certain time interval such as every hour, we do a complete re-cluster of the data. The reverse-engineered logic is explained in Algorithm 1.

The recommendation algorithm works by finding the clusters of which the current data point is a part. We check if the data point is marked as part of a cluster or as an outlier. If the point is an outlier, we provide a random selection of songs weighted by the existing cluster population. We perform the same check at every level starting with the timestamp, moving to heart rate and ending with song metadata. If the data point is a part of the cluster at every parameter level, we use the centroid of the song cluster. We compute the distance of the centroid from every other song in the user's music library. We first find the nearest songs of the same genre, followed by songs of different genres. We model the centroid as the crux of the recommendations to signify the trend of the user's listening behaviour within the particular heart rate range and timestamp range. This way a user would receive recommendations relevant to the current genre but also would pick up songs from other genres that are similar in nature. Hence our content based recommendations do not solely look for the genre or the metadata vector alone but model the user's music listening behaviour as well. The entire approach can be seen in Fig. 2.

There might be scenarios where users might listen to different kinds of music during the varied activities at a particular time interval of the day. For instance, a user might listen to songs of the pop genre during the weekday commute to work in the morning but might listen to rock songs during a workout on a weekend morning. Our model of nested clusters prioritizes the parameters such that these kinds of user behaviour are recognized and differentiated whilst providing recommendations.

Each user is represented by a vector of zeros of dimensions 1516×1. The first dimension denotes the number of genres that exist in music, hence we call this the user's genre vector. For all the genres of the currently playing track, we increment the value at the index of the particular genre. Whenever similar users listen to new tracks, they are added to the user's music library with a certain probability. To calculate the similarity between two users, we find the cosine similarity between their respective genre vectors. This can be done by storing the concatenated matrix of all the normalized genre vectors of the users and performing a matrix multiplication with the user's genre vector. This approach handles the collaborative filtering based recommendations where the songs listened to by other similar users are recommended to the current user.

Algorithm 1: addToCluster: Algorithm to add data point to existing clusters

Result: new datapoint is added to hierarchical clusters

for *every new datapoint* **do**
  $outlier\_cluster\_labels \leftarrow DBSCAN(outliers + datapoint, \epsilon, MIN\_PTS)$;
  if $outlier\_cluster\_label[datapoint] = outlier$ **then**
    $points\_to\_update \leftarrow datapoint$;
  else
    $outliers\_to\_change = outliers.filter(outlier\_cluster\_label[outlier] != "outlier")$;
    $points\_to\_update \leftarrow datapoint + outliers\_to\_change$;
  end
  $near\_existing\_cluster \leftarrow$ False;
  $cluster\_index \leftarrow 0$;
  while $cluster\_index < count(clusters)$ ***and*** $near\_existing\_cluster = False$ **do**
    $point\_count \leftarrow 0$;
    for *every cluster_point* **do**
      if $distance(cluster\_point, datapoint) < \epsilon$ **then**
        $point\_count \leftarrow point\_count + 1$;
      end
    end
    if $point\_count >= MIN\_PTS$ **then**
      $near\_existing\_cluster = True$;
    end
    $cluster\_index \leftarrow cluster\_index + 1$;
  end
  if $near\_existing\_cluster = True$ **then**
    $addPointsToClusterLevel(points\_to\_update)$;
    $datapoint[parameter][cluster\_index] = cluster\_index - 1$;
    $addToCluster(next\_parameter)$;
  else
    if $count(points\_to\_update) = 1$ **then**
      $datapoint[parameter][cluster\_index] =$ "$outlier$";
    else
      $createNewParameterCluster(points\_to\_update, parameter)$;
      $datapoint[parameter][cluster\_index] = count(clusters) + 1$;
      $makeNewClusters(points\_to\_update, next\_parameter)$;
    end
  end
end

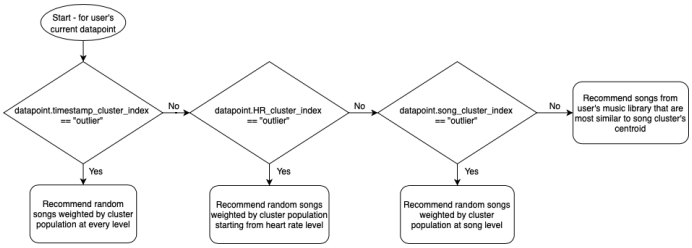

Fig. 2. Content-Based Recommendations Algorithm

C. Implementation Modules

Our system mainly consists of 3 modules

- Fitbit App - The Fitbit companion app on the user's Android phone communicates with the user's Fitbit smartwatch and sends body parameters to the server periodically. Currently, the user's heart rate is being sent as the body parameter.
- Android App - The Android application is created to communicate with Spotify. It is responsible for collecting currently playing songs and sending the information to the server whilst also adding accessing the recommendations provided by the server and adding them to the user's music queue.
- Central Server - The web server collates the incoming heart rate and currently playing song metadata along with the timestamp. It then feeds the data point to the personalized hybrid recommendation model. When the Android app requests for recommendations, it runs the recommendation algorithm that picks the best songs for the user's latest data point.

V. Training and Testing

Since the system involves personalized recommendations for a user, the general perception of the recommendations can be relative to the music taste of the user. Some users might like recommendations that are solely based on the genre while some users might like recommendations that consist of a similar music tempo as the current songs. For our fine-tuning and testing, we take the approach of asking the test users to perform a blind test of the recommendations provided by the server for a specific period. We perform the same testing routine multiple times with different parameters without letting the users know about the changes made.

During this testing routine, the users wore a Fitbit watch and listened to music through the Spotify app with our Android app running in the background. The users were asked to model their normal music routine with these devices, generally performing a range of activities during different times of the day and varying their genre of music to listen to. They were also given complete freedom to either play our recommendations or remove them from the music queue. After the training period, each user was asked to rate their recommendations on a scale of 1 to 5, called the Mean Opinion Score (MOS). Whichever parameters had the highest MOS was chosen as the final parameters.

We further requested the test users to test the recommendations without the heart rate as a parameter, using just the timestamp and song metadata as parameters. A higher grade for the recommendations with heart rate as a parameter would go to show that the idea of modelling the body parameter data is crucial to enhancing the music recommendations of the user.

VI. Results

We can see the average ratings given by 3 test users in Fig. 3. P3 got the highest average rating of 3.33 with (timestamp ϵ, heart rate ϵ, song metadata ϵ) values (3600, 10, 0.5). All the observations had the MIN_PTS set as 5.

Parameter Name	ϵ Parameter Values	Average Score
P1	(5400, 20, 0.1)	2.5
P2	(5400, 15, 0.05)	1.5
P3	(3600, 10, 0.5)	3.33
P4	(7200, 20, 0.02)	1.33

Our test users strongly felt that there is a need for a system that models their music around their daily activities and moods. They were strongly interested in making a music system that models the timestamp, heart rate and song types a part of their daily lives, affirming our analysis that there is a need for such advanced music systems. We have successfully modelled activity through the heart rate values for the Spotify track recommendations, something not available by using the standalone app previously.

Users felt that when they started playing songs from a new genre, the recommendations provided were not always similar. We rationalize the problem in the following manner - The data points would first be treated as outliers and hence recommendations would be randomly chosen weighted by cluster population. Once a new cluster is made with these data points, the centroid would be used to find the nearest songs from the music library. If there are no songs of this genre from the music library, the nearest songs are found using cosine similarity on the metadata vector of the songs. Therefore, it goes to show that the metadata vector provided by Spotify's Audio Features API does not always have a strong correlation with genre even if certain attributes such as tempo, bass, etc match.

Users felt that for the same genre the recommendations were being added from the same list of 10-12 songs, with little variation beyond that. We expected this problem as we weren't taking into consideration the number of times the same song was being recommended. Hence the same song might be recommended since it is of the same genre as the current track and is more similar than the others.

VII. Future Work

Currently, the recommendations are fetched for every song without taking the sequence of plays into consideration. Making use of user feedback and session listening history would be an important next step to improving the experience of using our app [3]. User feedback can be the user's behaviour in the app, such as the time passed from the song's start, the

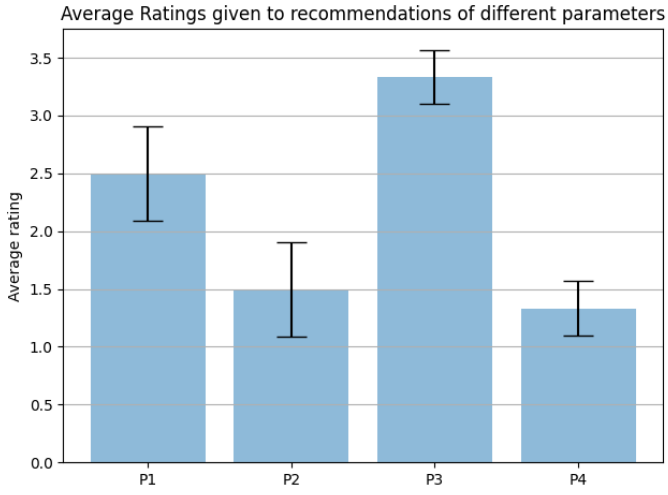

Fig. 3. Average Ratings for different Parameter based Recommendations

number of times the recommended songs were played, etc. The behavioural parameters of the user can be used to vary the parameters such as ϵ of the model.

There might be scenarios where the user might receive the same recommendations for several songs. One of the improvements to the model can be to use the number of times a song is played as a measure of the user's interest in the song. Another parameter that discounts the frequency would be the number of times a song is recommended without it being played.

As of now, each song played by the user leads to 1 to 3 recommendations added to the queue. Instead, there can be a queue with a fixed set of songs that is updated with the user's track change.

One important improvement that can be done is to move the cluster modelling and other personalized computation to the mobile device. This would substantially reduce the load on the server and would reduce any delays in retrieving recommendations.

REFERENCES

[1] Andjelkovic, Ivana & Parra, Denis & O'Donovan, John. (2016). Moodplay: Interactive Mood-based Music Discovery and Recommendation. 10.1145/2930238.2930280.

[2] Xinxi Wang, David Rosenblum, and Ye Wang. 2012. Context-aware mobile music recommendation for daily activities. In Proceedings of the 20th ACM international conference on Multimedia (MM '12). Association for Computing Machinery, New York, NY, USA, 99–108. DOI:https://doi.org/10.1145/2393347.2393368

[3] H. Kim, G. Y. Kim and J. Y. Kim, "Music Recommendation System Using Human Activity Recognition From Accelerometer Data," in IEEE Transactions on Consumer Electronics, vol. 65, no. 3, pp. 349-358, Aug. 2019, doi: 10.1109/TCE.2019.2924177.

[4] Ayata, Deger & Yaslan, Yusuf & Kamasak, Mustafa. (2018). Emotion Based Music Recommendation System Using Wearable Physiological Sensors. IEEE Transactions on Consumer Electronics. PP. 1-1. 10.1109/TCE.2018.2844736.

[5] Han, Byeong-jun & Rho, Seungmin & Sanghoon, & Hwang, Eenjun & Han, B.-J & Jun, Sanghoon. (2010). Music emotion classification and context-based music recommendation. Multimed Tools Appl. 47. 433-460. 10.1007/s11042-009-0332-6.

[6] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet, "Recognizing Detailed Human Context In-the-Wild from Smartphones and Smartwatches" in IEEE Pervasive Computing magazine

[7] P. Resnick and H. R. Varian, "Recommender systems," Commun. ACM, vol. 40, no. 3, pp. 56–58, 1997.

[8] J. Y. G Linden, B Smith, "Amazon.com recommendations: item-to-item collaborative filtering," in IEEE, Internet Computing, vol. 7, 2003, pp. 76–80.

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96). AAAI Press, 226–231.

[10] Xu, D., Tian, Y. A Comprehensive Survey of Clustering Algorithms. Ann. Data. Sci. 2, 165–193 (2015). https://doi.org/10.1007/s40745-015-0040-1

[11] J. Oyelade et al., "Data Clustering: Algorithms and Its Applications," 2019 19th International Conference on Computational Science and Its Applications (ICCSA), 2019, pp. 71-81, doi: 10.1109/ICCSA.2019.000-1.

[12] Sharma, Arvind & Gupta, Rajendra & Tiwari, Akhilesh. (2016). Improved Density Based Spatial Clustering of Applications of Noise Clustering Algorithm for Knowledge Discovery in Spatial Data. Mathematical Problems in Engineering. 2016. 1-9. 10.1155/2016/1564516.

[13] Nijkamp, R.. "Prediction of product success: explaining song popularity by audio features from Spotify data." (2018).

[14] T. Hornung, C. Ziegler, S. Franz, M. Przyjaciel-Zablocki, A. Schätzle and G. Lausen, "Evaluating Hybrid Music Recommender Systems," 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013, pp. 57-64, doi: 10.1109/WI-IAT.2013.9.

[15] X. Li and T. Murata, "Multidimensional clustering based collaborative filtering approach for diversified recommendation," 2012 7th International Conference on Computer Science Education (ICCSE), 2012, pp. 905-910, doi: 10.1109/ICCSE.2012.6295214.

[16] H.P. Kriegel, P. Kroger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," ACM Transactions on Knowledge Discovery from Data, vol. 3, no. 1, 2009.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," ACM Computing Surveys, vol. 31, no. 3, pp.264–323, 1999.