A Personalized Music Recommender Service Based on Fuzzy Inference System

Md. Saidur Rahman, Md. Saifur Rahman, Shahnewaz Ul Islam Chowdhury, Ashfaq Mahmood, Rashedur M. Rahman
Department of Electrical and Computer Engineering,
North South University
Plot -15, Block -B, Bashundhara, Dhaka 1229, Bangladesh
saidurrahman@northsouth.edu, breekal7@gmail.com, muradchowdhury0215@live.com, ashfaq.bd92@gmail.com,
rashedur.rahman@northsouth.edu

Abstract—**In this paper, we are proposing a personalized music recommender service based on Mamdani Fuzzy Interference System (M-FIS). Collection of playlist is used for gathering users' choice and mood while listening to songs. Similarity between audio files is calculated based on Mel Frequency Cepstral Coefficients (MFCC). We have developed a recommender model based on M-FIS with the aforementioned similarities and playlists. We were able to gain an acceptable accuracy rate using FIS compared to other method reported in literature.**

Keywords-component; Music Recommender Service; Fuzzy Inference System(FIS); Mel Frequency Cepstral Coefficients (MFCC).

I. INTRODUCTION

In this modern era people have been much aware of what they listen to. And they also tend to listen to music to be similar to what they have listened to previously. This makes people look for new music every day and thus recommendation systems come in handy. Recommendation systems generally try to find a pattern in people's listening habits.

We are proposing a system that combines "metadata pattern matching" and "analysis of the music tracks" to reach a more accurate point in recommending a song that a person may want to listen to. Similarities among the songs are calculated based on Mel Frequency Cepstral Coefficients (MFCCs) value of the songs and the songs appearances' on the playlists in the sample dataset. The values for the later one is calculated using weighted average value for each song to another song based on the number of appearances and the order they appeared on the playlists. Using the similarity values of MFCC and "appearance on playlists", we have built a Mamdani Fuzzy Inference System which gives us a weighted matrix which represents numerical distances among songs. The lower the distance between two songs is, the more similar they are. For recommending music, we use the sorted output matrix of that FIS for any given song.

Section-II briefly describes related works. Section-III presents system design. Distance calculation is depicted in Section-IV. Section-V implements M-FIS with different membership functions. Section-VI shows result analysis. Finally, Section-VII concludes the paper.

II. RELATED WORKS

Kim et al [1] built recommendation model based on weighted clusters and K-Means algorithm on the values of calculated Hidden Markov Models (HMM) and MFCC considering every sample songs. Aucouturier and Patchet [2] measured similarity based on timbre and used Gaussian Mixed Model (GMM) on those for recommending music. Siddiquee et al [3] recommended songs using rules generated by Association Rule Mining (ARM) from users' playlists and similarity scores based on MFCC of the songs on the sample dataset. Sen and Martha [4] recommended songs based on users' mood detected through a number of Fuzzy Logic models with users' contextual sensor (GPS, Compass etc.) information fused with information from the web. Hoffman et al.[5] on their paper experimented with feature vectors on low-level signal descriptors of music files and used correlation analysis and Principal Component Analysis to optimize them for the music recommendation system.

III. SYSTEM DESIGN

We propose a music recommender system using a Fuzzy Inference System (FIS) based on similarities amongst the audio files from a defined set. Two different approaches are being used for finding mathematical distances amongst the audio files. In one of our approach, we will use MFCC values for comparing audio tracks with one another. A square matrix with respect to number of unique audio tracks will be generated as distances amongst the tracks using that method. We have developed a simple algorithm for creating a distance matrix based on the weight - each of them carries information due to their appearances in the playlist. This algorithm is well described in Section-IV(B) – "Playlist Based Distance Calculation". This approach will give us another square matrix as distance matrix. The resultant distances of those approaches will be used as inputs for the Mamdani type Fuzzy Inference System (M-FIS). For any given music file, our recommender system will use the output from the M-FIS, sorted in ascending order, for suggesting music(s) to the user. Figure-1 shows the architecture of the system.

In the following sections, distance calculation approaches, M-FIS implementations and music suggestion technique will be described with examples.

978-1-5090-0806-3/16/$31.00 copyright 2016 IEEE
ICIS 2016, June 26-29, 2016, Okayama, Japan

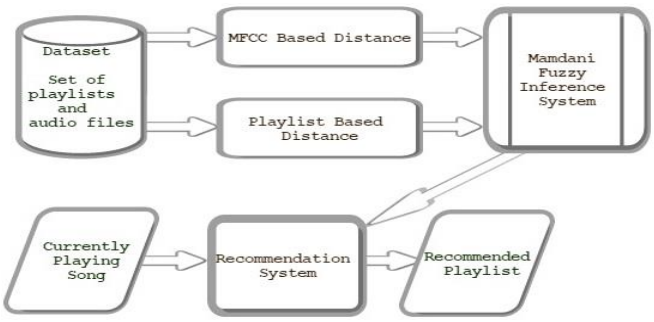

Figure 1. Architecture of the Recommender System

IV. DISTANCE CALCULATION

A. MFCC Based Distance Calculation

We are following the steps, suggested by Pampalk [6], for MFCC based distance calculation. We are using a matlab toolbox – 'ma', developed by Pampalk [7], for calculating distance amongst the audio files. This approach needs two steps – "Extracting Features" and "Feature Based Distance".

1) Extracting Features

From every single audio file, following four features have been extracted – "Single Gaussian values (G1) of MFCCs", "Fluctuation Pattern" (FP), "Fluctuation Pattern Bass" (FPB) and "Fluctuation Pattern Gravity" (FPG). All these four features get calculated based on the MFCC.

- MFCC Calculation:

We cut out 23ms segments of the signal of the audio file. Using Fast Fourier Transform, power spectrum of the signal gets calculated. Using filter bank, consisting of triangular filters, Mel-scale (Stevens, Volkmann & Newman, 1937) is derived from the power spectrum. Mel-scale can be defined as Equation 1.

$$m_{Mel} = 1127.01048 * \log\left(1 + \frac{f_{Hz}}{700}\right) \quad (1)$$

Response of a frequency band is defined by each triangular filter. Later these filters are normalized in such way that the frequency band is twice as high as each triangle. The Mel Frequency (M) gets calculated using the filter matrix and power spectrum. A logarithmic ratio scale, Decibel (dB), is used for the perception of loudness approximation. 1 dB is the reference value. With respect to the reference value, all audio signals are rescaled. Discrete Cosine Transformation is used for compressing the Mel power spectrum and stored in a matrix (DCT) of which each row is basically an eigenvector. Finally, MFCC calculation is done by following Equation 2.

$$mfcc = DCT * \log(10 * \log(M)) \quad (2)$$

- Features

1. Single Gaussian values (G1) of MFCCs: Using MFCCs, the mean, covariance and inverse covariance of them are extracted and modeled in a Gaussian model proposed by Mandel [8].
2. Fluctuation Pattern (FP): The attributes of the audio signal is described by the Fluctuation Pattern (FP) . FPs are used for describing oscillation variation of the loudness for each frequency band.
3. Fluctuation Pattern Bass (FPB): The total of the values in the two lowest frequency bands with a variation frequency higher than 1Hz is called Fluctuation Pattern Bass (FPB).
4. Fluctuation Pattern Gravity (FPG): The gravity center of FP on the frequency variation axis is called Fluctuation Pattern Gravity (FPG).

2) Feature Based Distance

For calculating distance between two audio files, we first calculate distance between each feature and then merge them with certain weight assigned to them. Euclidean distance is used for fluctuation pattern. The absolute subtraction is taken as distance for Fluctuation Pattern Bass and Fluctuation Pattern Gravity. Using the symmetric Kullback-Leibler [9] divergence, distance between Gaussian models has been calculated. A simple and straightforward normalized distance equation is used by Pampalk [6] and shown as Equation 3.

$$d = 0.7 * \frac{-\exp\left(\frac{d_{g1}}{450}\right)+0.7950}{0.1493} + 0.1 * \frac{d_{FP}-1688.4}{878.23} + 0.1 * \frac{d_{FPG}-1.2745}{1.1245} + 0.1 * \frac{d_{FPB}-1064.8}{932.79} \quad (3)$$

The weight has been found by Pampalk [6]. The difference amongst the weight refers to the contribution of that feature in similarity measurement. The distances amongst all the audio tracks are represented in a matrix. Finally we set the value of each song against itself with maximum value of the distance matrix assuming it is least likely to hear the same track with immediate repetition.

B. Playlist Based Distance

"Playlist Based Distance" (PBD) is calculated based on the audio files appearances on the playlists and in which order they appeared. These distances will be finally represented as another matrix. We are using the following algorithm for generating that matrix. The pseudo code of this algorithm is shown in Figure-2.

Step-1: Assuming we have N number of unique audio files in the playlist. Create a NxN PBD matrix and an NxN "counter" matrix initialized with zero. The "counter" for tracing the number of time a song has been encountered while measuring weight against its predecessor song in the playlist.

Step-2: Take the first playlist from the playlist set.

Step-3: Assign each song of the playlist a positional value starting from 0 in algebraically incremental rate. Put a song marker on the first song of the current playlist. Calculate the scoring value for each of those songs using Equation 4,

$$s = \frac{i}{m(m+1)/2} \quad (4)$$

Here, s is the scoring value of that song, m is the positional value of the last song in the current playlist and i is the zero based index for the current song set in Step-4. Scoring value refers to the possibility of user hearing that song after the immediate predecessor song depending on where the song duos are positioned in that playlist.

Step-4: Move song marker to the next song. And take the song pointed by the marker from the current playlist.

Step-5: Calculate the weight of that song using Equation 5,

$$w = \frac{v*t + c}{t+1} \quad (5)$$

Here, *w* is the new weight, *v* is the weight of that song from the PBD matrix (initially v is 0), *c* is the current scoring value of that song calculated in Step-3 using equation 4 and *t* is the number of time it encountered in other playlists from the "counter" matrix of Step-1.

Step-6: Update PBD matrix for the current song with the new weight. The matrix will update the cell for the current song against the previously marked song and not vice versa. Increment the current value of the "counter" matrix of Step-2 for the current song against the previously marked song by 1.

Step-7: If there is no more song in the playlist, go to step-8, otherwise go to step-4.

Step-8: If there is no more playlist in the playlist set, go to step-9. Otherwise, take the next playlist from the playlist set and go to step-3.

Step-9: Multiply each value with 100. Replace all the zero values with 100.

The weight values in the PBD matrix after calculation refer to the distances amongst audio tracks. A zero value seems no distance but in practical it means it never gets calculated thus maximum distance. Hence the value should be 100. Replacing all the zero values with 100 in step-9 takes care of this situation.

To understand the procedure more clearly, let us look at an example. Considering the example dataset of Table XIII, we will compute the value of PBD matrix for song1 after song2. So, our target set is {"song2", "song1"}. In the example dataset, playlist 1 and playlist 6 has our target set. First, considering playlist 1, we set the indexes, *i*, following the above method, {"song4:1", "song2:2", "song1:3"}. So, our value for *m* is 3. Using equation 2, scoring values are calculated {"song4:0.1667", "song 2:0.33", "song1:0.5"}. For calculating the weight value we will use equation 5. Initially PBD matrix is empty, so current weight value, *v*, for all song in the playlist is zero and *t* is also zero which means w=c or w=s. Now, take playlist 6, the second occurrence of target playlist. We use the same procedure as we did earlier for calculating the scoring value {"song1:0.1667", "song4:0.33", "song3:0.5"}. But now PBD matrix has weight value for song1 against song2. So, for song1 against song2, v= 0.5, t=1 and c=0.1667. Using the equation 5, new weight value, w = 0.33. As there is no other occurrence for song1 after song2, the weight value calculation is ended. At final stage, we scale the value multiplying 100 to keep in range 0-100. Pseudocode is given in Figure 2.

```
Input: A set of playlist
Output: A square matrix containing relative distances

function PlaylistBasedDistnace()
   PBD[][], Counter[][], Scores[]
   NoOfPlaylists = size of the given set of playlists, SetOfPlaylist
   N = number of unique songs in the SetOfPlaylist
   i = 0, j = 0
   while (i<N)
      while(j<N)
         PBD[i][j] = 0
         Counter[i][j] = 0
         j = j + 1
      end
      i = i + 1
   end
   PlaylistMarker = 0
   while(PlaylistMarker < NoOfPlaylists)
      m = number of songs in the SetOfPlaylist{PlaylistMarker} - 1
      k = 0;
      while (k < m+1)
         Scores[k] = i /((m*(m+1))/2)
      end
      marker = 0
      do
         marker = marker + 1
         v = PBD[marker][marker+1]
         t = Counter[marker][marker+1]
         c = Scores[marker]
         PBD[marker][marker+1] = (v * t + c)/(t+1)
         Counter[marker][marker+1] = t + 1
      while (marker < m)
      Empty Scores[]
      PlaylistMarker PlaylistMarker + 1
   end
   while (i<N)
      while(j<N)
         PBD[i][j] = PBD[i][j] * 100
         if(PBD[i][j] == 0)
            PBD[i][j] = 100
         end
         j = j + 1
      end
      i = i + 1
   end
return PBD as output
```

Figure 2. Pseudo-code for calculating Playlist Based Destance

V. IMPLEMENTING M-FIS WITH DIFFERENT MEMBERSHIP FUNCTIONS

We have used three different types of membership functions to implement our FIS. The FIS composition that was used is as follows. For min operation AND was used. OR is used for max operation. Aggregation operation uses max operator. Implication uses min operator and defuzzification was implemented by centroid method.

Now we introduce the membership function for input and output. The inputs for the system are "MFCC Distance" and "Playlist Distance". Each of their value is from the set {low, medium, high}. The output membership functions have three sets of values: {close, average, far}, {very close, close, average, far} and {very close, close, average, far, very far}. With these values, we have implemented "Triangular", "Gaussian" and "Trapezoidal" output membership function. So, there will be in total 9 FIS models. Tables I, Tables II and Tables III illustrate the rule sets for 3, 4 and 5 output membership functions.

TABLE I. FOR 3 OUTPUT FUZZY VALUE(CLOSE, AVERAGE, FAR)

MFCC \ playlist distance(weighted)	low	medium	high
low	close	close	average
medium	close	average	far
high	average	far	far

TABLE II. FOR 4 FUZZY OUTPUT VALUE (VERY CLOSE, CLOSE, AVERAGE, FAR)

MFCC \ playlist distance(weighted)	low	medium	high
low	very close	close	average
medium	close	average	far
high	average	far	far

TABLE III. FOR 5 FUZZY OUTPUT VALUE (VERY CLOSE, CLOSE, AVERAGE, FAR, VERY FAR)

MFCC \ playlist distance(weighted)	low	medium	high
low	very close	close	average
medium	close	average	far
high	average	far	very far

The details of "Triangular" and "Trapezoidal" membership functions and different input-output parameters are given in the following subsections.

A. Triangular Membership Functions

The generalized form of triangular membership function - TriMF is given below.

$$\text{TriMF}(x:a,b,c) = \begin{cases} 0, if(x \le a) \\ \frac{x-a}{b-a}, if(a \le x \le b) \\ \frac{c-x}{c-b}, if(b \le x \le c) \\ 0, if(x \ge c) \end{cases} \quad (6)$$

- Input

Table-IV enlists the values of the parameters (a, b and c) for creating triangular membership function for the fuzzy value of MFCC and Playlist based distance using Equation-6.

TABLE IV. PARAMETER VALUE OF TRIMF FOR MFCC AND PLALIST BASED DISTANCE

Distance	a	b	c
MFCC$\mu_{low}(x)$|Playlist$\mu_{low}(x)$	0|0	0|0	7.5|50
MFCC$\mu_{medium}(x)$|Playlist$\mu_{medium}(x)$	0|0	7.5|50	15|100
MFCC$\mu_{high}(x)$|Playlist$\mu_{high}(x)$	7.5|50	15|100	15|100

- Output

Table-V, Table-VI and Table-VII show the values of the parameters (a, b and c) for creating triangular membership function for the Outputs' fuzzy values using Equation-6.

TABLE V. PARAMETER VALUE OF TRIMF FOR 3 FUZZY OUTPUT VALUE

Output (for 3 fuzzy output value)	a	b	c
μclose(x)	0	0	0.5
μaverage(x)	0	0.5	1
μfar (x)	0.5	1	1

TABLE VI. PARAMETER VALUE OF TRIMF FOR 4 FUZZY OUTPUT VALUE

Output (for 4 fuzzy output value)	a	b	c
$\mu_{very\ close}(x)$	0	0	0.333
$\mu_{close}(x)$	0	0.333	0.667
$\mu_{average}(x)$	0.333	0.667	1
$\mu_{far}(x)$	0.667	1	1

TABLE VII. PARAMETER VALUE OF TRIMF FOR 5 FUZZY OUTPUT VALUE

Output (for 5 fuzzy output value)	a	b	c
$\mu_{very\ close}(x)$	0	0	0.25
$\mu_{close}(x)$	0	0.25	0.50
$\mu_{average}(x)$	0.25	0.50	0.75
$\mu_{far}(x)$	0.50	0.75	1
$\mu_{very\ far}(x)$	0.75	1	1

B. Trapezoidal Membership Functions

The generalized form of triangular membership function - TrapMF is given below.

$$\text{TrapMF}(x) = \begin{cases} 0, if(x \le a) \\ \frac{x-a}{b-a}, if(a \le x \le b) \\ 1, if(b \le x \le c) \\ \frac{d-x}{d-c}, (c \le x \le d) \\ 0, if(x \ge d) \end{cases} \quad (7)$$

- Input

Table-VIII enlists the values of the parameters (a, b, c and d) for creating trapezoidal membership function for the fuzzy value of MFCC and Playlist based distance using Equation-7.

TABLE VIII. PARAMETER VALUE OF TRAPMF FOR MFCC AND PLAYLIST BASED DISTANCE

Distance	a	b	c	d
MFCC$\mu_{low}(x)$|Playlist$\mu_{low}(x)$	0|0	0|0	0.75|5	6.75|45
MFCC$\mu_{medium}(x)$|Playlist$\mu_{medium}(x)$	0.75|5	6.75|45	8.25|55	14.25|95
MFCC$\mu_{high}(x)$|Playlist$\mu_{high}(x)$	8.25|55	14.25|95	15|100	15|100

- Output

Table-IX, Table-XI and Table-XII show the values of the parameters (a, b, c and d) for creating triangular membership function for the Outputs' fuzzy values using Equation-7.

TABLE IX. PARAMETER VALUE OF TRAPMF FOR 3 FUZZY OUTPUT VALUE

Output (for 3 fuzzy output value)	a	b	c	d
$\mu_{close}(x)$	0	0	0.05	0.45
$\mu_{average}(x)$	0.05	0.45	0.55	0.95
$\mu_{far}(x)$	0.55	0.95	1	1

TABLE X. PARAMETER VALUE OF TRAPMF FOR 4 FUZZY OUTPUT VALUE

Output (for 4 fuzzy output value)	a	b	c	d
$\mu_{very\ close}(x)$	0	0	0.033	0.3
$\mu_{close}(x)$	0.033	0.3	0.367	0.633
$\mu_{average}(x)$	0.367	0.633	0.7	0.967
$\mu_{far}(x)$	0.7	0.967	1	1

TABLE XI. PARAMETER VALUE OF TRAPMF FOR 5 FUZZY OUTPUT VALUE

Output (for 5 fuzzy output value)	a	b	c	d
$\mu_{very\ close}(x)$	0	0	0.025	0.225
$\mu_{close}(x)$	0.025	0.225	0.275	0.475
$\mu_{average}(x)$	0.275	0.475	0.525	0.725
$\mu_{far}(x)$	0.525	0.725	0.775	0.975
$\mu_{vrey\ far}(x)$	0.775	0.975	1	1

We have also used Gaussian membership functions in this research with different membership functions. But due to space limitation we could not provide the details of those methods.

VI. RESULT ANALYSIS

After implementing the M-FIS, we will have an NxN distance matrix. Whenever a song gets inserted into the system as an input, the recommendation service takes that matrix and fetches the row for that song. Then the system sorts the row in ascending order and delivers a playlist of songs consisting top songs from that sorted row. We first demonstrate result with some sample data set for explanation. Then we will explain our result analysis in detail on last.fm data set

A. Sample Dataset

Here we are going to showcase our proposed recommendation system with 5 dummy songs and necessary matrix values.

Table XII is an example of the MFCC distance calculated for one song against another.

TABLE XII. EXAMPLE MFCC BASED DISTANCE MATRIX

mfcc	Song1	Song2	Song3	Song4	Song5
Song1	15	10.2	9.1	12.4	8.5
Song2	10.2	15	8.9	10.1	8.1
Song3	9.1	8.9	15	13.1	7.9
Song4	12.4	10.1	13.1	15	14.1
Song5	8.5	8.1	7.9	14.1	15

Table XIII is an example of the training playlist dataset.

TABLE XIII. EXAMPLE PLAYLIST

Playlist1	Song3 Song4 Song2 Song1
Playlist2	Song5 Song3 Song1
Playlist3	Song4 Song1 Song5 Song3
Playlist4	Song2 Song4 Song5 Song3 Song1
Playlist5	Song1 Song4 Song3 Song2
Playlist6	Song2 Song1 Song4 Song5

TABLE XIV. EXAMPLE PLAYLIST BASED DISTANCE MATRIX

playlist	Song1	Song2	Song3	Song4	Song5
Song1	100	100	100	25	33.33
Song2	33.33	100	100	10	100
Song3	53.33	50	100	16.67	100
Song4	16.67	33.33	33.33	100	35
Song5	100	100	37.78	100	100

Table XIV represents the weighted average distance of a song against another song calculated by the algorithm described in methodology section above. It is noticeable that the table values are not diagonally symmetric. Because someone hears song1 after song2 that does not mean he or she hears song2 after song1. Our procedure for calculating PBD matrix takes the precedence in consideration.

TABLE XV. FIS OUTPUT MATRIX

output	Song1	Song2	Song3	Song4	Song5
Song1	0.92000	0.76706	0.75652	0.52585	0.45326
Song2	0.50494	0.92000	0.75513	0.41473	0.75118
Song3	0.56499	0.55697	0.92000	0.51635	0.75066
Song4	0.49752	0.50247	0.57284	0.92000	0.61883
Song5	0.75285	0.75118	0.44783	0.86558	0.92000

Table XV is the output distance matrix of M-FIS implementation using inputs of Table-XII and Table-XIV. Now for suggesting next songs for any particular song, *S,* we take the values of other songs of the row marked by *S* and sort those values in ascending order. Then we start recommend songs for *S* in the order of their corresponding sorted values. For example, the least value of a song against Song1 is Song5 with a value of 0.45326 and that Song5 will be recommended right after Song1 is recommended at any given point.

B. Real Life Music Dataset

The dataset[14] was originally acquired by Cornell University which contains people's song playlists and was put together by Shuo Chen from the Department of Computer Science of the university. Yes.com [10] was used to collect the playlist. Last.fm [11] was used to collect tag data. This set of data ranges within the time interval from December , 2010 to May 2011and contains about 75 thousand songs and 3 million transactions. The dataset named "yes_complete" was acquired for our use from other datasets available. Next, we collected sample audio MP3 files in 30 seconds of length using Echonest[12] and 7digital[13] API, which we later cropped to 22 seconds, mono channel and with a sampling rate of 22 KHz. However, we could obtain a total of only 1388 songs and removed all other metadata not associated with the songs acquired.

After implementing the Fuzzy Inference System (FIS), we got an output matrix of size NxN, which consists of an overall distance from a particular song to other song. The lower the distance, the higher the possibility of a song being recommended after a song has been listened to. Our system works with the number of recommended songs to be given to people and a trend follows that the accuracy of exact match increases with the number recommended songs. If the immediate next song of the users' playlist belongs to the song list recommended by our proposed system, we consider the situation as "exact match". Figure 3 (4, 5, respectively) is the graphical illustration of the possibility of a user listening to the next song as per recommended as our system when 5 (10, 15, respectively) songs are recommended.

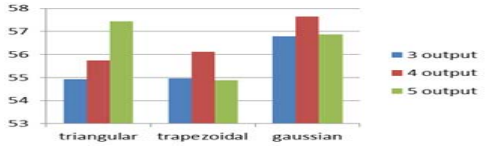

Figure 3. Accuray graph when 5 songs are recommended

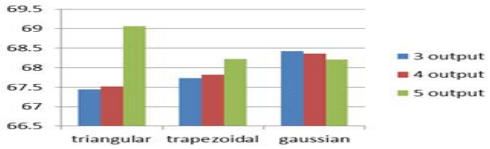

Figure 4. Accuray graph when 10 songs are recommended

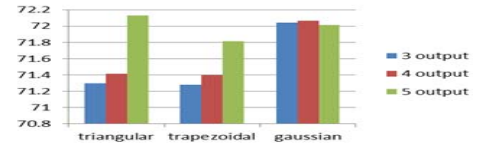

Figure 5. Accuray graph when 15 songs are recommended

From the above charts we can observe that the Gaussian membership function works well on an average scenario for all of the parameters for the recommendation system, but the Triangular membership function works better than the rest when 5 fuzzy outputs are considered to establish a Fuzzy Inference System. A comparison of the implemented Fuzzy Inference System with the previously implemented ARM by Siddiquee and et al' in their paper [3] is shown in Table-XVI. From Figure-6, we can easily observe that accuracy has increased in FIS implementation over ARM implementation.

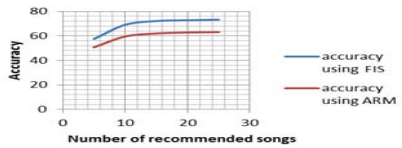

Figure 6. Comparison of the implemented FIS with ARM

TABLE XVI. COMPARISON OF ACCURACY RATE USING FIS AND ARM

Number of Recommended songs	accuracy using FIS	accuracy using ARM
5	57.43	50.70
10	69.06	59.51
15	72.13	61.99
20	72.75	62.77
25	73.28	63.06

VII. CONCLUSION AND FUTURE WORK

The Triangular membership function for 5 fuzzy output membership functions performed better than the Gaussian and Trapezoidal membership functions. The Triangular membership function was implemented in the FIS for recommending music. The accuracy percentage gradually increases as the number of recommended songs is increased. Compared to a non-fuzzy approach such as the ARM, the FIS proved to be more effective while recommending songs based on music similarity. For our future work, we are planning to use different approach to the dataset so that we can use both FIS and ANFIS.

VIII. REFERENCES

[1] Kim, K., Lee, D., Yoon, T.B., & Lee, J.H. (2008). A Music Recommendation System Based on Personal Preference Analysis, IEEE.

[2] Aucouturier,J.J., & Pachet,F.(2002). Music similarity measures: What's the use? , ISMIR.

[3] Md Mahfuzur Rahman Siddiquee and et al, A Personalized Music Discovery Service Based On Data Mining, 14th IEEE/ACIS International Conference on Computer and Information Science 2015

[4] Sen, A., & Larson, M. (2015). From Sensors to Songs: A learning-free novel music recommendation system using contextual sensor data. LocalRec'15.

[5] Hoffmann, P., Kaczmarek, A., Spaleniak, P., & Kostek, B. (2014). Music Recommendation System. Journal of telecommunications and information technology, 59-69.

[6] Elias Pampalk, "Computational Models of Music Similarity and their Application in Music Information Retrieval", MIREX'06.

[7] Elias Pampalk, "A Matlab Toolbox to Compute Music Similarity fromAudio", MIREX,2004

[8] Mandel, M. I., & Ellis, D. P. (2006). Song-level features and SVM for music classification. In International Symposium on Music Information Retrieval (ISMIR).

[9] William D. Penny, "Kullback – Leibler divergences of normal, gamma, dirichlet and wishert densities" - Technical Report , Wellcome Department of Cognitive Neurology., 2001

[10] Yes.com [online] Available from: http://yes.com[Accessed 24th January, 2015]

[11] Last.fm [online] Available from: http://www.last.fm[Accessed 24th January, 2015]

[12] Echonest.com [online] Available from: http://the.echonest.com[Accessed 24th January, 2015]

[13] 7digital.com [online] Available from: http://www.7digital.com [Accessed 24th January, 2015]

[14] Playlist Dataset [online] Available from: http://www.cs.cornell.edu/~shuochen/lme/data_page.html [Accessed 24thJanuary,2015]