Personalized Recommender System for Bollywood Music

A Project Report

submitted by

ANKIT VAIDYA

in partial fulfilment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

May 2016

THESIS CERTIFICATE

This is to certify that the thesis entitled **Personalized Recommender System for Bollywood Music**, submitted by **Ankit Vaidya**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bonafide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Hema A. Murthy
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to to express my sincere thanks and deep sense of indebtedness to my guide Dr. Hema A. Murthy for her guidance and motivation throughout my work. Her inspiring suggestions motivated me to solve problems efficiently. I am also grateful to my guide also for providing access to the DONLAB servers: Dell-d1 and Dell-d2 without which none of my experiments could have been performed.

I would thank Dr. Hema A. Murthy for teaching Pattern Recognition course, Dr. C. Chandra Sekhar for teaching Kernel Methods for Pattern Analysis course and Dr. Sayan Ranu for teaching Data Mining course which helped me a lot in my project work.

I would also thank Jom Kuriakose, Padma Sundari, Jyotsna Thapliyal, Ankita Agrawal, Akshay Gaikwad, Shreya Khare and all my colleagues in DON LAB who always helped me whenever needed in my research.

Lastly, I am thankful to god for giving me enough luck to get into IIT Madras and I am thankful to my parents for all the moral support and the amazing oppurtunities they have given me over the years.

ABSTRACT

KEYWORDS: Recommender system, MFCC, lyrics, melody, genre, Gaussian mixture model, decision tree, hidden Markov model, RLCS, bag of words

Recommender systems are being used quite often in areas related to audio, video, e-commerce, search engines etc. In this work, an attempt is made to learn specific user′s preference. It tries to learn from user′s history of choice and then recommends whether a new song will be liked by a person or not. The first attempt in this regard was to predict the genre of the music by analyzing timbre present in songs. Gaussian mixture model and decision trees proved fruitful in predicting genre with features being MFCC.

The second attempt was to recommend songs to a specific user. This was attempted by learning user′s history of choice and then recommending new songs by matching the similarity between songs. The factors that were considered for finding the similarity were lyrics, melody (change in pitch with time) and genre of the songs. A statistical model for each of these factors was developed for both likes and dislikes of a specific user. Classical ″Bag of Words″ model for text was used for classification using lyrics. Melody, being single dimensional feature was worked upon using Hidden Markov Model taking the advantage of repeating pitch in songs. Classification based on genre was done by first pre-processing MFCC features and then using a classifier to perform the classification. A test song is then tested with all the developed models and probability obtained with these models is used for giving the final decision. The system is currently performing with around 75% accuracy and has been tested for three different users.

Some other attempts were also made to directly match a test song with the training songs to predict the category of songs. Such efforts included finding deviation between probability distribution of two songs using KL divergence, finding similarity between songs using techniques like cross-correlation, Rough Longest Common Subsequence (RLCS) and Markov Model like technique. But these techniques did not prove very satisfactory.

TABLE OF CONTENTS

[1]Details of classification techniques can be found in appendix.

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
RF	Random Forest
RLCS	Rough Longest Common Subsequence
TF-IDF	Term Frequency-Inverse Document Frequency
MFCC	Mel Frequency Cepstral Coefficients
SVM	Support Vector Machine
IOI	Inter Onset Interval
LCS	Longest Common Subsequence

CHAPTER 1

INTRODUCTION

1.1 Overview

Recommender systems are being used quite often everywhere in today's era which include areas like e-commerce, audio, video, search engines etc. These systems try to rate an unknown item for a given user based on certain criteria which help in deciding whether a person is likely to go for an unknown item or not. Two broad categories of recommender systems are content-based recommender systems and collaborative filtering based recommender systems. Collaborative filtering based recommender systems recommends a new item depending on the fact that if two or more people have similar kind of choice for past items, then there will be high probability that they will have similar liking for new items as well. On the other hand, content-based recommender systems focus on the content of the item and recommends it to the user based on the history of choice of the user. The content used for recommendation can vary from purpose to purpose. So, this area of study is vast. Our work is based on recommending new songs to a user based on the content of the music and user's history of choice.

For achieving the purpose, we initially started working on genre classification of Bollywood songs. A dataset having seven classes of genres was created for achieving the purpose. Based on studies of Scaringella *et al.* [2006] and other researchers, it was concluded that genre can be best identified by MFCC (Mel Frequency Cepstral Coefficients) obtained from songs. Various machine learning classifiers like Gaussian mixture model, support vector machine, hidden Markov model and decision trees were used for accomplishing the task. Classification accuracy of up to 80% was achieved using Gaussian mixture model. Decision trees also gave fairly good accuracy after reducing huge amount of data to small amount using Cooper summarization technique devised by Cooper and Foote [2002].

After this task, we moved on with another task where we tried to match new user queries with the user's history of likes and dislikes and gave recommendations accordingly. For this, we initially started with a user specific dataset containing songs that were rated as per user's choice. We started with comparison of melody and timbre features of songs for this task. For obtaining melody, pitch features were used and for obtaining timbre, MFCC features were used. Various direct song to song matching techniques like K-L divergence, cross-correlation, Gaussian mixture model, Markov model, Rough Longest Common Subsequence and Hidden Markov Model were applied, but none proved fruitful up to expectation. So, we felt that some different and *intelligent* approach should be followed to get to the results and not the direct one.

Since Bollywood music plays a lot of emphasis on lyrics of songs with lyricists spending lot of time writing lyrics, any song's taste should get identified using its lyrics. Hence, lyrics were then used to determine likeness. Second thing that we did was to modify the pitch features and convert it to *tonic normalized* pitch features. As per study.com, *"tonic or tonic pitch, in music is what we refer to as the beginning and ending note of the diatonic scale that is used to compose a piece of music"*. So by tonic normalizing each song with its tonic, we were bringing pitch of every song to the same level where it could be compared to other songs. After modifying pitch features, a hidden Markov model was trained based on the principles of *raga verification* which improved the results. The third attempt was to modify the MFCC features in the same fashion as was done in genre classification task using technique by Cooper and Foote [2002] and choosing the small set of features within a song that best identifies a song.

We then tried to combine lyrics, melody and genre using scores fusion technique and the results improved some of the times, while resulting in confusion rest of the time.

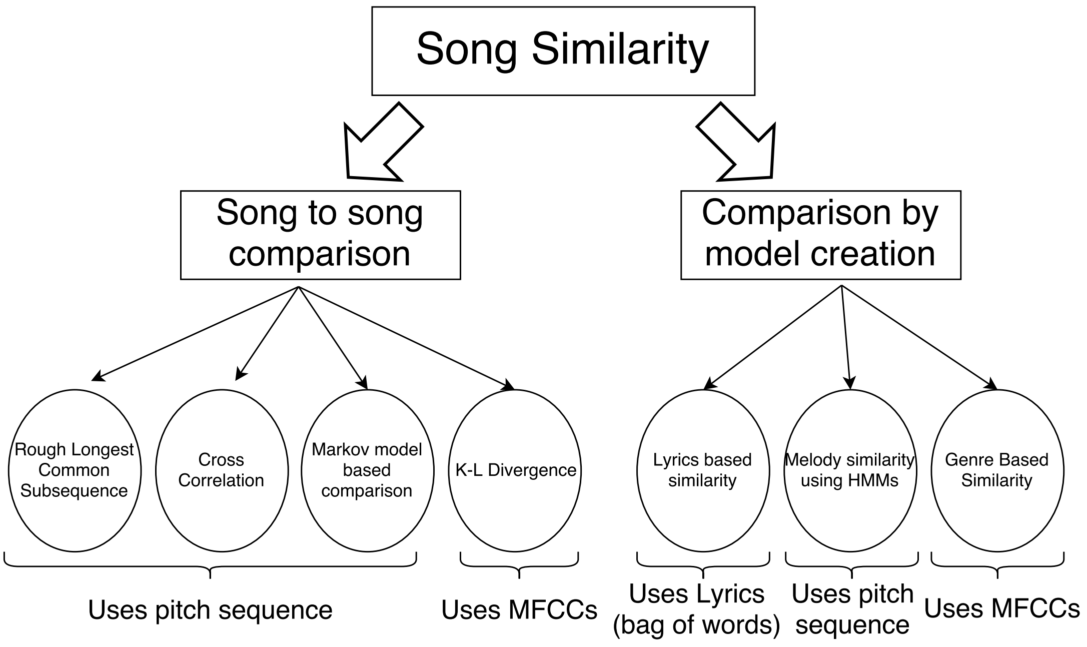


Figure 1.1: Techniques applied for song similarity match.

1.2 Motivation

Music is personal and listening choice is also personal. The first idea of genre classification of songs was based on this fact only. Many music websites have a section where they categorize various genres of music so that a user can listen to the choice of song he/she wants. Various genres were picked from a section of online music website *saavn.com* and their classification was attempted.

After achieving significant performance on classification of various genres, we aimed at the other problem of recommending songs to users based on history of choice. Recommender systems are trending a lot in current era, but the technique involved in recommendation is collaborative filtering. This technique requires large amount of user data and requires many users to give the best recommendation. Also, it suffers from cold start problem where if a new item arrives, it can't be recommended as no other user has liked it. Considering the two mentioned issues, content based recommendation can prove useful where recommendations can be given to an isolated user with even less amount of train data and also doesn't suffer from cold start problem.

Although the difficult part in content based recommendation is the choice of content for recommendation. Even for recommending one type of item, many types of content need to be considered which increases the complexity of problem. The reason behind this is that the choice of humans do not stick to one content for liking or not liking a particular item. So, lot of experimentation and analysis was required while attempting this problem.

1.3 Major Contribution

- System is based on content based recommendation rather than widely used collaborative filtering based recommendation.
- System eliminates the cold start problem where a new song is hard to recommend.
- System takes into consideration various aspects of music like genre, melody and lyrics to recommend which makes it highly versatile.

1.4 Organization of Thesis

Chapter 2 discusses in detail the genre classification task performed for seven classes of songs. Next chapter in line is chapter 3 where initial attempts for classification of songs based on likes and dislikes of user is given. This chapter discusses direct song to song matching techniques that were tried for recommending songs. After attempting this task, model based techniques were tried which is discussed in chapter 4. Experiments performed on all the above discussed techniques are discussed in the respective chapters only. Chapter 5 concludes the work with critical analysis of work done.

CHAPTER 2

GENRE CLASSIFICATION

2.1 Introduction

Many of the online music repositories like *www.gaana.com* and *www.saavn.com* classify songs in their repositories into various genres. This help a user to listen to songs of specific genre s/he wants. The attempt to classify Bollywood songs into various genres was motivated from these websites only. Figure 2.1 shows how songs are classified into various genres on *saavn.com*.

This chapter is divided into four sections. Section 2.2 describes related work done in the area of genre classification. Section 2.3 describes the dataset used for the classification task. Section 2.4 defines the feature extraction from audio files used for classification and the last section describes the pre-processing techniques and final classification techniques used for classification of genres.

2.2 Related Work

Genre classification has been done on western music, but not on Bollywood music. Also, genres of western music are very much different from Bollywood music. In western music, there are genres like jazz, hip hop, rock, pop etc which are not so popular in Bollywood music. Variation in melody in Bollywood music is more as compared to western music. So, that work is not exactly comparable to current

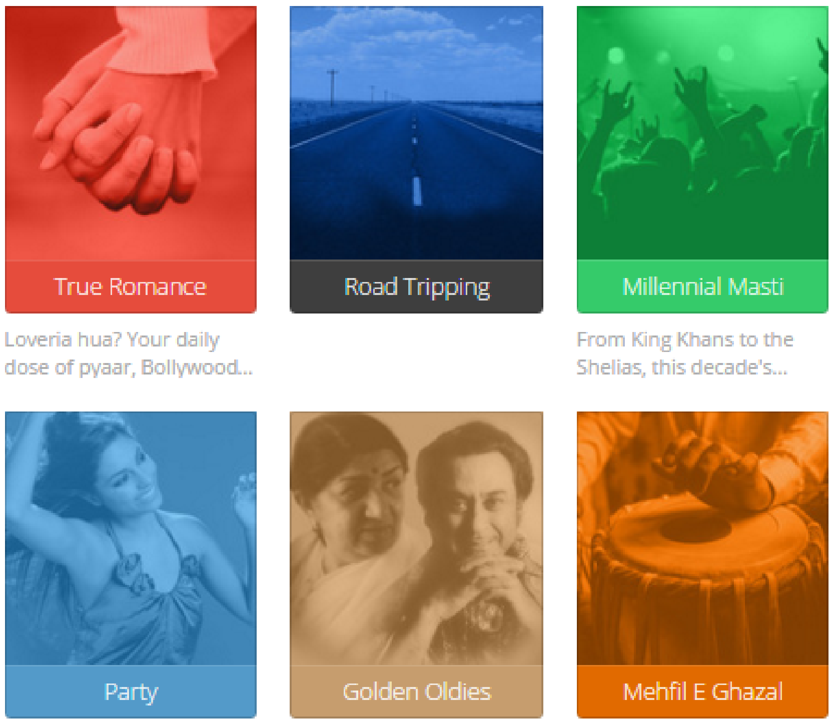

True Romance
Road Tripping
Millennial Masti
Loveria hua? Your daily dose of pyaar, Bollywood...
From King Khans to the Shelias, this decade's...
Party
Golden Oldies
Mehfil E Ghazal

Figure 2.1: Snapshot taken from *saavn.com* showing various classified genres.

work. For genre classification, researchers like Pye [2000] and Scaringella *et al.* [2006] used MFCC features and SVMs were outperforming other classifiers in case of western music. Pitch feature based genre classification has been attempted by Salamon *et al.* [2012].

2.3 Dataset

Data set for this task was created manually. Data set consists of seven classes of genres, all taken from Indian Bollywood music. These genres are as

- Oldest Bollywood songs from year 1950 to 1970.
- Purely classical songs based on Indian ragas.
- Ghazals.
- New party songs.

- New light songs.
- Old party songs.
- Old light songs.

Data set contained about 40 songs of each class of which about 70% were used for training and the rest were used for testing.

2.4 Feature Selection

This is the major part of our work because it is choice of features for any classifier that results in the efficiency and effectiveness of classifier. We used Mel Frequency Cepstral Coefficients (MFCC) for the extraction of features from the songs.

The MFCCs used in my work has following configuration :

- Sampling rate : 44.1 KHz
- Window Length : 0.1 s
- Window shift : 0.01 s
- Number of Cepstral Coefficients : 13, 20
- Number of filters : 26
- FFT size : 1024
- Low frequency : 200 Hz
- High frequency : None
- Pre-emphasis Coefficients : 0.97
- Lifter to Cepstral Coeff : 22

The choice of this MFCC[1] configuration is playing a major role in performance of the system. This configuration is chosen based on the work that was done earlier in the field of genre classification of music.

[1]details of MFCC extraction is available in appendix.

By observing the configuration, one can see that the window length is very small (0.1 second) and window shift is further small (0.01 second). Such configuration with low window length and shift increases the resolution of extraction and gives the best detail that can be obtained from the song. The average length of a Bollywood song is about 5 minutes, so the number of feature vectors extracted from a song is very huge. In our case, we were getting around 50000 feature vectors for a song. Also, the model was trained for about 70% of songs. So, the data that was there was huge.

2.5 Song Summarization

The raw set of MFCC features for a single song was so huge that it was not intuitive to apply techniques like SVM and decision trees directly on them. So, options were explored for reducing the feature matrix size per song. In order to account for the time series nature inherent to songs, it was necessary to take the consequent MFCC feature of a particular length of each song. In most papers related to music content analysis, the second 30 second clip was taken for this analysis. However, it was not intuitive that this approach will work in case of Bollywood music.

So, in order to summarize the huge number of MFCC features, a summarization technique by Cooper and Foote [2002] was used. It is based on cosine similarity analysis where the cosine similarity between every other feature vector is calculated and stored in a matrix format. Conclusions are later drawn from this matrix. However, to create the similarity matrix, $O(n^2)$ operations are needed. A point here is an MFCC feature vector. Since a single song has about 50,000 MFCC features, matrix computation was an expensive process. Hence, we

1. Sampled the songs at 10 different starting points
2. Computed the summary of length 200 for each of the segment
3. Merged these summaries
4. Computed the final summary of length 500 to represent the entire song

The process of summarization is as :

1. If there are n features, a cosine similarity matrix of size $n \times n$ is formed.
2. All rows are summed up of this $n \times n$ matrix and a column matrix is obtained. i^{th} element in this column vector shows similarity of i^{th} feature vector with all other feature vectors.
3. A consecutive length of $n/20$ having maximum sum is chosen giving maximum similarity.

A sample similarity matrix of a song is shown in figure 2.2. The segments that are light blue in color (or light colored in case of black and white image) are those which are more similar to the rest of the song. Figure 2.4 summarizes the process

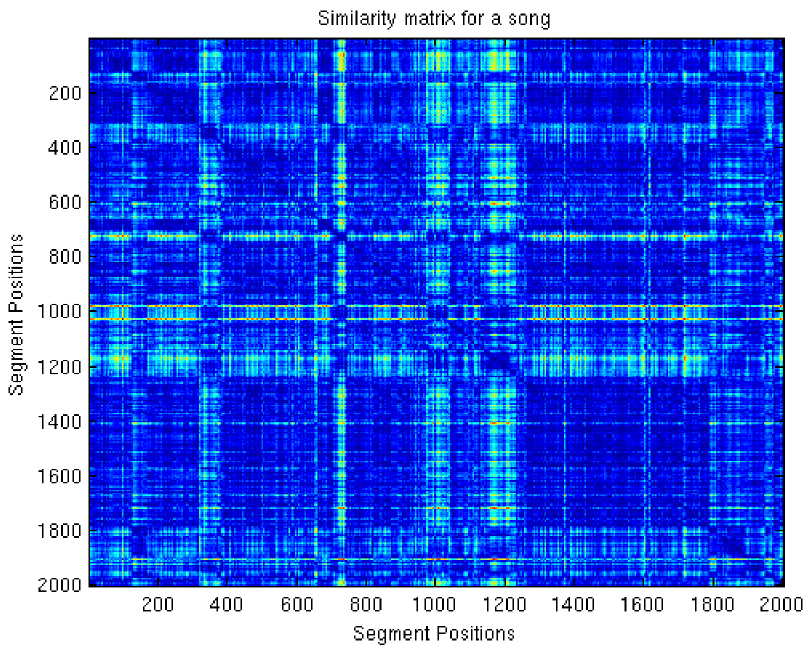


Figure 2.2: Similarity matrix

of song summarization at a broader level while figure 2.3 shows how each matrix is calculated and is worked upon.

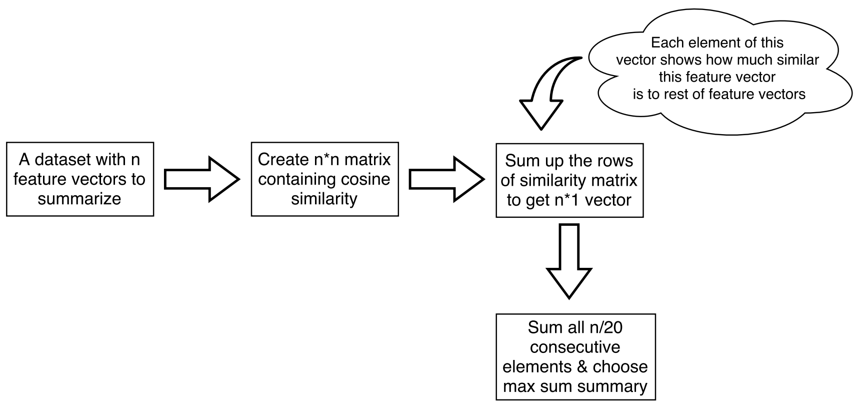

A dataset with n feature vectors to summarize
Create n*n matrix containing cosine similarity
Sum up the rows of similarity matrix to get n*1 vector
Each element of this vector shows how much similar this feature vector is to rest of feature vectors
Sum all n/20 consecutive elements & choose max sum summary

Figure 2.3: Process of song summarization at smaller level.

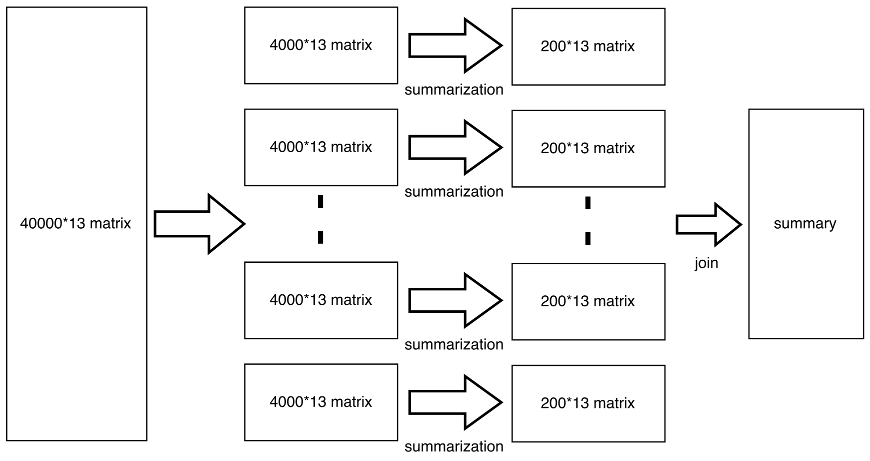


Figure 2.4: Process of song summarization at broader level.

2.6 Classifiers Used

Following classifiers were used for genre classification :

1. Gaussian Mixture Model[2] : Gaussian mixture model with 90 mixtures was used. This number was chosen after experimentation. The complete data set of songs was used for the experiment.
2. Continuous Density Hidden Markov Model[2]: Ten states were chosen in the model, approximating it with the number of 'Swara' in Indian music. Each state was a mixture of 3 Gaussians as there exists multimodality even in one 'Swara'. HTK toolkit was used to train and test the Hidden Markov Model. Since the training time of HMMs was huge for the complete songs, we had to go for 50 seconds clip for each song. The clip was not chosen from the beginning of the song, but was chosen from in between.
3. Random Forest[2]: Random forest were used after summarizing the data using Cooper summarization technique. Number of trees used in random forests were 50. This was again obtained after experimentation. Matlab toolkit was used for implementing random forests.
4. Support Vector Machine[2] : C-SVM with Gaussian kernel was used in this technique. Software used was SVM torch. We experimented with various values of C and kernel width, but it did not prove very much fruitful.

2.7 Experimental Results

As talked in section 2.3, seven classes of genres were classified using various machine learning techniques and using MFCC as features. GMM, SVM and RF were ran using matlab toolkit while HMM was ran using HTK toolkit. Following table shows the performances of various classifiers.

[2] Details of classifier can be found in Appendix.

Model Name	Accuracy	Training Time Taken	Space Taken
GMM	80%	10 hours	30 GB RAM
HMM	65%	2 days	1.5 GB RAM
SVM	53%	4 hrs	1000 MB RAM
RF	75%	10 mins	300 MB RAM

Table 2.1: Performance of various models for genre classification task.

2.7.1 Inferences

Following inferences were made from the results in table 2.1 :

1. As seen in the table, GMM is most accurate. This is due to the soft clustering nature of GMM and also due to its ability to model any kind of distribution through an appropriate number of clusters.
2. Next in line is Random Forests. It is amazingly fast in terms of the training period. However, the size of the model is large in case of RF, around 180 MB. Also, it required preprocessing of songs to produce summary clips.
3. The issue with models other than GMM is that we were unable to use the full training data. The reduction of song length to smaller sizes might be the cause of lesser performance.
4. In HMM, analysis to find intelligent analogies from song data to states could help in deciding the number of states more appropriately.
5. In the case of SVM, poor performance is due to the highly overlapping nature of classes and difficulty to reach the correct value of parameters.

CHAPTER 3

DIRECT SONG MATCHING

3.1 Introduction

After performing the task of genre classification, we moved ahead with the task of performing song classification based on the likes and dislikes of a user. In order to predict this, we did the song matching based on very obvious and traditional techniques. The features used were MFCC which can define genre of the songs as discussed in chapter 2 and the other is pitch sequence, which can define the melody of a song.

The dataset we were working on contained songs rated in five different categories. So, the first attempt was to work on all five categories but soon it was realized that the problem is not that easy. So, the problem was reduced to two class problem where motive was just to classify into likes and dislikes with respect to a specific user. Although most techniques were found unsatisfactory for this task.

3.2 Related Work

There has been a lot of work related to song matching in the area of music information retrieval. But this work is related to mostly query by humming or cover song detection. Researchers like Shifrin *et al.* [2002], Zhu and Shasha [2003] and Ghias *et al.* [1995] have worked on various techniques for solving the problem of query by humming and cover song detection. Their work included techniques like string matching of pitch sequence and creation of hidden Markov model for every song and then matching it. Researchers like Schnitzer *et al.* [2010] have worked on K-L divergence between songs to identify the song similarity. In most of these techniques, goal is exact match between songs. Hence, a lot of works has been been done in this area. But the problem that we are dealing with is related to approximate match and not exact match, since no two songs will be exactly similar in the playlist of a person. Also, most of the times, it is not necessary that two approximately matching songs are liked by the same user, as the criteria for matching songs can be different from user's criteria of like or dislike. This was observed after getting not so good results after experimenting with above techniques.

3.3 Dataset

As presented in introduction, dataset for this task was obtained from a music expert. This dataset contained about 300 songs that were liked by the expert and about same number of songs that were disliked. Song matching was focused on two features, MFCC (Mel-Frequency Cepstral Coefficients) and pitch sequence (melody). Configuration of MFCCs is same as used in section 2.4. Melody was extracted from melody extraction toolkit named Melodia, developed by Salamon *et al.* [In Press (2013]. Default configuration of the software was used for feature extraction.

3.4 Song Matching Techniques

3.4.1 KL Divergence

In statistics and probability, KL divergence or Kullback-Leibler divergence is a measure of the difference in two probability distributions. It is not a symmetric distance and it defines the amount of information loss when one probability distribution is approximated over other probability distribution. This is also known as the relative entropy of one probability distribution over other. Expression for KL divergence (D_{KL}) for two probability distribution P and Q is given by

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} \tag{3.1}$$

Expression of KL divergence(D_{KL}) for two multivariate Gaussian distribution P and Q is given by

$$D_{KL}(P||Q) = \frac{1}{2}[log \frac{|\Sigma_2|}{|\Sigma_1|} - d + Tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1)] \tag{3.2}$$

where Σ_1 and Σ_2 are covariance of both the distributions respectively and μ_1 and μ_2 are means of both the distributions respectively. Since this distance is not symmetric and is non intuitive, so difference between two distribution is approximated by Jensen-Shannon divergence. This is given by

$$D_{JS}(P||Q) = \frac{D_{KL}(P||Q) + D_{KL}(Q||P)}{2} \tag{3.3}$$

Hershey and Olsen [2007] have also approximated this KL divergence for Gaussian Mixture Model. We have used the same in our work. Exact implementation of KL divergence between GMMs can be seen in their paper.

Work by Schnitzer *et al.* [2010] on song similarity has shown the use of the KL divergence on MFCC features with single Gaussian distribution. We tried following experiments, but since the likings and dislikings of user were not directly related to genre similarity of songs, satisfactory results were not obtained.

- Finding KL divergence between two songs, with each song being single Gaussian.
- Finding KL divergence between two songs, with each song being mixture of Gaussians.
- Converting each class into m mixture of Gaussians and each song into n mixture of Gaussians (n<m), then finding KL divergence between each pair of Gaussian component and considering the weighted minimum of all as divergence between two songs.

3.4.2 Cross Correlation

Cross correlation is a measure of similarity between two time series signals as a function of lag of one signal with respect to another. This is also called sliding dot product. So, this distance measure is trying to align two signals in every possible way and hence gives the maximum similarity two signals can have in their time shifted versions. Pitch in music can also be considered as a time series and hence this distance measure somewhat proved better than the other techniques. Expression for cross correlation $w[t]$ of two discrete time series signals $f[t]$ and $g[t]$ is given by

$$w[t] = f[t] * g[t] = \sum_{k=-\infty}^{\infty} f[k]g[t+k] \tag{3.4}$$

In the context of current work, melody in a song can be considered as signal. Cross correlation between mean normalized pitch of two songs was found and was divided by energy of both pitch sequence to give a measure of similarity between two pitch sequence. Division by energy was done in order to normalize the score.

3.4.3 Markov Model (based on work of Shifrin *et al.* [2002])

If we want to match two songs using traditional HMM, then first we need to create an HMM model for every train song. But since we have very limited data for each song, we cannot apply forward-backward algorithm to train and create each HMM model. So, an alternative approach is suggested which is more like a Markov chain based on note onset within a song. Since the note onset was not known in our work, we have used change in pitch as note onset. This method is used on pitch features.

Following is defined in order to construct it :

- A pitch transition between pitch n and pitch n+1 is represented by *deltaPitch* and *IOIratio*.
- deltaPitch is the difference between pitch n and pitch n+1 (not considering same pitch values at two samples).
- Inter onset interval(IOI) is the difference between onset of pitch n and pitch n+1.
- IOIratio is the ratio of consecutive inter onset intervals.

After finding deltaPitch and IOIratio, these are quantized into specific numbers using two global code books. Two HMMs are created per song, where both the HMMs have same states and state transitions but different observations. With single HMM, number of observations would lead to a larger number, as it will contain all the possible combinations of IOIratio and deltaPitch, so two different HMMs are being created. States are a made up of both deltaPitch and IOIratio.

The final probability of encountering any observation o_s, given any hidden state s can be written as

$$P(o|s) = P(deltaPitch_o|deltaPitch_s) * P(IOIratio_o|IOIratio_s) \tag{3.5}$$

After this, we need to find the three tuples of HMMs, namely initial state probabilities, state transition probabilities and observation probabilities for each state. Initial state probabilities are given by

$$\pi_i = \begin{cases} p, & \text{if } s_i = first \\ \frac{1-p}{|S|-1}, & otherwise \end{cases} \tag{3.6}$$

Here, p is the probability, $|\ S\ |$ is the number of states in the model and $first$ is the first observation state. p was kept less than one for the sequence to begin from any arbitrary state and not only from the first state. State transition probability can be calculated by counting the state transition from one state to another in actual sequence (state is a combination of deltaPitch and IOIratio). Observation probabilities can also be calculated by counting how often o_i is seen in a particular state s, compared to the total number of times s is entered. This can be written as

$$P(o_i|s) = \frac{count(o_i|s)}{\sum_{j=1}^{|O|} count(o_j, s)} \tag{3.7}$$

After creating this hidden Markov Model, further evaluations can be done in a traditional HMM way as usually done. A sample model based on the work is shown in figure 3.1. We tried the method as mentioned in the paper by Shifrin *et al.* [2002] for matching hummed queries with actual songs in database. But this method was too slow and also did not prove useful. This was tried on pitch

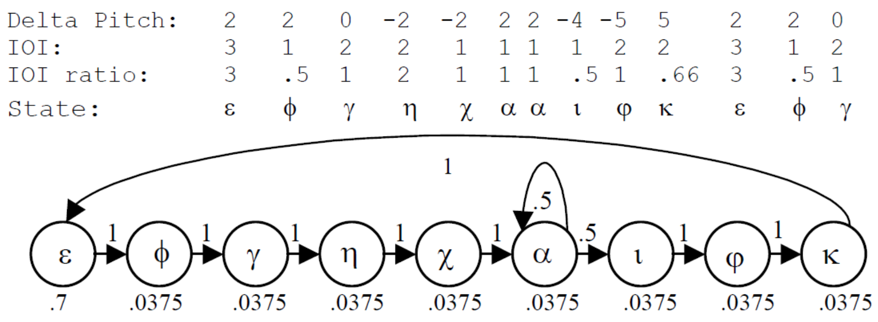

Delta Pitch: 2 2 0 -2 -2 2 2 -4 -5 5 2 2 0
IOI: 3 1 2 2 1 1 1 1 2 2 3 1 2
IOI ratio: 3 .5 1 2 1 1 1 .5 1 .66 3 .5 1
State: ε ϕ γ η χ α α ι φ κ ε ϕ γ
1
.5
.5
ε ϕ γ η χ α ι φ κ
.7 .0375 .0375 .0375 .0375 .0375 .0375 .0375 .0375

Figure 3.1: A sample model based on Shilfrin's work

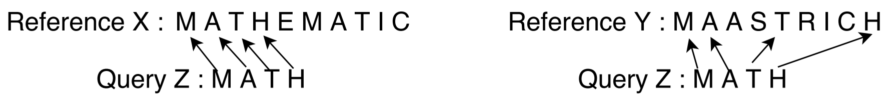
features.

3.4.4 Rough Longest Common Subsequence

Longest common subsequence is one of the most common technique used for matching strings. It is a dynamic programming technique and uses a two dimensional array for the calculation of longest common subsequence. But there are few heuristics that are applied on it by HWEI-JEN LIN and WANG [2011] in their paper "Music matching based on rough longest common subsequence".

In their paper, following two modifications were made in the traditional LCS :

- Instead of just longest common subsequence, density of longest common subsequence is considered. Figure shows the illustration : In the figure,

  
  Reference X : M A T H E M A T I C
  Query Z : M A T H
  Reference Y : M A A S T R I C H
  Query Z : M A T H

  query Z is matched against reference X and reference Y, and both correspond to LCS of length four, but it is more intuitive that reference X is more similar to query Z than reference Y. The reason is simple, longest common subsequence is spread over larger length in reference Y but lesser length in reference X. So, RLCS takes this into consideration.

- As the name says, rough longest common subsequence is a rough measure of finding longest common subsequence. Even if two elements are roughly equal, i.e. difference between them is less than a particular threshold, then the proportion by which they are roughly equal is added to the length of longest subsequence.

Expression for longest common subsequence is given by :

$$cost[i,j] = \begin{cases} 0, & \text{if } i.j = 0 \\ cost[i-1][j-1]+1, & \text{if } r_i = q_j \\ max(cost[i][j-1], cost[i-1][j]), & \text{if } r_i! = q_j \end{cases} \tag{3.8}$$

where r_i and q_j are the i^{th} and j^{th} element of reference and query respectively. $cost[i][j]$ gives the length of longest common subsequence upto i^{th} element of reference and j^{th} element of query. Now, the expression for RLCS is given as :

$$cost[i,j] = \begin{cases} 0, & \text{if } i.j = 0 \\ cost[i-1][j-1] + (1 - diff/threshold), & \text{if } diff \leq threshold \\ max(cost[i][j-1], cost[i-1][j]) - \delta, & \text{if } diff \geq threshold \end{cases} \tag{3.9}$$

In the above equation, $diff$ is the difference between i^{th} element of reference and j^{th} element of query. *threshold* tells the amount of difference one is fine with while finding rough longest common subsequence. δ is the penalty given when i^{th} element of reference and j^{th} element of query are not even roughly equal.

In the context of our work, pitch sequence was divided into pitch-value and duration-value (based on the paper by HWEI-JEN LIN and WANG [2011]). Duration-value corresponded to the amount of time for which a pitch value did not change. Difference between two elements of query and reference was a weighted measure of Manhattan distance. Expression for the same is given as

$$diff = w \times \Delta pitchValue + (1-w) \times \Delta durationValue$$

where w is the weight. In this method, songs in the dataset were segmented to 'pallavi' lines or chorus lines. This idea was obtained from motif spotting technique used in Indian classical ragas, a paper by Dutta and Murthy [2014]. The intuition behind doing this is that a person decides whether he likes a song or not just by listening the first line of song. Also, since different songs are sung in different musical scale, hence pitch of the songs was tonic normalized before finding the RLCS distance between them. Details of tonic normalization can be found in section 4.2.1.

Change in cost with match in RLCS can be depicted from figure 3.2. This figure shows how a query is matched against a reference. Portion of graph where the curve moves diagonally shows match of two elements of reference and query, whereas portion of graph where the curve is either diagonal or vertical shows the mismatch between elements of query and reference. Curve in the graph goes on denser with the increase in match. The darkness of complete curve is normalized between 0 and 1 throughout the starting match and the ending match.

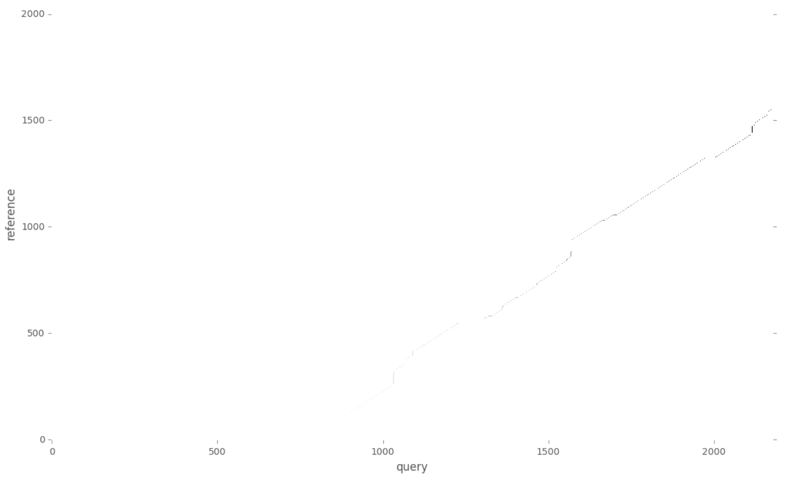


Figure 3.2: Illustration of match in RLCS.

3.5 Experimental Results

Experiments here were done to classify song into one of the category, either song liked or disliked by user. Initially, dataset used for the purpose was created by music expert where the songs were rated into five categories. These five categories were ratings based on liking of user. Experiments were initially performed to categorize songs into these five categories but there was lot of confusion. So, it was thought to work on two classes and not all five classes. These two classes were combination of high rated songs and low rated songs. In fact, performance of direct song to song matching was not upto mark. This can be observed in the table below :

Technique	Accuracy	Precision	Recall
Shilfrin's model on pitch	40%	0.28	0.5
Cross correlation on pitch	62%	0.72	0.66
KL divergence on MFCC	58%	0.68	0.65
RLCS on pitch	65%	0.70	0.65

Table 3.1: Performance of various techniques with direct song to song match.

3.5.1 Configuration and Performance of Techniques

A bit of detail on configuration and performance is as:

1. Shilfrin's model was too slow to work upon. The only configuration that was tried was with 25 states as mentioned in the paper by them. Also, pitch being used was not tonic normalized as done in few other experiments.
2. Cross correlation method was very fast and the good thing about it was that, there were no parameters to play with. Although that is a bit of disadvantage as well.
3. KL divergence has also not much parameters to play with. Three different kinds of experimentation were done, details of which is present in section 3.4, but result of all were same, so we have presented just the final result.

4. Rough longest common subsequence was the technique that proved best with respect to song matching technique, however this best result was in itself not satisfactory. There were three parameters to play with, namely, weight(w), threshold on distance for roughness(τ) and penalty for each mismatch(δ). The configuration which gave best result was w=1 (showing no weight should be given to duration and just pitch sequence should be considered), τ=0.15 (this is the normalized distance) and δ=0.5. This technique being dynamic programming technique, was on the slower side.

3.5.2 Inferences

Following inferences were made after performing these experiments:

1. Shilfrin's model being proved highly useful in in the area of query by humming did not prove useful to our work. The reason behind it could be, model looks for exact match while we were looking for similar taste of songs.
2. K-L divergence did not prove useful because it was applied on MFCC features which define genre and user's preference was not directly based on genre of the songs but some other factors as well. Also, K-L divergence was obtained between normal distribution of two complete songs. Later it was concluded that dominant clips in songs can prove more useful than using whole songs.
3. Cross correlation proved to be most useful among all. This technique is fast but still results are not promising from an application point of view.
4. Rough longest common sub-sequence being logically thought and devised, it helped in finding similar songs but could not prove to be useful since that similarity was not corresponding to user's choice. Also, there were lot of parameters to be played upon it to get the best results which made the technique slow for even 20 seconds of song clip.

3.6 Summary

This chapter focused mainly on direct song matching techniques without any kind of intelligent pre processing involved. Although all of these ideas were taken from various papers based on song similarity where techniques proved helpful to the researchers, but just finding similar songs did not work very well for us. We had to classify based on what human brain thinks when it comes to liking or disliking a song. But this stage was necessary for us to go through as said *"bad things have to happen before good can"*. Next chapter focuses on techniques where instead of direct songs matching, model was created for likes and disliked, mostly with a preprocessing involved.

CHAPTER 4

CLASSIFICATION BASED ON MODEL CREATION

4.1 Introduction

After working on number of song similarity techniques which involved direct song to song match, it was observed that there is no direct connection between song similarity and likes and dislikes of user. So, it was thought to go for model based techniques. After attempting direct model based methods, it was felt that some pre processing of features or intelligent machine learning is required to reach the goal. So that was done and results improved. Also, lyrics of songs as a new feature was added as lyrics can tell a lot about the feelings in a song. This was again a very important step which took the work in correct direction.

4.2 Dataset

Dataset for this problem is same as defined in section 3.3. The difference is that, along with MFCC and melody, lyrics of songs was also experimented and worked upon. Lyrics of about 300 songs per class was obtained manually from web. Since lyrics contained on web were scripted in roman (hindi words scripted in roman), there were lots of errors in it. So, errors were processed manually. Also, dictionary of words present in whole set of lyrics was reduced by combining words having similar meaning and same reference. This was again done manually. MFCC features were also summarized using technique devised by Cooper and Foote [2002]. The details of MFCC summarization can be found in section 2.5. Pitch features were also preprocessed and were tonic normalized. Tonic identification was performed by a musician. Although automatic tonic estimation techniques exist for classical music, tonic for Bollywood music is quite difficult owing to the large number of instruments used. So, experiments related to tonic normalized pitch were done on about 220 songs, divided into two class of like and dislike. Details of tonic normalization is as:

4.2.1 Tonic Normalization

In music, each note of a scale has a special name, called a scale degree. In a scale, first and last note is called tonic. So, same song can be sung in various tonics and if this is the case, using pitch sequence, it will become impossible to match two songs. In such a case, songs can be same but pitch sequence will be different. So, some way to bring the pitch sequence of all the songs to same level is required. This is done using tonic normalization. For this, tonic of every song needs to be found first. This was done by music expert in Don Lab, IIT Madras. After obtaining tonic for every song, following steps are followed to tonic normalize every song.

1. Divide the pitch sequence by tonic.
2. Find the log (base 2) of pitch sequence obtained after step 1. Call it logPitch.
3. Subtract sequence o f logPitch from its floor. This brings logPitch in range of 0 to 1. Call this base.
4. Find the second power of base and multiply it with 120. Call this cent.
5. Subtract 120 from cents and multiply with 10. This is tonic normalized pitch in range from 0 to 1200.

4.3 Classification techniques used[1]

This section discusses the various classification techniques used for classification using model creation. Different classification techniques used with different features are discussed in this section.

4.3.1 Gaussian Mixture Model with MFCC

Initially, Gaussian mixture model with MFCC features were tried with various numbers of mixtures. Choice of number of mixtures is based on the 5-cross validation that was done on validation data. This was one of the first attempts when we tried to classify songs into likes and dislikes. It did not work. But as soon as many experiments began to fail, we thought of applying the same Cooper summarization technique that was used in chapter 2 of genre classification task. This technique has proved helpful for many researchers and the same happened with us. Details of technique are already explained in the thesis in section 2.5. After performing this, performance was gained by about 10%. Matlab toolkit was used for implementing GMMs.

The main reason behind this technique being proved useful is that when we were trying with MFCC features of full song, there was lot of confusion as there was lot of data that was redundant. So, when the redundant data was reduced to only meaningful one using the technique, classifier was able to perform better.

[1]Details of classification techniques can be found in appendix.

4.3.2 Hidden Markov Model with MFCC

After trying GMMs on MFCC features, an attempt was given to check whether sequence information is playing any major role in predicting likes and dislikes. Ten states were chosen in the model, approximating it with the number of 'Swara' in Indian music. Each state was a mixture of three Gaussians as there exists multimodality even in one 'Swara'. HTK toolkit was used to train and test the Hidden Markov Model. Since the training time of HMMs was huge for the complete songs, we had to go for 50 seconds clip for each song. The clip was not chosen from the beginning of the song, but was chosen from in between. Even after performing this, results were not good and something out of the box was needed to be thought.

4.3.3 Hidden Markov Model with Pitch

The earlier way of using HMMs with MFCC was not intelligent. Task of initialization of means of states was given to HTK toolkit and hence good results were not seen. But same thing was not repeated here. The first step while working with pitch was to normalize the tonic and bring tonic of all songs to the same level so as to use them for comparison.

After normalizing the tonic, the pitch histogram of collection of likes and dislike, separately was seen and it looked something like in figure 4.1. In the figure, one can observe that certain pitch values are repeating again and again. So, the idea was to initialize the means of states of HMM to the peaks in the histogram. This ensures uniform distribution of pitch values across various states. Work based on Indian classical raga verification is also based on same fundamentals.

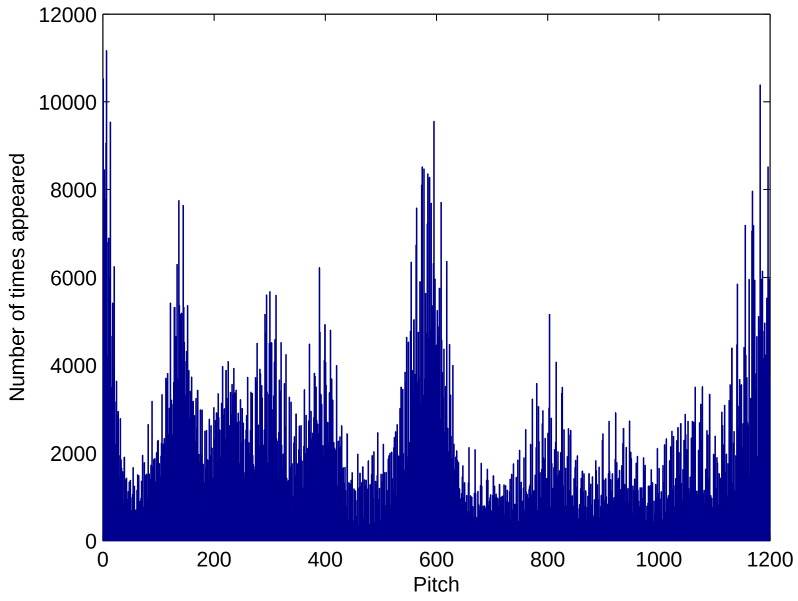


Figure 4.1: A sample histogram of pitch of all songs combined.

4.3.4 Classification using Lyrics

When it comes to music, we sometimes forget to feel the emotions present in lyrics. This is what happened with us. We just kept the focus on music and never thought to consider lyrics. Lyricists spend significant amount of time on writing the lyrics of song and try to fill that with the emotions. So, lyrics must contain something which can decide whether a person will like a song or not. Figure 4.2 shows how the lyrics were processed to perform classification. Firstly, lyrics of all songs

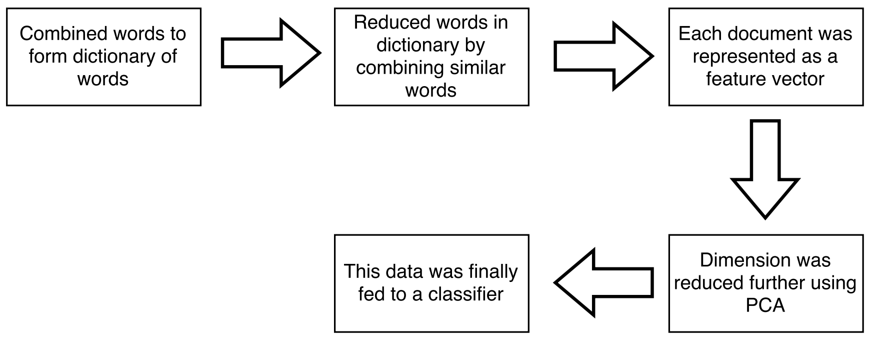

Combined words to form dictionary of words
Reduced words in dictionary by combining similar words
Each document was represented as a feature vector
Dimension was reduced further using PCA
This data was finally fed to a classifier

Figure 4.2: A simplified process showing classification using lyrics.

were collected manually for web. Hindi lyrics were obtained in roman script from web. Because of this, lot of errors were present in the lyrics. So, dictionary of all the words was created and error was removed from them. Even after reducing errors, dictionary obtained contained huge number of words. For about 600 songs, dictionary contained 9000 words (for one-gram). This was again reduced by combining similar words. Similar words refer to words having similar meaning or synonyms. This process was done manually and this reduced the dictionary size from 9000 words to about 4000 (for one-gram). After processing these words, every document was represented as a feature vector having dimension equal to length of dictionary, where weight of each dimension was considered as a product of term-frequency and inverse document frequency (tf-idf). TF-IDF feature vectors are widely used for processing bag of words model. Experiments were carried out for one-gram and two-grams of data.

Dimension of the dataset was further reduced using PCA. Choice of number of dimension in PCA was experimented in the range from 50 to 500. While doing so, it was ensured that dimension is as minimum and 99.5% variance of data of data is retained. This data was then given to two classifiers, Gaussian Mixture Models and Support Vector Machine. A bit of detail on classifiers is as :

- Gaussian Mixture Model: GMMs with various number of mixtures was experimented for each class of songs. Choice of number of mixtures was done using 5-cross validation with validation data. Experimentation was done for both one-grams and two grams of data. Toolkit used was python scikit learn toolkit.
- Support Vector Machine: C-SVM was experimented with gaussian kernel was experimented with various values of C and kernel width. This was again experimented on one-grams as well as two-grams using python scikit learn toolkit for SVMs.

4.3.5 Late Fusion of Scores

After getting the performance on various classifiers having different features, it was intuitive to combine the scores of various classifiers to know if there is any disjoint set of scores that is being formed in order to improve the current performance of classifiers. Scores were fused of three best performing models, namely HMM with pitch, GMM on MFCC and GMM on lyrics. In order to perform the operation, following techniques were used :

1. **Random Forest** : First way used was to combine the scores and feed them to Random Forest. Since the scores from the three classifiers were in different range, they were normalized in the range 0 to 1. These scores were then fed to random forests in usual way where data was divided into train, test and validation, and performance of score fusion was evaluated.
2. **Weighted Scores** : In this technique, scores of the three techniques were normalized and a weight corresponding to score of each feature was found that corresponded best on training, such that

   $$w1 + w2 + w3 = 1$$

   After getting the values of weights from training, final score for comparison between likes and dislikes was calculated as :

   $$finalScore = w1 \times score1 + w2 \times score2 + w3 \times score3$$

   and this $finalScore$ was used for final evaluation.

4.4 Early Recommendation

Since the technique that is proving promising for recommendation is based on creation of model and requires data, so recommendation with less number of songs is a difficult task. Most of the experiments done till now are done on either about 200 songs or 600 songs, which in itself is not less.

In order to handle the situation where training data for recommendation is very less, some very preliminary concept of semi supervised learning are used. Same models are generated as talked about in this chapter earlier, namely HMM with pitch and GMM with lyrics and MFCCs, but there are two differences,

1. Data for training the models is less.
2. Configuration of model is difficult to decide as no validation data exist.

In order to deal with these two issues, following steps are considered :

1. While prediction, only those predictions are given which are getting predicted with at least certain confidence threshold.
2. Parameters for the model are decided based on certain criteria discussed in more detail in the chapter below.

Following variations are made to the three models used for classification with less number of songs:

1. **HMM with Pitch** : The confidence threshold used here is 0.85. This is decided after experimentation. Another important thing to decide here is choice of number of states, and this was decided to be five after experimentation done on three different users.
2. **GMM with MFCC** : This model also used the confidence threshold of 0.85. Choice of number of mixtures in GMM model was decided after analyzing the elbow point in plot of log-likelihood of train data and number of mixtures. After certain number of mixtures, it was observed that change in log-likelihood becomes very less. The number of mixtures where this happens is usually called the elbow point and is the optimal number of mixture that is chosen. A sample elbow analysis is shown in figure 4.3.
3. **GMM with Lyrics** : Unlike above two classifiers, where even with 20 songs, we have large number of feature sequence or feature vectors which are not very less for training the model! However, while using lyrics, we just have one feature vector corresponding to each song, with huge dimension. In order to work in this situation, dimension of the data was reduced to one less than number of examples using PCA and diagonal covariance was used in Gaussian mixture model. Also, the choice of number of mixture were experimented from 1 to 5 in the same way as done with GMM with MFCC.

4.5 Experimental Results

As seen in the experiments of last chapter, techniques used there were quite direct and did not use any kind of preprocessing of data. So, something new was needed to be tried to classify the songs into likes and dislikes. So, model based techniques were tried, as talked in previous sections of the chapter. Even in model based classification techniques, direct GMM and HMM model creation techniques having raw MFCC features did not prove useful. So, lyrics in addition to pitch and MFCC features were experimented but with processing involved. The process of feature extraction and processing of these features has been talked about in previous sections of the chapter. Following subsections illustrates the performance for each technique applied. Configuration used in these classifiers is discussed after the performance of various techniques in the current section.

4.5.1 GMM and HMM on Raw MFCC

Performance of GMM and HMM on raw MFCC (i.e. without processing) can be seen in table 4.1. GMM was trained on complete set of songs, while HMM was trained on about 50 second clips of songs chosen randomly between songs. Both of the models were trained on about 300 songs per class.

Classifier	Accuracy	Precision	Recall
GMM	56%	0.63	0.65
HMM	54%	0.61	0.57

Table 4.1: Performance of GMM and HMM on raw MFCC features

4.5.2 GMM and SVM on Lyrics

Performance of lyrics on GMM and SVM with one-gram and two-gram of words considered can be found in table 4.2. This table shows the best performance obtained after 5-cross validation on validation data. Accuracy mentioned in the table is average accuracy over the five folds. The performance shown is on about 600 songs, divided in two classes of like and dislike.

Classifier	Accuracy(Avg.)	Precision(Avg.)	Recall(Avg.)
GMM (1 gram)	70 %	0.74	0.81
GMM (2 gram)	72 %	0.78	0.79
C-SVM (1 gram)	66 %	0.71	0.76
C-SVM (2 gram)	68 %	0.76	0.66

Table 4.2: Performance of GMM and SVM on TF-IDF features of lyrics.

4.5.3 GMM on summarized MFCC

Performance of summarized MFCCs on GMM can be found in table 4.3. This table shows the best performance obtained after 5-cross validation on validation data. Accuracy mentioned in the table is average accuracy over the five folds. Again, this was computed on 600 songs, divided in two classes. MFCCs obtained from songs were first summarized using Cooper summarization and then experiments were performed.

Classifier	Accuracy (Avg)	Precision (Avg)	Recall (Avg)
GMM	68 %	0.63	0.85

Table 4.3: Performance of GMM on summarized MFCC features

4.5.4 HMM on pitch

Performance of HMM on pitch features can be found in table 4.4. Since pitch in this experiment was tonic normalized and tonic was obtained manually, which was bit tedious task to perform on Bollywood music, this experiment was performed on 100 songs per class. Tonic was found manually by a musician and the accuracy shown in the table in average accuracy on validation data over the 5 folds of validation.

Number of states in HMM	Accuracy	Precision	Recall
4	60 %	0.57	0.68
5	60 %	0.66	0.55
6	58 %	0.60	0.51
7	54 %	0.53	0.46
8	51 %	0.54	0.45

Table 4.4: Performance of HMM on tonic normalized pitch.

4.5.5 Fusion of Scores And Scalability Check

Scores obtained from three techniques, namely GMM on lyrics, HMM on normalized pitch and GMM on summarized MFCC were fed to two different techniques as discussed in section 4.3.5, namely random forests and weighted fusion. Also, in order to check the scalability of applied techniques, these techniques were tested on two more users. For every user, techniques were tested on a database of about 200 songs where every user classified a song as either liked or disliked. Table 4.5 shows the results of fusion of scores on three different users.

Feature	User1	User2	User3
GMM with Lyrics	72 %	**76 %**	73 %
HMM with Pitch	60 %	62 %	54 %
GMM with MFCCs	68 %	61 %	**79 %**
Fused with decision trees	68 %	**78 %**	76 %
Fused using weighted scores	64 %	68 %	70 %

Table 4.5: Performance on three users along with fusion of scores.

4.5.6 Early Recommendation

As discussed in section 4.4, this subsection discusses the situation where train data is very less. Since this is not a normal situation, experimentation for this is done on all the three users. Table 4.6 shows the performance of the three classifiers along with the threshold selected. Choice of threshold was such that large number of songs with low confidence gets eliminated.

Feature	Threshold	User1	User2	User3
GMM with Lyrics	0.9	76 %	64 %	70 %
HMM with Pitch	0.85	77 %	86 %	57 %
GMM with MFCCs	0.85	61 %	59 %	68 %

Table 4.6: Performance on three users having less number of songs for training.

The number of mixture for GMM were chosen using elbow analysis of plot of log likelihood and number of mixtures. A sample plot is shown in figure 4.3. Seeing the figure, optimal number of mixtures look around 20.

The choice of number of states in HMM modeled with pitch sequence was found after experimentation. Optimal number of states was found to be five after experimentation. Table 4.7 shows the performance after varying the number of states, obtained after 5-cross validation.

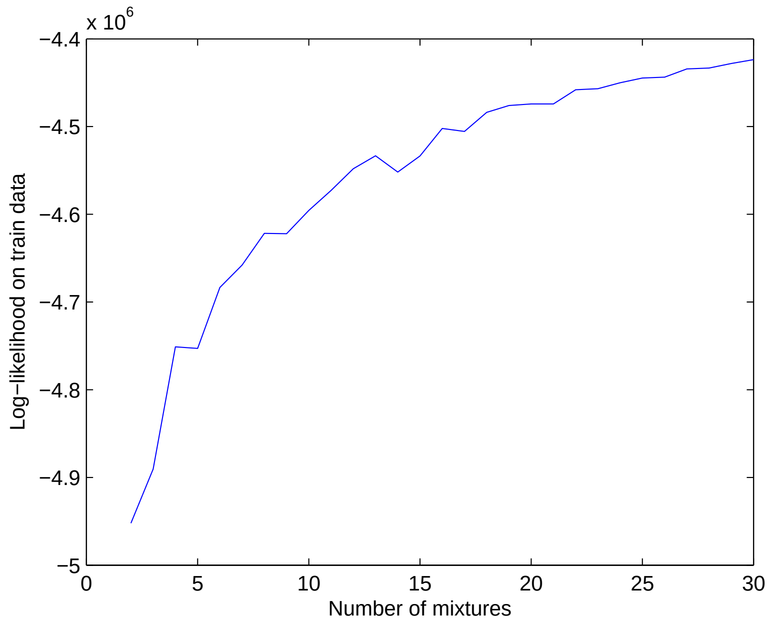


Figure 4.3: Elbow analysis for the choice of number of mixtures for early recommendation.

Number of states	User1	User2	User3
4	52 %	66 %	35 %
5	**77** %	**86** %	**57** %
6	64 %	68 %	55 %
7	67 %	57 %	48 %
8	42 %	88 %	48 %

Table 4.7: Performance of HMM with pitch with varying number of states for three different users.

4.5.7 Configuration of Classifiers

Following parameters were played with while experimenting with various classifiers :

- With Gaussian Mixture Model, experimentation was done with various mixture components. Configuration with best average accuracy over 5-folds of validation was chosen but in case of early recommendation when less data is present, plot of log-likelihood with mixture helped us decide the number of mixtures. Matlab toolkit and python scikit-learn was used for the implementation of it. With lyrics, only diagonal covariance was used since number of examples were not huge to approximate large number of values of covariance matrix.
- With SVM, experimentation was done with value of C (weight given to slack variables) and kernel width. Because of large amount of data, these values were experimented by changing them exponentially. Python scikit-learn toolkit was used for experimentation on it.
- With HMM on pitch, experimentation was done with various number of states, ranging from 4 to 15. Here, the means of the states were initialized with the peak in the histogram. HMM used here was continuous density HMM.
- With HMM on MFCC, experimentation was done ranging the number of states from 8 to 12, approximating it to be equal to number of 'swara' in Indian music. Each state again was mixture of three Gaussians. HTK toolkit was used for implementing and evaluating the performance of HMM.
- Matlab's tree bagger toolkit was used to implement Random Forests. Experimentation was done with various number of trees in random forests to reach the final decision.

4.5.8 Inferences

- Raw MFCC did not prove useful with either of HMM and GMM.
- Since lyricists write lyrics with a lot of emphasis on feeling, use of lyrics for classification proved fruitful.
- Summarized MFCCs defined the genre of the songs and choice of a song is dependent on taste of user. So, for some users, like user 3, MFCCs were outperforming other two features.

- HMM on normalized pitch definitely performed better than actual pitch but still was not outperforming MFCC and lyrics. The reason could be high occurrence of lower ′sa′, ′pa′ and upper ′sa′, i.e. high occurrence of pitch values 0, 600 and 1200 which was creating confusion between two classes. This is also evident from the fact that when number of songs were less, HMM on pitch was performing well.
- For some of the users, fusion of scores was proving helpful with Random Forests, while for some users, confusion was getting created on score fusion. With weighted sum of scores, performance was poor than Random Forests for all the three users. This result was expected also as Random Forest is widely known classifier for such an application while with weighted score, it was pretty normal technique.
- The number of states corresponding to best performance in HMM of pitch gave some relation between number of clusters in histogram of pitch and number of states chosen.
- Early recommendation of songs gave good results with pitch and lyrics but not performed well with MFCCs.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

As the topic of the report suggest, recommendation can be either in the form of classified genres where a user can choose what genre he is interested in listening or the recommendation can also be based on history of likes or dislikes that the user had. Both of the problems were tackled and fairly good results were obtained on both of them. As seen in recommending songs based on likes and dislikes, it was difficult to classify with direct song matching technique as different features were working for different users with pre processing in the features involved. So, all the three features, namely, melody, genre and lyrics, must be taken into account to give a proper recommendation and also some better way to combine these features must be devised.

Experimental results clearly show that model based techniques with appropriate choice and processing of features, can improve results significantly. Experiments began with successful classification of seven genres but soon the trouble part was met when no direct method that was thought was working well for us, although some gave fair results. Soon, experimentation on lyrics was attempted and it was felt that lyrics play a big role in choice of songs, knowingly or unknowingly. Apart from this, summarized MFCC were playing an important part if user's choice is dependent on timbre of songs. It was found that MFCC and lyrics were outperforming melody in most cases when number of training songs were on larger side but lyrics and melody outperformed MFCCs in case of early recommendation. Later, an attempt was given to fuse the scores of the three classifiers and for one of the user, performance improved after fusing the scores with random forest.

5.2 Criticism and Possible Future Work

- Recommendation given is in the form of positive or negative response. This can be improved by classifying songs into different ratings based on user′s preference.
- Some other features can be explored like rhythm in music apart from features currently worked.
- Improved way to fuse the features must be worked upon to get better recommendation, otherwise it is difficult to choose the feature for recommendation.
- Some better way to decide the number of mixtures in GMM in case of early recommendation can be thought.

APPENDIX A

CLASSIFICATION AND FEATURE EXTRACTION TECHNIQUES USED

A.1 Classification Techniques

A.1.1 Gaussian Mixture Model

In statistics, a mixture model is a probabilistic model for representing the presence of subpopulations within an overall population. As the name indicates, train data is modeled as a mixture of Gaussian functions. This is basically linear superposition of Gaussians in the form

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \tag{A.1}$$

The biggest advantage of clustering using Gaussian mixture model over traditional clustering method is that each point can be represented more than one cluster. So, it is kind of best way to represent multi modality in a system. Three parameters in Gaussian mixture model, namely Σ (covariance), μ (mean) and π (component proportion) are estimated using expectation maximization (EM) method. EM algorithm has applications in a wide variety of tasks and has been used in the context of various machine learning models. The assumption that Gaussian mixture models take is that all the points are identically and independently distributed. So, the log of likelihood of Eqn (1) over all the points is given by

$$lnp(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} ln \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \tag{A.2}$$

Now the parameters of the model are estimated by maximizing this log likelihood function over an iterative procedure. This is kind of chicken and egg problem where we first input a set of parameters to the model and in return we get an improved set of parameters. This is done until the point of convergence. The initial set of parameters given to the problem are not selected randomly but are generally the output of k-means clustering. A sample mixture of Gaussians can be viewed in figure A.1.

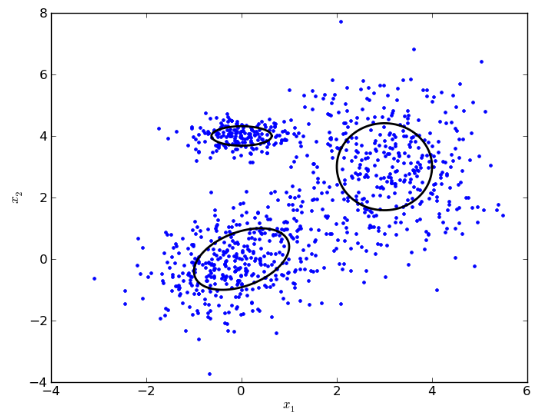


Figure A.1: A sample mixture of Gaussians

A.1.2 Hidden Markov Model (HMM)

Most of the classifiers used in machine learning do not consider the sequence information present in data and they just consider the data as static. The problems that have inherent temporality in them and consists of process that unfolds in time, HMMs have found great use in such problems.

The model in HMM is represented by a three-member tuple involving the initial state probabilities, state transition probabilities and observation symbol probabilities. Initial state probabilities define the probability of starting from a particular state. State transition probabilities define the probability of transition from one state to another. Observation probabilities define the probability of observing a symbol from every state.

Three major issues of hidden Markov model are:

- Testing
- Finding optimal state sequence
- Training

The good news here is that all the three problems are solved and if we have set of observation sequence, then we can define all the three tuples of Hidden Markov Model. We used HTK toolkit for training HMM devised by Young and Young [1994].

Two kinds of HMMs used today are Discrete HMMs and continuous density HMMs. In discrete HMMs, observation probabilities in each state are discrete but on the other hand, in continuous density HMM, observation probability is a Gaussian or mixture of Gaussian with its usual parameters.

A.1.3 Support Vector Machine (SVM)

Support vector machine is one of the most important classifiers in machine learning today. The discussion on support vector machine started after the arrival of perceptron algorithm which focused on obtaining a separating hyperplane between two classes which are linearly separable. But the difference is, Support Vector

Machine instead of obtaining just a hyperplane tries to obtain a maximal margin hyperplane. This means that the hyperplane is situated exactly between corner points (also called support vectors) of the two classes. Hyperplane that we need in support vector machine is such that distance of obtained hyperplane from the support vectors of both the classes is maximum from both the classes.

But in a real world, there is hardly any data where the two classes are linearly separable. So, if the data of two classes is overlapping, there is a provision of error in support vector machines. The amount of error tolerance can be specified while training the model. Although in a real world, Support Vector Machines are hardly used in actual data space. This is because of the fact that real world data is not linearly separable. Also as per Cover's theorem, *A pattern recognition problem when cast in higher dimensional space is more likely to be linearly separable than in lower dimensional space.* So, the data is first moved into kernel space which is higher dimensional space or even can be infinite dimensional space. Now in this space, a maximal separating hyperplane is found. Results obtained from support vector machine is highly dependent on parameters given to it. Two of the parameters given to it are kernel width and error tolerance.

A.1.4 Random Forest and Decision Trees

The decision tree is amongst one of the important classifiers in machine learning today. Based on the training data, this algorithm creates a tree where every node from root to leaf gives a decision of which class should the test data belong to. Choice of decision at every node is based on the statistical parameters like variance in the training data. Usually, training time for decision tree is huge, so it is not used when the dataset is large. If we somehow can reduce the size of dataset, then it can be used.

The decision tree is generally not used alone but is used in combination of many. Such a combination of decision trees is known as random forest. Decision given by most number of decision trees is considered to be the final decision.

A.2 Feature Extraction

A.2.1 Mel Frequency Cepstral Coefficients (MFCC)

In sound processing, the mel-frequency cepstrum is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up Mel Frequency Cepstrum (MFC). They are derived from a type of cepstral representation of the audio clip which is a nonlinear ”spectrum of spectrum”. The difference between the cepstrum and the MFC is that in the latter, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping allows for better representation of sound, like in audio compression. MFCCs are commonly derived as follows:

1. Take the Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

A.2.2 Melody (Pitch)

Melody extraction is the task of automatically estimating the fundamental frequency corresponding to the pitch of the predominant melodic line of a piece of polyphonic music. Melody extraction is a process of estimating when melody is present and when it is not. It is also a process of estimating correct pitch when the melody is present. The current pitch extraction algorithm used in the work is devised by Salamon *et al.* [In Press (2013].

REFERENCES

Breiman, L. and **A. Cutler** (2004). Random forests. URL `https://www.stat.berkeley.edu/~breiman/RandomForests/`.

Buitinck, L., **G. Louppe**, **M. Blondel**, **F. Pedregosa**, **A. Mueller**, **O. Grisel**, **V. Niculae**, **P. Prettenhofer**, **A. Gramfort**, **J. Grobler**, **R. Layton**, **J. VanderPlas**, **A. Joly**, **B. Holt**, and **G. Varoquaux**, API design for machine learning software: experiences from the scikit-learn project. *In ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2**, 121–167.

Burred, J. J. and **A. Lerch**, A hierarchical approach to automatic musical genre classification. *In in Proc. Of the 6 th Int. Conf. on Digital Audio Effects (DAFx*. 2003.

Cooper, M. and **J. Foote**, Automatic music summarization via similarity analysis. *In in Proc. Int. Conf. Music Information Retrieval, 2002*. 2002.

Downie, J. S., **A. F. Ehmann**, **M. Bay**, and **M. C. Jones**, *Advances in Music Information Retrieval*, chapter The Music Information Retrieval Evaluation eXchange: Some Observations and Insights. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-11674-2, 93–115. URL `http://dx.doi.org/10.1007/978-3-642-11674-2_5`.

Duda, R. O., **P. E. Hart**, and **D. G. Stork**, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

Dutta, S. and **H. A. Murthy**, A modified rough longest common subsequence algorithm for motif spotting in an alapana of carnatic music. *In Communications (NCC), 2014 Twentieth National Conference on*. 2014.

Ghias, A., **J. Logan**, **D. Chamberlin**, and **B. C. Smith**, Query by humming: Musical information retrieval in an audio database. *In Proceedings of the Third ACM International Conference on Multimedia*, MULTIMEDIA '95. ACM, New York, NY, USA, 1995. ISBN 0-89791-751-0. URL `http://doi.acm.org/10.1145/217279.215273`.

Hershey, J. R. and **P. A. Olsen** (2007). Approximating the kullback leibler divergence between gaussian mixture models.

HWEI-JEN LIN, H.-H. W. and **C.-W. WANG**, Music matching based on rough longest common subsequence. 2011.

Joachims, T., Advances in kernel methods. chapter Making Large-scale Support Vector Machine Learning Practical. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-19416-3, 169–184. URL `http://dl.acm.org/citation.cfm?id=299094.299104`.

MATLAB, *version 7.10.0 (R2014a)*. The MathWorks Inc., Natick, Massachusetts, 2014.

Pye, D., Content-based methods for the management of digital music. *In Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, volume 6. 2000. ISSN 1520-6149.

Salamon, J., **E. Gómez**, **D. P. W. Ellis**, and **G. Richard** (In Press (2013)). Melody extraction from polyphonic music signals: Approaches, applications and challenges. *IEEE Signal Processing Magazine*.

Salamon, J., **B. Rocha**, and **E. Gmez**, Musical genre classification using melody features extracted from polyphonic music signals. *In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2012. ISSN 1520-6149.

Scaringella, N., **G. Zoia**, and **D. Mlynek** (2006). Automatic genre classification of music content: a survey. *IEEE Signal Processing Magazine*, **23**(2), 133–141. ISSN 1053-5888.

Schnitzer, D., **A. Flexer**, and **G. Widmer** (2010). A fast audio similarity retrieval method for millions of music tracks. *Multimedia Tools and Applications*, **58**(1), 23–40. ISSN 1573-7721. URL `http://dx.doi.org/10.1007/s11042-010-0679-8`.

Shifrin, J., **B. Pardo**, **C. Meek**, and **W. Birmingham**, Hmm-based musical query retrieval. *In Proceedings of the 2Nd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '02. ACM, New York, NY, USA, 2002. ISBN 1-58113-513-0. URL `http://doi.acm.org/10.1145/544220.544291`.

Wikipedia (2004). Random forestsWikipedia, the free encyclopedia. URL `https://en.wikipedia.org/wiki/Random_forest`.

Young, S. and **S. Young** (1994). The htk hidden markov model toolkit: Design and philosophy. *Entropic Cambridge Research Laboratory, Ltd*, **2**, 2–44.

Zhu, Y. and **D. Shasha**, Warping indexes with envelope transforms for query by humming. *In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03. ACM, New York, NY, USA, 2003. ISBN 1-58113-634-X. URL `http://doi.acm.org/10.1145/872757.872780`.