CS4089 Project
Report

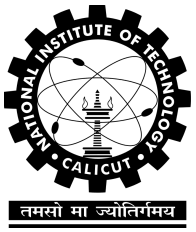
The Piper

Submitted in partial fulfillment of
the requirements for the award of the degree of
Bachelor of Technology
in
Computer Science and Engineering

Submitted by

Roll No	Names of Students
B130203CS	Aakanksha N S
B130236CS	Akshaye A P
B130611CS	Khot Nirant Prakash
B130004CS	Ritvik Vinodkumar

Under the guidance of
Mrs. Anu Mary Chacko



Department of Computer Science and Engineering
National Institute of Technology Calicut
Calicut, Kerala, India – 673 601

Winter Semester 2017

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person, nor material which has been accepted for the award of any other degree or diploma of the university or any other institute of higher learning, except where due acknowledgement has been made in the text.

Place: NIT Calicut
Date : 4 May 2017

Signature:

Name : Aakanksha N S
Roll No : B130203CS

Signature:

Name : Akshaye A P
Roll No : B130236CS

Signature:

Name : Khot Nirant Prakash
Roll No : B130611CS

Signature:

Name : Ritvik Vinodkumar
Roll No : B130004CS

CERTIFICATE

This is to certify that the project entitled **"THE PIPER"** *submitted by* **AAKANKSHA N S B130203CS**, **AKSHAYE A P B130236CS** , **KHOT NIRANT PRAKASH B130611CS** *and* **RITVIK VINODKUMAR B130004CS** *to National Institute of Technology Calicut towards partial fulfilment of the requirements for the award of Degree of Bachelor of Technology in Computer Science Engineering is a bonafide record of the work carried out by them under my supervision and guidance.*

Mrs. Anu Mary Chacko
Project Guide

Dr. Saidalavi Kalady
Head of the Department

Office Seal

Abstract

Automatic Music Genre Classification and Lyrics Identification can serve as an important tool in a lot of music applications. This project explores the use of machine learning approach for genre classification and audio fingerprinting approach for lyrics identification of a song. In this project, we make a comparison between four machine learning algorithms namely k nearest neighbours, Random Forests, Support Vector Machines and Neural Networks, used for genre classification and compare their accuracies to determine Neural Networks to be the best algorithm among them. Neural Networks has then been incorporated into the application to perform the genre classification functionality. A detailed procedure for audio fingerprint generation and matching has been undertaken, which is used to identify a particular song, after which an online retrieval of it's lyrics is done. The design of the project including the data-set used, various algorithms tested, the input and the output of the project, and a step-wise procedure that was followed to make the application has been discussed in the report. The application fetches the genre and identifies the lyrics in most cases, however if the recording is too noisy, it might return the incorrect genre and fail to identify the lyrics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Classification of music into different genres is essentially, a human classification and entirely subjective. There are no strict restrictions as to what classifies a specific genre. However, for the purpose of Music Information Retrieval (MIR), an alternative approach for music genre classification has been suggested, where genre has been classified according to specific characteristics of a song. To implement a system that identifies lyrics of a particular song, we make use of audio fingerprinting to obtain the information about this audio. This information is then used to retrieve it's lyrics. These two systems for getting the genre and lyrics are packaged into an android application.

1.1 Problem Statement

To predict the genre of a given song and identify it's lyrics.

1.2 Motivation

People are interested in the lyrics of a song they listen to. Hence, quick retrieval of lyrics is favorable to music lovers. Moreover, every individual has his/her own specific set of preferences when it comes to genre and hence genre classification becomes an important part of any music application. Genre and lyrics together give a sense of the mood of the song. Few applications that this project could be used for have been identified:

1.2.1 Smart Graphic Equalizers

Equalizers offer different options to customize the music or audio that is playing on your device. Genre classification can help in automatically tweaking the equalizers of a music application to suit the song that is playing.

1.2.2 Cleaning up song meta data

Many songs lack the meta data needed to be searched and accessed by users. The genre field in the song meta data can be updated by the application once it has been identified which will help songs to be better searched by genre.

1.2.3 Music Recommender

Given the activity of a user on a music application, a music recommender can suggest a list of songs or artists that fits their profile. Such a recommender can use a genre classifier to identify the genres that the user most frequently listen to and then incorporate the data into the recommender algorithm.

1.2.4 Song recognition

Often people hear an unfamiliar song that they like, and would like to know the name of the song. This can be achieved using audio fingerprinting.

1.2.5 Convenient lyrics retrieval

One can lookup the lyrics of a song online if they know the name of the song. However, for an unknown song this is not convenient. The fingerprinting approach can be used to identify the lyrics of an unknown song as well as to fetch the lyrics of a known song faster.

Chapter 2

Literature Survey

2.1 Genre classification

Automatic rather than manual classification of audio signals into musical genres has been explored in detail by Tzanetakis et al.[6], which talks about the various factors that contribute in deciding the genre like timbral texture, rhythmic content and pitch content. It concluded that automatic classification is possible with satisfactory accuracy. Li et al.[7] provides detailed comparisons of various factors that affect automatic musical genre classification performance.

2.2 Machine Learning Approaches

Kotsiantis et al.[8] talks about various supervised machine learning algorithms. Miguel Francisco et al.[1] deal with music genre classification in particular and give a comparison between the various types of feature extraction techniques that can be used to predict the genre. The paper concluded that the MFCC feature extraction approach gives the best results. It also mentions that high number of classes reduces accuracy.

Jensen et al.[9] evaluates mel frequency cepstral coefficient (MFCC) estimation techniques. The performance of this method is evaluated in genre classification using a probabilistic classifier based on Gaussian Mixture models. Han et al.[3] introduces a new algorithm of extracting MFCCs for speech recognition. The new algorithm reduces the computation power by 53% compared to the conventional algorithm. However, new method reduces the accuracy to some extent.

Breiman[2] gives an overview of Random forests classification technique. Peterson[10] explains the k nearest neighbour algorithm and it's implemen-

tation. The neural networks algorithm has been explained by Singh et al.[4]. A comparison between accuracies of machine learning algorithms performing genre classification using MFCCs has been discussed by Haggblade et al.[11]. It concludes that the four best algorithms for genre classification are kNN, Neural networks, SVM and Random Forests.

Following design decisions were made for genre classification based on the literature survey:

- The MFCC approach has been employed for feature extraction
- Only few highly distinct genre classes have been selected
- The conventional method for MFCC computation has been followed
- kNN, Neural Networks, SVM and Random Forests are the machine learning algorithms that have been tested and compared to determine the best approach for the project.

2.3 Audio Fingerprinting

Cano et al.[12] gives a review of audio fingerprinting i.e. it's meaning, potential uses, advantages and different algorithms that can be used. Haitsma et al.[13] mention that the advantage of using fingerprints instead of the multimedia content itself is three-fold:

- Reduced memory/storage requirements as fingerprints are relatively small
- Efficient comparison as perceptual irrelevancies have already been removed from fingerprints
- Efficient searching as the dataset to be searched is smaller

Cano et al.[16] talks about the various steps involved in audio fingerprint generation and matching. Shazam, an application closely related to the lyrics identification part of this project, has been discussed in detail by Wang et al.[17]. An extension to the project has been discussed by Mesaros et al.[18]. It deals with an implementation of automatic recognition of lyrics in singing. Though the results are far from practical, it is an interesting area of research.

Chapter 3

Genre Classification

According to Music Information Retrieval(MIR), genres are classified based on the fact that members of a particular genre share certain characteristics such as instrumentation, rhythmic structure and pitch content of the song. This is where machine learning comes into play. Machine learning can be employed to use these characteristics of a song to predict its genre. The dataset used for doing this has been detailed below. The two main steps for machine learning have also been detailed: Feature Extraction and Classification.

3.1 Dataset

The GTZAN dataset was used, which consists of 1000 audio tracks, each 30 seconds long in .au format. There are 10 genres represented, each consisting of 100 tracks. Since increase in the number of classes leads to reduced accuracy of the classifier as mentioned in [1], the four most distinct genres namely metal, jazz, classical and pop were selected for use. Hence,the dataset comprises of these four genres. A 90-10 split was used, where 90% of the data was used as the training set and 10% for the test set.

3.2 Feature Extraction [1]

MFCCs are one of the most widely used feature extraction methods in music-related fields for machine learning. They represent a set of short term power spectrum characteristics of audio. These are based on cepstral representation (a nonlinear "spectrum of a spectrum), but unlike the cepstrum, the Mel-Frequency Cepstrum uses frequency bands that are equally spaced over the mel-scale, and therefore are expected to produce a better representation of

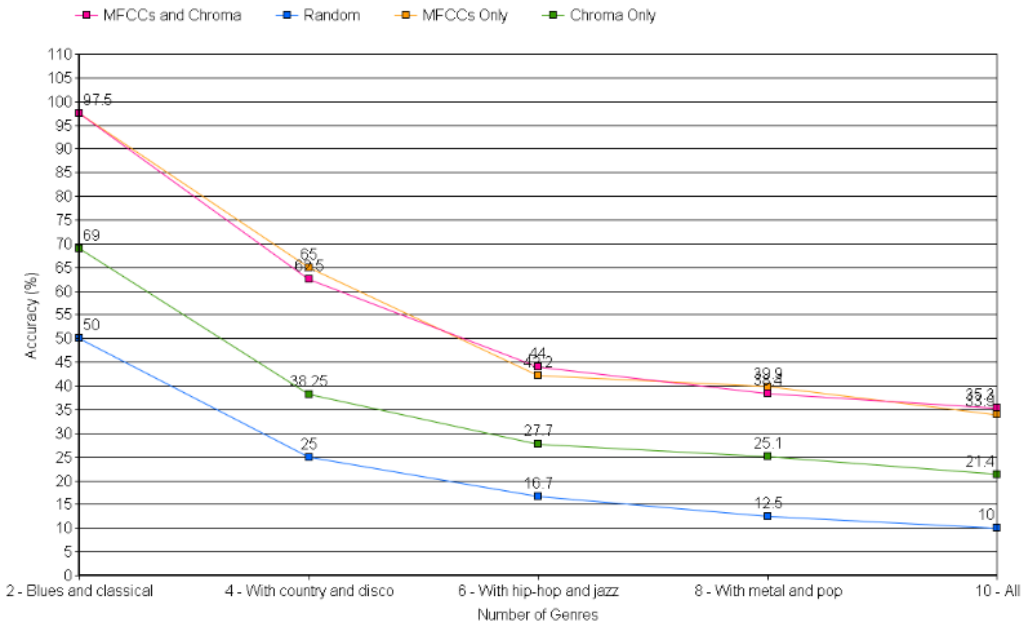


Figure 3.1: Number of Genres vs Accuracy [1]

sound, closer to actual human perception. The method followed to extract these features is mentioned below:

- Dividing the signal into several short frames. This step ensures that the audio signal is kept constant.

- Calculating the periodogram estimate of the power spectrum for each frame. This is done to know the frequencies present in the short frames.

- Pushing the power spectrum into the mel filterbank and summing up the energy collected in each filter. This lets us know the amount of energy existing in the various frequency regions.

- Calculating the logarithm of the filterbank energies computed in the previous step.

- Calculating the Discrete Cosine Transform (DCT) of the result. It decorrelates the filterbank energies with each other. The first 13 DCT coefficients are retained and the higher ones are removed as they are capable of introducing errors by representing changes in the filterbank energies.

A Java code was written to perform the above mentioned steps which takes as input a file in .wav format. This code was then used to extract MFCC features of the audio files in our dataset and write them into a single .csv file.

3.3 Machine Learning and Selection of the Best Algorithm

The next step was to select the best machine learning algorithm. For this, Knime (an open source tool for performing data analytics) was used. Various classification techniques including k-nearest neighbor (k-NN), k-means, SVM, neural networks, random forests [11] and regression analysis were adopted. Given below is a brief description of the four best classification techniques:

SVM

The support vector machine is a supervised classification system that finds the maximum margin hyperplane separating two classes of data. If the data are not linearly separable in the feature space, as is often the case, they can be projected into a higher dimensional space by means of a Mercer kernel, K(.).

The space of possible classier functions consists of weighted linear combinations of key training instances in this kernel space. The SVM training algorithm chooses these instances and weights to optimize the margin between classifier boundary and training examples.

Table 3.1: **SVM**

	Actual				
Predicted		**Jazz**	**Classical**	**Metal**	**Pop**
	Jazz	2	0	0	0
	Classical	7	10	0	0
	Metal	1	0	10	0
	Pop	0	0	0	10
	Accuracy	***20%***	***100%***	***100%***	***100%***

Overall Accuracy
80%

SVM gives the least accuracy for Jazz among all the four algorithms with 20% whereas it gives 100% accuracy for Classical, Metal as well as Pop, which is shown in Table 3.1.

k-NN

The k-nearest neighbors (k-NN) is effective considering its ease of implementation. It is an instance-based learning algorithm. The basis consists of i) determining the distance from query instance and all training samples; ii) sort these distances, using Euclidean measure, from the smallest and choose

the k-nearest neighbours; iii) gather the categories of its nearest neighbours, and iv) predict values for the query instances based on the majority of these categories.

Table 3.2: **k-NN**

		Actual				
		Jazz	**Classical**	**Metal**	**Pop**	***Overall Accuracy***
Predicted	**Jazz**	6	1	0	0	
	Classical	3	8	0	0	
	Metal	0	1	10	0	***85%***
	Pop	1	0	0	10	
	Accuracy	***60%***	***80%***	***100%***	***100%***	

As shown in Table 3.2, k-NN gives decent accuracies of 60% and 80% for Jazz and Classical respectively, whereas 100% for both Metal and Pop, making up the overall accuracy 85%.

Random Forests

Random forests were introduced as a classification tool and have been tested to be very powerful to do high-dimensional classification tasks. In this algorithm, prediction is obtained by aggregating classification or regression trees each constructed using a different random sample of the data, and choosing splits of the trees from subsets of the available predictors, randomly chosen at each node.

Table 3.3: **Random Forest**

		Actual				
		Jazz	**Classical**	**Metal**	**Pop**	***Overall Accuracy***
Predicted	**Jazz**	7	4	1	0	
	Classical	2	6	0	0	
	Metal	0	0	9	0	***80%***
	Pop	1	0	0	10	
	Accuracy	***70%***	***60%***	***90%***	***100%***	

The accuracy for Random Forests has been shown in Table 3.3. It gave the least accuracy for Classical among all four algorithms (60%), whereas 70% for Jazz, 90% for Metal and 100% for Pop, making up the overall accuracy 80%.

Neural Networks

Multilayer perceptron neural networks (MLP) are effective tools used for a wide range of purposes from pattern classification over modeling to implementations for control. A multilayer perceptron(MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate output. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network. MLP is a modification of the standard linear perceptron, which can distinguish data that is not linearly separable.

Table 3.4: **Neural Networks**

	Actual				
Predicted		**Jazz**	**Classical**	**Metal**	**Pop**
	Jazz	6	0	0	0
	Classical	4	10	0	0
	Metal	0	0	10	0
	Pop	0	0	0	10
	Accuracy	***60%***	***100%***	***100%***	***100%***

Overall Accuracy
90%

Neural Networks gave a 60% accuracy for Jazz as well as 100% for Classical, Metal and Pop, making up its overall accuracy 90%.

As we can see from Table 3.4, Neural networks turned out to be the most accurate approach for genre classification. Hence, this technique was adopted in the implementation of our prototype. Hence, using Knime (an open source tool for performing data analytics) and testing out various classification techniques including k-nearest neighbor (k-NN), k-means, SVM, neural networks, random forests and regression analysis, Multi Layer Perceptron neural networks was determined as the best method based on overall accuracy.

3.4 Detailed design of Neural Networks

3.4.1 Multi Layer Perceptron

A Neural Network (NN) consists of several components interconnected and organized in layers. These components are called Artificial Neurons (AN). An AN accepts a number of inputs and outputs the class it believes the item

belongs to. Each AN is itself a classifier, only a simpler one whose accuracy has limitations when used for complex problems (non-linear problems). Hence, we interconnect a number of ANs to form a NN in order to overcome the limitations of simple isolated classifiers. Each AN can be a perceptron or a logistic regression unit, which are themselves standalone classification algorithms. NNs with multiple layers of perceptrons are powerful classifiers in that we can use them to model very complex, non-linear classification problems.

3.4.2 Structure

A NN is organized in layers of interconnected ANs: the first layer is called input layer, the last layer is called output layer and all the layers in between are called hidden layers.

The input layer consists of a number of ANs that depends on the number of input features. Features are engineered to describe the class instances. In our task, the features are the MFCC values.

The output layer consisting of a number of ANs equal to the number of classes in the problem. When given a new, unseen example, each AN of the output layer assigns a probability that this example belongs to each particular class, based on its training.

Between the input and output layers, there may be several hidden layers of ANs, but for many problems one or two hidden layers are enough. More units in a layer allow a greater level of detail at the same level of abstraction. However, too few hidden nodes can lead to under-fitting and too many hidden nodes can lead to over-fitting.

From the data set, there are thirteen MFCC values that are used to predict the genre. The algorithms tries to predict one out of four different genres (Jazz, Classical, Metal, Pop). Hence, 13 nodes were used in the input layer of the neural network and 4 nodes in the output layer. One hidden layer consisting of 13 nodes is used. As a result, the neural network makes use of a total of 3 layers.

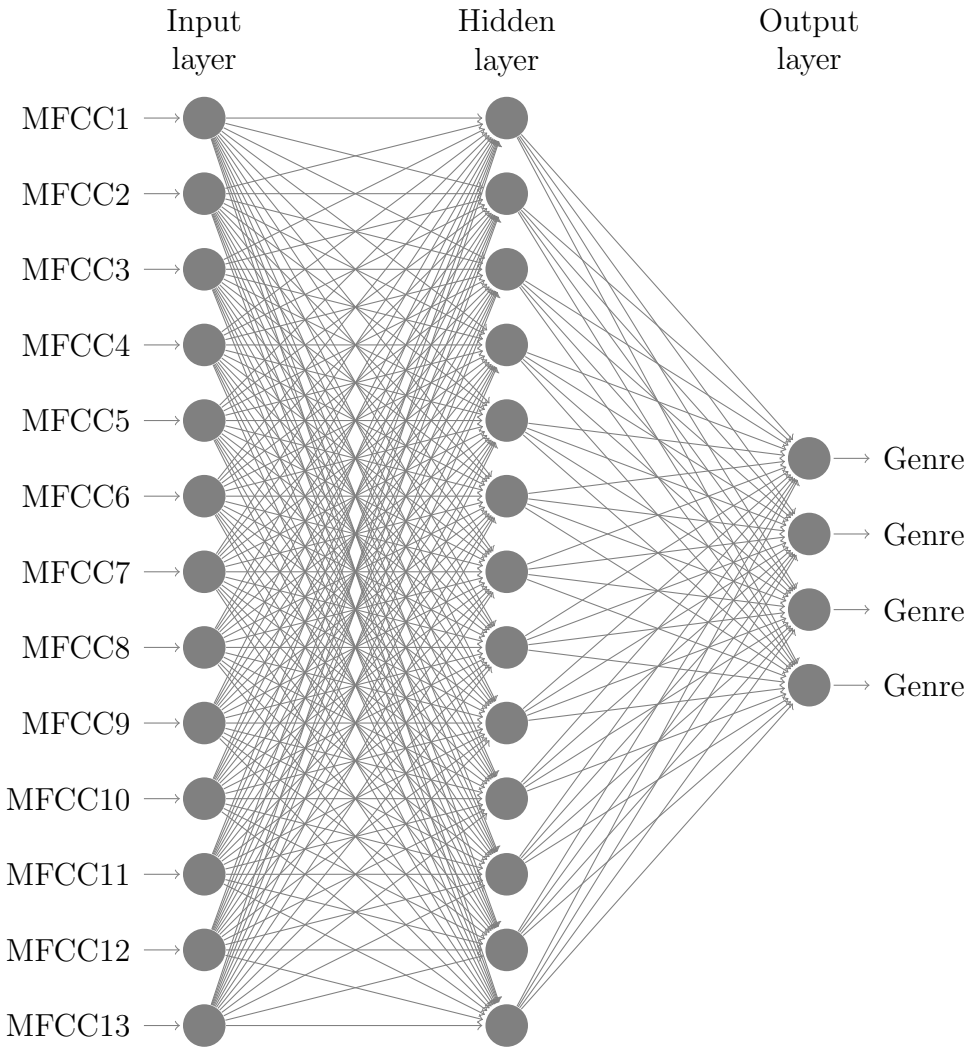


Figure 3.2: Structure of the Neural Network

3.4.3 Training

Forward Propagation

All connections between ANs in the network are given a weight. These weights represent the importance of the respective inputs to the overall output. The values of weights between each layer were initialized using randomization to values within specific bounds. This ensures that nodes will not update to the same value repeatedly. Changing the value of these weights essentially allows the network to arrive at it's desired output.

In forward propagation, one forward pass is made through the network for

the training set, to arrive at the corresponding output set, i.e. get an initial hypothesis for all the training examples in the training set. The difference between this initial output and the desired output is represented by the Cost Function (output error).

Backpropagation

A neural network learns with the use of feedback. This feedback process is called *Backpropagation*. This involves comparing the output a network produces with the output it was meant to produce, and using the difference between them to modify the weights of the connections between the units in the network, working from the output units through the hidden units to the input units, going backwards. *Backpropagation* is essentially about minimizing our cost function.

Optimization Function

An optimization function to minimize the cost function with the weights is used. These new values for weights are obtained using *fmincg* function. It has a similar goal as gradient descent which is to optimize the solution's vector space towards a global minimum. The global minimum is that theoretical solution with the lowest possible error (minimum cost function). However, *fmincg* is a function which works similarly to the widely used *fminunc* function, which is an advanced optimizer that is able to train our cost functions efficiently as long as we provide them with the gradient computations. *Fmincg* is preferred because it is more efficient at doing gradient descent for especially complex hypotheses. This function was repeated through 100 iterations to obtain optimized weights and minimized cost function.

In time, *backpropagation* causes the network to learn, reducing the difference between actual and intended output to the point where the two exactly coincide, so the network figures things out exactly as it should.

3.4.4 Prediction

Once a neural network is trained to a satisfactory level it may be used as an analytical tool on other data. Optimized values for the weights of the neural network are obtained. It reaches a point where you can present it with an entirely new set of inputs it's never seen before and see how it responds. To do this, the user no longer specifies any training runs and instead allows the network to work in forward propagation mode only. New inputs are presented to the input pattern where they filter into and are processed by

the middle layers as though training were taking place, however, at this point the output is retained and no backpropagation occurs. The output of a forward propagation run is the predicted model for the data which can then be used for further analysis and interpretation.

These new weights, along with MFCC values, are used to predict the genre of an input song. A training set accuracy of almost 95% and test set accuracy of more than 82% were obtained using the optimized weights.

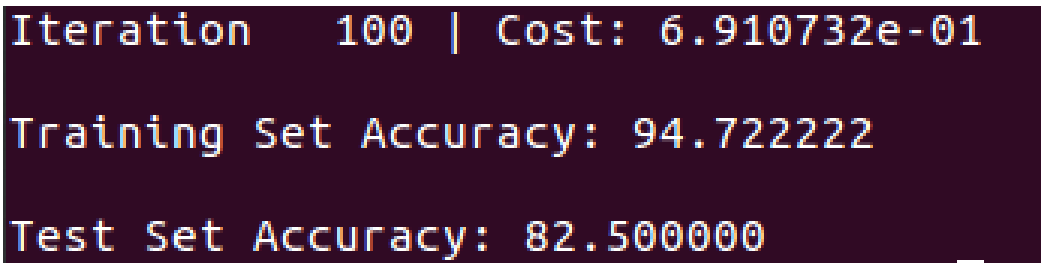


Figure 3.3: Accuracies output for test and training set

Chapter 4

Lyrics Identification

For identifying the lyrics, a fingerprint of the input song is generated, compared with a database of fingerprints to identify the name of the song, using which lyrics of the song are obtained. Identifying lyrics using the inherent waveforms of a song is difficult and inefficient. Instead, we make use of Audio Fingerprinting. An audio fingerprint is a compact content based signature that summarizes an audio recording [12]. This information about the audio is then used to retrieve it's lyrics.

4.1 Dataset

Lyrics identification is a matching problem rather than a classification problem. Contrary to genre classification, to have an effective lyrics identification application, a very huge dataset is required. Hence an online database of audio fingerprints that enables searching provided by ACRCloud has been used. ACRCloud provides 40 million music fingerprints of worldwide music and regional music. The major languages of songs include English, Chinese, Spanish, Hindi etc.

4.2 Fingerprint Generation

Audio Fingerprinting is a process which takes in an audio signal, analyzes it, generates a digital summary of the signal which can be further used for the signal identification or a quick matching of similar items from a database. It basically creates a short detail of the signal in a limited number of bits. Fingerprinting is used in a lot of practical applications like identifying songs, melodies, tunes; Broadcast Monitoring, Video File Identification, etc.

$$F(x) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi(x\frac{n}{N})}$$

Figure 4.1: Discrete Fourier Transform

4.2.1 Discrete Fourier Transform

The time domain signal represents the amplitude change of the signal over time. Any signal in the time domain can be represented in the frequency domain by simply giving the set of frequencies and amplitude corresponding to each sinusoid that makes up the signal. The frequency domain acts as a fingerprint of the time domain signal in many ways. Hence, there is a need to convert the time domain audio signal into frequency domain.

Discrete Fourier Transform (DFT) is used to get the frequency representation of the audio signal. The DFT is a mathematical methodology which converts a finite list of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids, ordered by their frequencies. The numerical algorithm used for the calculation of DFT is called Fast Fourier Transform (FFT).

A disadvantage of taking FFT over the whole song is that information about the time of occurrence of each frequency is lost. To overcome this, the song is divided into parts or 'chunks' and each chunk is transformed separately (FFT is taken on each chunk).

4.2.2 Peak Finding

In one song itself, the range of strong frequencies might vary between low C - C1 (32.70 Hz) and high C - C8 (4,186.01 Hz), which is a huge interval to cover. So instead of analyzing the entire frequency range at once, several smaller intervals called samples are chosen, based on the common frequencies of important musical components and each of them is analyzed separately. In this project, 40 Hz - 80 Hz and 80 Hz - 120 Hz are the ranges used for the low tones , and 120 Hz - 180 Hz and 180 Hz - 300 Hz for the middle and higher tones.

For each chunk of the song, the frequency with highest magnitude is identified.

4.2.3 Hashing

Hashing is the transformation of a list of characters or integers into a usually shorter fixed-length value or key that represents the original list. Hashing is used to index and retrieve items in a database because it is faster to find the item using the shorter hashed key than to find it using the original value.

In this project, for each chunk, if p1, p2, p3, p4 are considered as the peaks of the four ranges, the hash value H of the chunk is given by:

H = (p4-(p4 % FUZ_FACTOR))*100000000+(p3-(p3 % FUZ_FACTOR))*100000 +(p2-(p2 % FUZ_FACTOR))*100+(p1-(p1 % FUZ_FACTOR))

This forms the signature for a particular chunk of the song, and this signature becomes part of the fingerprint of the song as a whole. The fuzz factor denotes the degree of noise and other disturbances in the recording and it arises since the recording is not done in perfect conditions.

4.3 Fingerprint matching

To identify the input song, first it's fingerprint is generated. Then, a search is performed on the database for matching hash tags. Many of the hash tags will correspond to multiple songs. Each time a hash tag is matched, the number of possible matches gets smaller, but it is likely that this information alone will not narrow the match down to a single song. So there is one more thing that is needed to check with the music recognition algorithm, and that is the timing. The sample recorded can be from any point in the song, so simply matching the timestamp of the matched hash with the timestamp of the sample does not work. However, with multiple matched hashes, the relative timing of the matches can be analyzed, and therefore increase the certainty. Taking i1 and i2 as moments in the recorded song, and j1 and j2 as moments in the song from database, it can be said that we have two matches with time difference match if:

RecordedHash(i1) = SongInDBHash(j1) AND RecordedHash(i2) = SongInDB-Hash(j2) AND abs(i1 - i2) = abs (j1 - j2)

This gives flexibility to record the song from the beginning, middle, or end.

Finally, it is unlikely that every single moment of the song recorded will match every corresponding moment of the same song in the library, recorded in the studio. The recording could include a lot of noise that introduces some

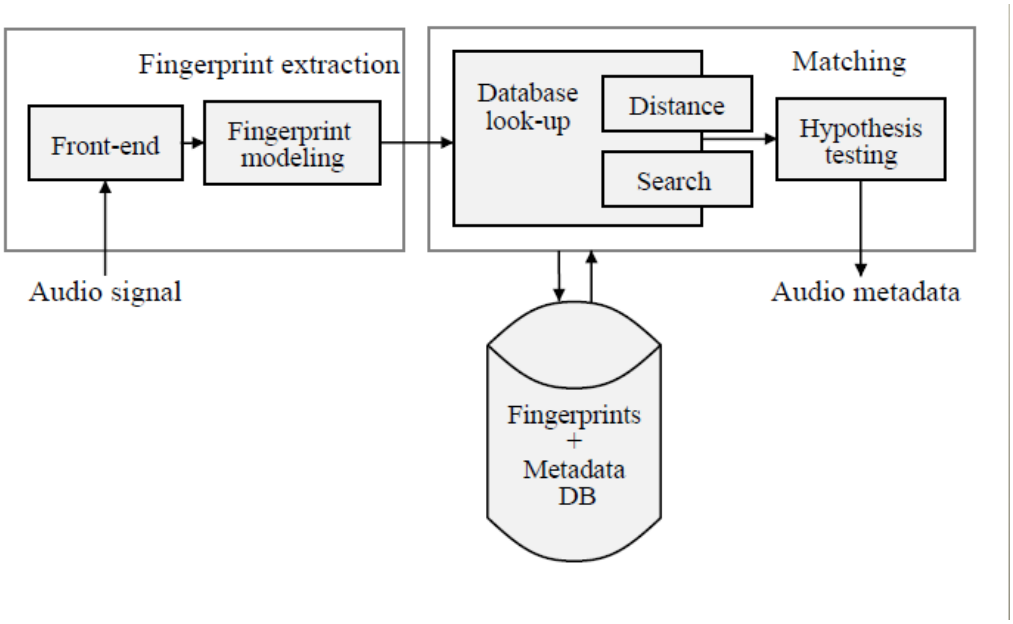


Figure 4.2: Audio recognition using fingerprinting

error in the matches. So instead of trying to eliminate all but the correct song from our list of matches, at the very end, all the matched songs are sorted in descending order of likelihood, and the first song on the ranking list is selected.

Hash Tag	Time in Seconds	Song
30 51 99 121 195	53.52	Song A by artist A
33 56 92 151 185	12.32	Song B by artist B
39 26 89 141 251	15.34	Song C by artist C

Figure 4.3: Sample audio fingerprint generation and matching

Chapter 5

Application - The Piper

To demonstrate that the above detailed theories have practical potential, a prototype was first developed as a Proof of concept. The prototype was a Java application that could predict the genre of any song given as input. An Android application with full capabilities was further implemented that can identify the lyrics and predict the genre of any song.

5.1 Preprocessing

All audio files are converted into .wav format to make it easier to read these files and perform feature extraction. For this, a Java code has been written using the Java Audio System library to convert audio files of any format to .wav format. This code was then used to convert the files in the dataset (which were in .au format).

5.2 Prototype for Genre Classification

An Java application was developed that does the following:

- Take as input an audio file in any format
- Converts it into .wav format
- Extract MFCC features of the wav file and writes to a .csv file
- Classify it using the command-line mode of Knime
- Extract the genre from the .csv output of Knime

5.3 Android Application

An Android application to predict the genre and identify the lyrics of a song using the processes mentioned before was developed. The application can successfully record a song and predict it's genre and lyrics, provided the recording is clear enough. The user also has an option to select a song from phone memory as an input to the application.

5.3.1 Design

The application allows the user to choose an input audio either by recording a song played externally or by choosing from the existing audio files in the android device. The user is provided an option to preview the media that he/she has selected. The user can then proceed by clicking the proceed button, at which time the application processes the input audio for both genre classification and lyrics identification. On successful completion of song identification, the application shows the result screen where it displays the genre of the song as well as it's lyrics with the title of the song and it's artist.

If an audio input couldn't be identified for lyrics, then the application will not proceed to the next screen and will prompt the user to try again with a different input. If the input has been identified but the lyrics are not available, then it will display the title and artist of the song along with the message "Lyrics Not Available" on the result screen.

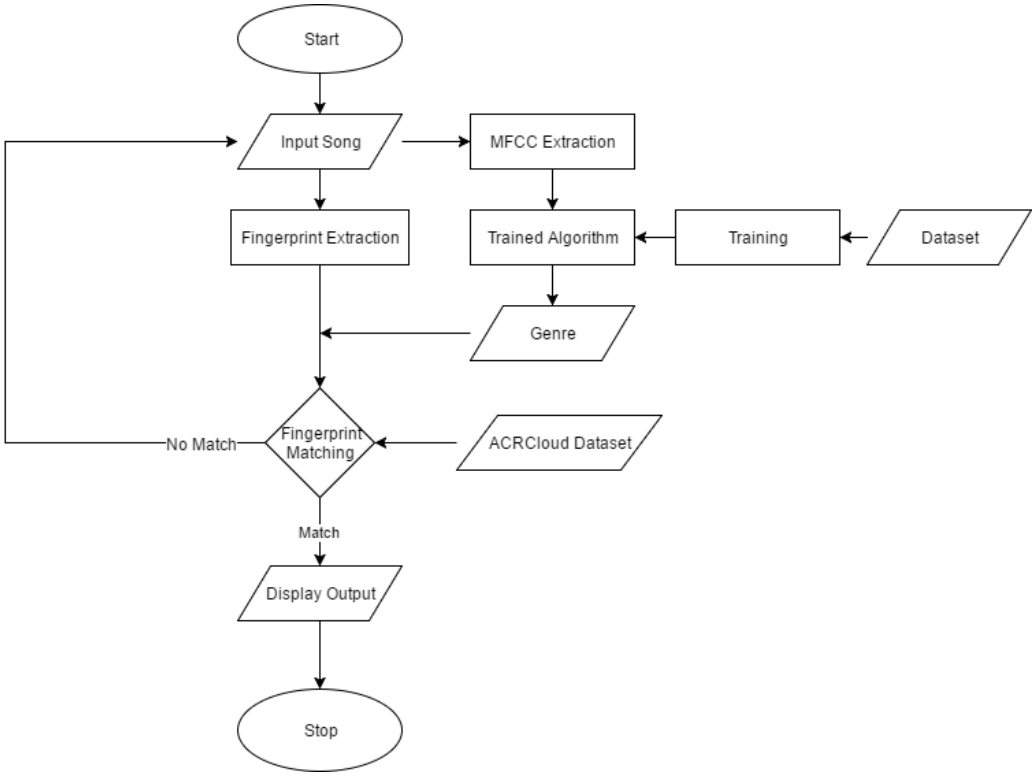


Figure 5.1: Flowchart depicting the flow of control in the application

Audio Recorder

The application includes a recorder which allows the user to record an externally played song for ten seconds. The audio is saved in a folder called "Piper", created by the app in the phone memory, in mp3 format. The recorder performs real-time noise suppression and other audio enhancers while recording, such that the final recorded audio is as noise-free as possible to enhance the process of music information retrieval.

Audio Preprocessing

The audio is fist converted to .wav format for easier conversion into raw data. If the input source is an audio file in the phone, then the audio is trimmed to 30 seconds if it's greater than that length. The 30 seconds of the audio is selected from the middle part of the song, rather than the first 30 seconds. This step is skipped for recorded input as it is only 10 seconds in length.

Genre Classification

The preprocessed audio input is now converted to raw byte data and this is used for MFCC extraction. These values are extracted by the steps mentioned earlier and the computed MFCC values are sent to the server for classification.

Neural Network

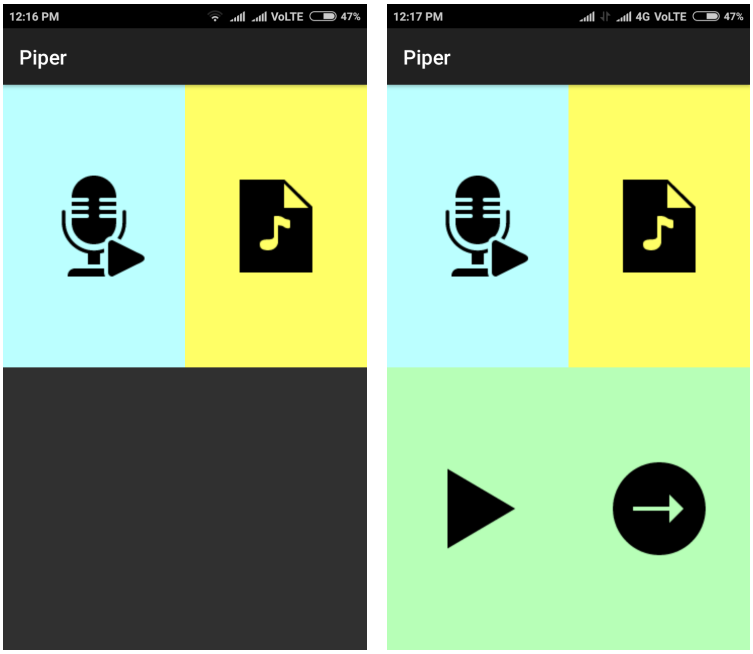
The neural network used for classifying genres are hosted on Athena server. After an initial training, the parameter values of the neural network are stored. Only these parameter values are required for classifying a new unseen input. The MFCC values are sent from the app to the server. These values are run through the neural network to obtain a prediction. The predicted genre is returned back to the app.

Song Identification

For identifying the song, the fingerprint of the preprocessed input audio is computed by the methods aforementioned. The fingerprint is computed at data points 5 seconds apart, each of 10 seconds of audio. The computed fingerprint values are sent to the ACRCloud for matching, using an HTTP POST API provided by the cloud providers. The details of the song, including it's title and artist, are sent as the result of the POST method.

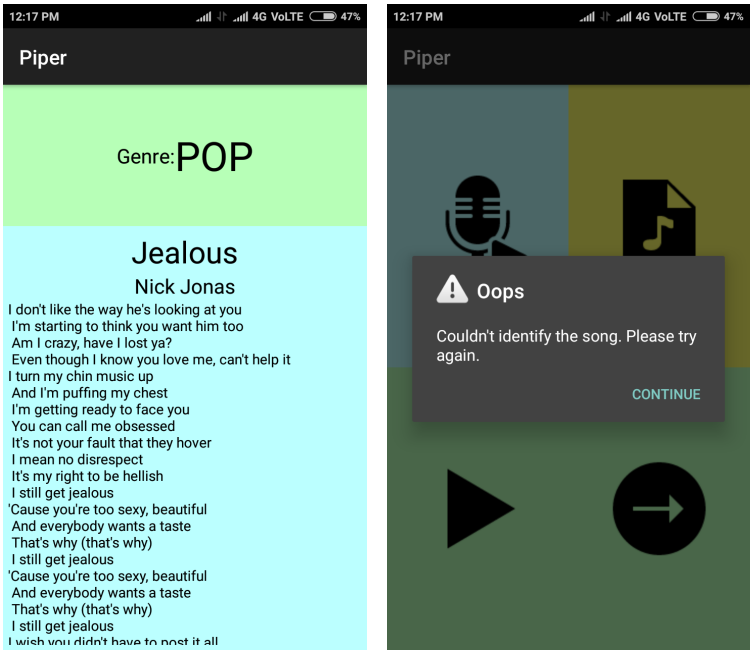
Lyrics Retrieval

The lyrics of the identified song are obtained from the website www.metrolyrics.com. MetroLyrics is a searchable lyrics database featuring 1000000+ song lyrics from 20000 artists. The URL format of the website follows a specific format and this format is taken advantage of to find the corresponding lyrics. For example, the webpage of the lyrics for a song of artist 'B', titled 'A' will be found at www.metrolyrics.com/A-lyrics-B, with white spaces between all words replaced by hyphens. Thus, the URL in this format is generated with the song title and artist's name. The webpage is obtained and the lyrics are parsed from the HTML code of the webpage.



(a) Initial User Interface

(b) UI after choosing/recording audio



(c) Result screen

(d) UI if song couldn't be identified

Figure 5.2: Screenshots of the application UI

Chapter 6

Conclusion

The project attempts to perform music genre classification using machine learning principles, lyrics identification using audio fingerprinting and creation of an Android application that serves for both the purposes of genre classification as well as lyrics identification.

Only four genres are chosen for classification as number of genres and accuracy are inversely proportional to each other. The classifier that gives the best accuracy is the Neural Networks approach (90%) whereas other classifiers give a maximum accuracy of 85%. As a result, Neural Networks is used for the prototype development.

The lyrics of a song are identified by creating a fingerprint of the song and matching it with an entry in a standard dataset. The fingerprint is basically a key which is created by applying a suitable hash function on certain parameters of the song. These parameters are obtained by converting the song into a frequency array using Fourier Transforms, and then finding peaks in frequency ranges for every part of the song, thus creating a unique set of numbers for every chunk.

The android application that can record a song for 10 seconds, cancel out noise and replay the song is developed. A song can be chosen from the internal memory of the device as well. The chosen or the recorded song is taken as the input and performs the functions of genre classification using Neural Networks which is implemented on the server. The process of fingerprint generation and matching takes place in the back end, and outputs the genre as well as the lyrics of the song.

Acknowledgments

We place on record and warmly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspiring guidance offered by our project guide, Mrs. Anu Mary Chacko in bringing this project and report to a successful completion.

We are grateful to Mr. Saidalavi Kalady, Head of the Department of Computer Science and Engineering, for permitting us to make use of the facilities available in the department to carry out the project successfully. We would also like to thank our panel memebers, Mrs. Saleena N, Mrs. Subhashini R and Mrs. Athira for their valuable feedback and suggestions. Last but not the least, we express our sincere thanks to all our friends, teachers and our parents who have patiently extended their help for accomplishing this undertaking.

Aakanksha NS (B130203CS)
Akshaye AP (B130236CS)
Khot Nirant Prakash (B130611CS)
Ritvik Vinodkumar (B130004CS)

May,2017
National Institute of Technology Calicut

References

[1] M. Francisco and D. M. Kim, "Music genre classification and variance comparison on number of genres," Tech. Rep.

[2] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] W. Han, C.-F. Chan, C.-S. Choy, and K.-P. Pun, "An efficient mfcc extraction method in speech recognition," in *2006 IEEE international symposium on circuits and systems*. IEEE, 2006, pp. 4–pp.

[4] Y. Singh and A. S. Chauhan, "Neural networks in data mining," *Journal of Theoretical and Applied Information Technology*, vol. 5, no. 6, pp. 36–42, 2009.

[5] J. Foote, "An overview of audio information retrieval," *Multimedia systems*, vol. 7, no. 1, pp. 2–10, 1999.

[6] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, Jul 2002.

[7] T. Li and G. Tzanetakis, "Factors in automatic musical genre classification of audio signals," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, Oct 2003, pp. 143–146.

[8] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," 2007.

[9] J. H. Jensen, M. G. Christensen, M. N. Murthi, and S. H. Jensen, "Evaluation of mfcc estimation techniques for music similarity," in *Signal Processing Conference, 2006 14th European*, Sept 2006, pp. 1–5.

[10] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[11] M. Haggblade, Y. Hong, and K. Kao, "Music genre classification," *Department of Computer Science, Stanford University*, 2011.

[12] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A review of audio fingerprinting," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 41, no. 3, pp. 271–284, 2005.

[13] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system." in *Ismir*, vol. 2002, 2002, pp. 107–115.

[14] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "Knime-the konstanz information miner: version 2.0 and beyond," *AcM SIGKDD explorations Newsletter*, vol. 11, no. 1, pp. 26–31, 2009.

[15] B. C. Zapata, *Android studio application development.* Packt Publishing Ltd, 2013.

[16] P. Cano, E. Batle, T. Kalker, and J. Haitsma, "A review of algorithms for audio fingerprinting," in *Multimedia Signal Processing, 2002 IEEE Workshop on.* IEEE, 2002, pp. 169–173.

[17] A. Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.

[18] A. Mesaros and T. Virtanen, "Automatic recognition of lyrics in singing," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2010, no. 1, p. 546047, 2010.

[19] C. Weare, "System and method for audio fingerprinting," Nov. 8 2005, uS Patent 6,963,975. [Online]. Available: https://www.google.com/patents/US6963975

[20] F. Holm and W. Hicken, "Audio fingerprinting system and method," Mar. 14 2006, uS Patent 7,013,301. [Online]. Available: http://www.google.com/patents/US7013301