A

Major Project

On

GENDER VOICE RECOGNITION USING MACHINE LEARNING ALGORITHMS

(Submitted in partial fulfilment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Neeraj Shiwani (177R1A05M6)

Bommaji Arun Pal (177R1A05C4)

Arushi Shandilya (177R1A0563)

Under the Guidance of

A.KIRAN KUMAR

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-22

DEPARTMENT OF COMPUTE SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"GENDER VOICE RECOGNITION USING MACHINE LEARNING ALGORITHMS"** is being submitted by **NEERAJ SHIWANI (177R1A05M6), BOMMAJI ARUN PAL (177R1A05C4), ARUSHI SHANDILYA (177R1A0563)** in partial fulfilment of the requirements for the award of the Degree of bachelor of Technology in Computer Science & Engineering, to the CMR Technical Campus, Kandlakoya, during academic year 2021-2022, is a bonafide work carried out by them under my guidance and supervision.

The results presented in this Project Work have been verified and are found to be satisfactory. The results embodied in this Project Work have not been submitted to any other University for the award of any other degree or diploma.

Mr. A.Kiran Kumar
Assistant Professor
INTERNAL GUIDE

Dr. A. RajiReddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on_____

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Mr. A.Kiran Kumar**, Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work.The blessing,help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC**) Mr. A. Uday Kiran, Mr. J. Narasimharao, Dr. T. S. Mastan Rao, Mrs. G. Latha , Mr. A Kiran Kumar,** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju**, Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy**, Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy**, Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project

NEERAJ SHIWANI (177R1A05M6)
BOMMAJI ARUN PAL (177R1A05C4)
ARUSHI SHANDILYA (177R1A0563)

ABSTRACT

Gender identification is one of the major problems in the area of signal processing. The system deals with finding the gender of a person using vocal features. One of the most challenging problems faced is feature selection from wide range of features, which is discriminating factor in classifying the gender of a person. The objective of this project is to design a system that determines the speaker gender using the pitch of the speaker's voice. Identifying the gender from the properties of voice data set i.e., pitch, median, frequency etc. can be possible by using machine learning. In this project, we are trying to classify gender into male or female based on the dataset containing various attributes related to voice like pitch, frequency etc. The data set have features with explanation data points recorded samples of male and female voices. The data set can be trained with different machine learning algorithms. The proposed system can detect the gender of the speaker with real time test data a new solution to detect the gender of the speaker using Fast Fourier Transform with Logistic Regression

LIST OF FIGURES

TABLE OF CONTENTS

INTRODUCTION

INTRODUCTION

Human based gender voice recognition system is used to recognize the gender of the person whether the person is male or female.

These gender voice recognition machines have currently received symbolic attention in computer vision community. The ability to do automatic recognition of human gender is essential for several systems that process human-source information like information retrieval, human-robot intercommunication etc. Gender voice recognition has great importance in games, business intelligence, demographic survey and forensics.

1.1 Project Scope

To learn and create a voice based gender voice recognition using machine learning algorithms

1.2 Project Purpose

The basic purpose of our project is to developed decision making ability's and it also help them in various aspects to increase their analytical abilities.

1.3 Project Feature

In this project the proposed methodology is to detect the gender of the speaker with real time test data a new solution to detect the gender of the speaker. The test data (our voice) is recorded and converted into numerical values and also the background noise the subtracted from the test data. With that the feature scope of the project can et more useful and innovative as the human needs

1.4 MOTIVATION

An automatic male-female voice discrimination system has two steps: extraction of audio features like ZCR, STE, Pitch, Tempo, MFCC etc from the input speech signal and discrimination based on the extracted feature. After extracting features, using supervised classifier (like k-NN classifier) voice separator will be built which is computationally inexpensive and capable of discriminating a male and female voice. The performance of the systems will be evaluated for a wide range of speech quality. As our main objective is to determine to which class or gender a particular speech sample belongs to, with the help of feature extraction and subsequent classification. We propose a methodology for solving the problem. Raw data collected would be pre-processed for missing data, anomalies and outliers. Then an algorithm would be trained on this data to create a model. This model would be used for forecasting the final results. ETL stands for Extract, Transform and load. It is a tool which is a combination of three functions. It is used to get data 8 from one database and transform it into a suitable format. Data preprocessing is a data mining technique used to transform sample raw data into an

understandable format. Real world collected data may be inconsistent, incomplete or contains an error and hence data preprocessing is required.

1.5 EXISTING SYSTEM

An automatic male-female voice discrimination system has two steps: extraction of audio features like ZCR, STE, Pitch, Tempo, MFCC etc from the input speech signal and discrimination based on the extracted feature. After extracting features, using supervised classifier (like k-NN classifier) voice separator will be built which is computationally inexpensive and capable of discriminating a male and female voice.

The performance of the systems will be evaluated for a wide range of speech quality. As our main objective is to determine to which class or gender a particular speech sample belongs to, with the help of feature extraction and subsequent classification.

1.6 PROBLEM STATEMENT

The problem statement is a simple way of asking what the designer or team is trying to do with the game. By identifying the design problem you are looking to solve, you eliminate situations in which you are simply copying the work of others and making something dull and uninspired.

1.7 PROPOSED SYSTEM

Human based gender voice recognition system is used to recognize the gender of the person whether the person is male or female.

Where as in proposed system we are adding the (GMM) Gaussian mixture models, A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as vocal-tract related spectral features in a speaker recognition system. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori(MAP) estimation from a well-trained prior model.

1.8 OBJECTIVE

The Objective of this project is to implement a Human based gender voice recognition system by using python programming language with Google Colaboratory and pre build data base from google drive

SYSTEM ANALYSIS

SYSTEM ANALYSIS

2.1 HARDWARE AND SOFTWARE REQUIREMENTS

The development and deployment of the application requires the following general and specific minimum requirements for hardware:

Component	Minimum requirement
System	Intel i-3,5,7 Processor
Hard disk	200GB
Monitor	Colour Monitor
RAM	4 GB/8GB

The development and deployment of the application requires the following general and specific minimum requirements for software:

Component	Minimum requirement
Operating System	Windows 8,10,Mac
Coding Language	Python
Python Language libraries	Numpy,scipy,scikit-learn,python_speech_features
Software Environment/IDE	Google Colaboratory

2.2 SOFTWARE REQUIREMENTS SPECIFICATION

1. Functional Requirements Output Design

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

•External Outputs, whose destination is outside the organization.

•Internal Outputs whose destination is within organization and they are the.

•User's main interface with the computer.

•Operational outputs whose use is purely within the computer department.

•Interface outputs, which involve the user in communicating directly.

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

Input Design

Input design is a part of overall system design. The main objective during the input design is as given below:

•To produce a cost-effective method of input.

•To achieve the highest possible level of accuracy.

•To ensure that the input is acceptable and understood by the user.

Input Stages

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

Input Types

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

Input Media

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to:

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security

- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled. **Error Detection**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors are always likely to occur, these types of errors can be discovered by using validations to check the input data.

Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

TECHNOLOGIES USED

3. TECHNOLOGIES USED

3.1 PYTHON PROGRAMMING

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and UNIX shell and other scripting languages.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula3, C, C++, Algol-68, SmallTalk, and UNIX shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.

Dynamic vs Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of "thing" each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point. If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn't require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program. With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values.

This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive.
The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

3.2 PYTHON LIBRARIES

These are the Python libraries that we have used in Gender Voice Recognition project:

- **Numpy:** Numpy stands for Numerical Python. Numpy is a Python library that provides a simple yet powerful data structure the n-dimensional array. It is used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
- **Pandas:** Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- **Pyaudio:** Pyaudio provides bindings for Port Audio, the cross-platform audio I/O library. This means that we can use Pyaudio to play and record audio on a variety of platforms, including Windows, Linux, and Mac.
- **Tkinter:** Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.
- **Scipy:** Scipy is Python library used for scientific computing and technical computing. It contains modules for optimization, linear algebra, integration, interpolation, special functions, Fourier transform, signal and image processing, ODE solvers etc

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be *imported* into other modules or into the *main* module

Testing code

As indicated above, code is usually developed in a file using an editor. To test the code, import it into a Python session and try to run it. Usually there is an error, so you go back to the file, make a correction, and test again. This process is repeated until you are satisfied that the code works. The entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carred out, the function can or can not return one or more values.There are three types of functions in python: help() ,min() ,print().

Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in firstname and familiy name (surname).

An example is a network: each network device (workstation, server, printer,) needs a unique name and address. Yet another example is the directory structure of file systems. The same file name can be used in different directories, the files can be uniquely accessed via the

pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- global names of a module
- local names in a function or method invocation
- built-in names: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python-Data Base Communication

Connector/Python provides a connect() call used to establish connections to the MySQL server. The following sections describe the permitted arguments for connect() and describe how to use option files that supply additional arguments.

A database is an organized collection of data. The data are typically organized to model aspects of reality in a way that supports processes requiring this information.

The term "database" can both refer to the data themselves or to the database management system. The Database management system is a software application for the interaction between users database itself.

Databases are popular for many applications, especially for use with web applications or customer-oriented programsUsers don't have to be human users. They can be other programs and applications as well. We will learn how Python or better a Python program can interact as a user of an SQLdatabase.

The Python standard for database interfaces is the Python DB-API, which is used by Python's database interfaces.

The DB-API has been defined as a common interface, which can be used to access relational databases.

In other words, the code in Python for communicating with a database should be the same, regardless of the database and the database module used. Even though we use lots of SQL examples, this is not an introduction into SQL but a tutorial on the Python interface.

SQLite is a simple relational database system, which saves its data in regular data files or even in the internal memory of the computer, i.e. the RAM. It was developed for embedded applications, like Mozilla-Firefox (Bookmarks), Symbian OS or Android.

SQLITE is "quite" fast, even though it uses a simple file. It can be used for large databases as well. If you want to use SQLite, you have to import the module sqlite3. To use a database, you have to create first a Connection object.

The connection object will represent the database. The argument of connection - in the following example "companys.db" - functions both as the name of the file, where the data will be stored, and as the name of the database. If a file with this name exists, it will be opened.

It has to be a SQLite database file of course! In the following example, we will open a database called company.

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.

For notes detailing the changes in each release of Connector/Python, see MySQL Connector/Python Release Notes. MySQL Connector/Python includes support for:

- Almost all features provided by MySQL Server up to and including MySQL Server version 5.7.
- Converting parameter values back and forth between Python and MySQL data types, for example Python datetime and MySQL DATETIME. You can turn automatic conversion on for convenience, or off for optimal performance.
- All MySQL extensions to standard SQL syntax.
- Protocol compression, which enables compressing the data stream between the client and server.
- Connections using TCP/IP sockets and on Unix using Unix sockets.
- Secure TCP/IP connections using SSL.
- Self-contained driver. Connector/Python does not require the MySQL client library or any Python modules outside the standard library

Google Colaboratory

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. We can use any languages in it and it gives 100gb rom and 12gb ram online

S YSTEM DESIGN

4. SYSTEM DESIGN

4.1 SOFTWARE DESIGN

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed; design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module are usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented.

In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. In other works, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue.

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user. Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

4.2 ARCHITECTURE

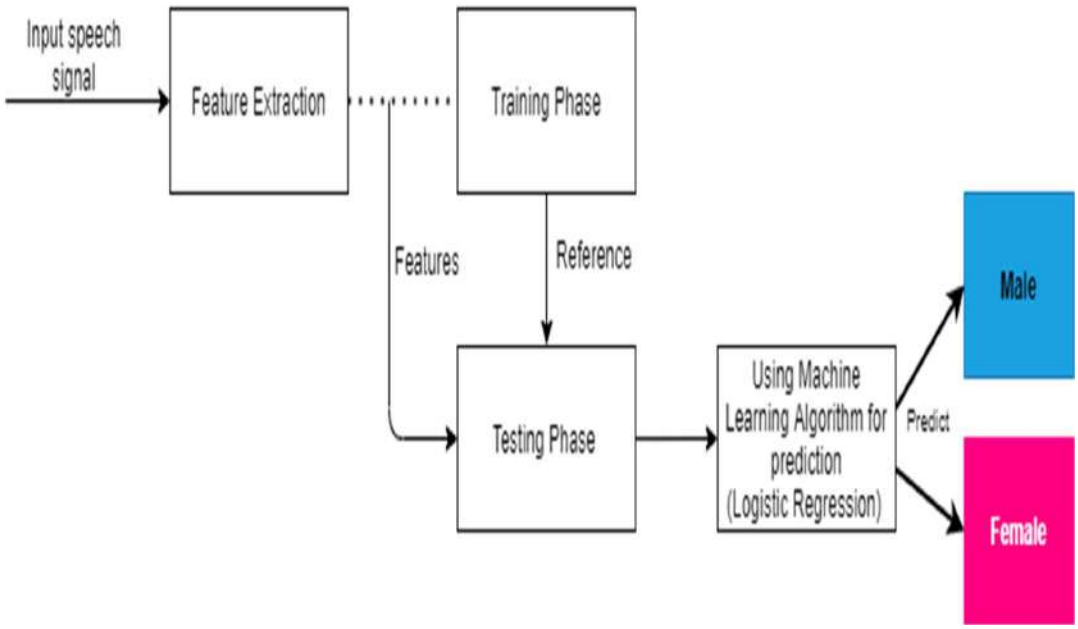
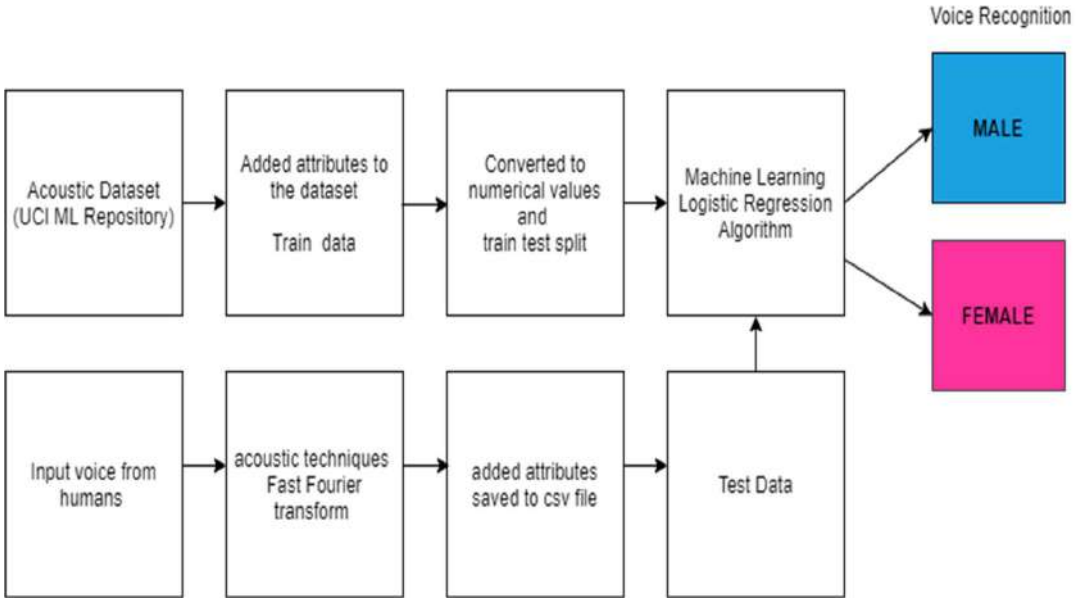


Figure 4.1 System Architecture for Gender Voice Recognition



Flow Diagram for Gender Voice Recognition

4.3 DATAFLOW DIAGRAM

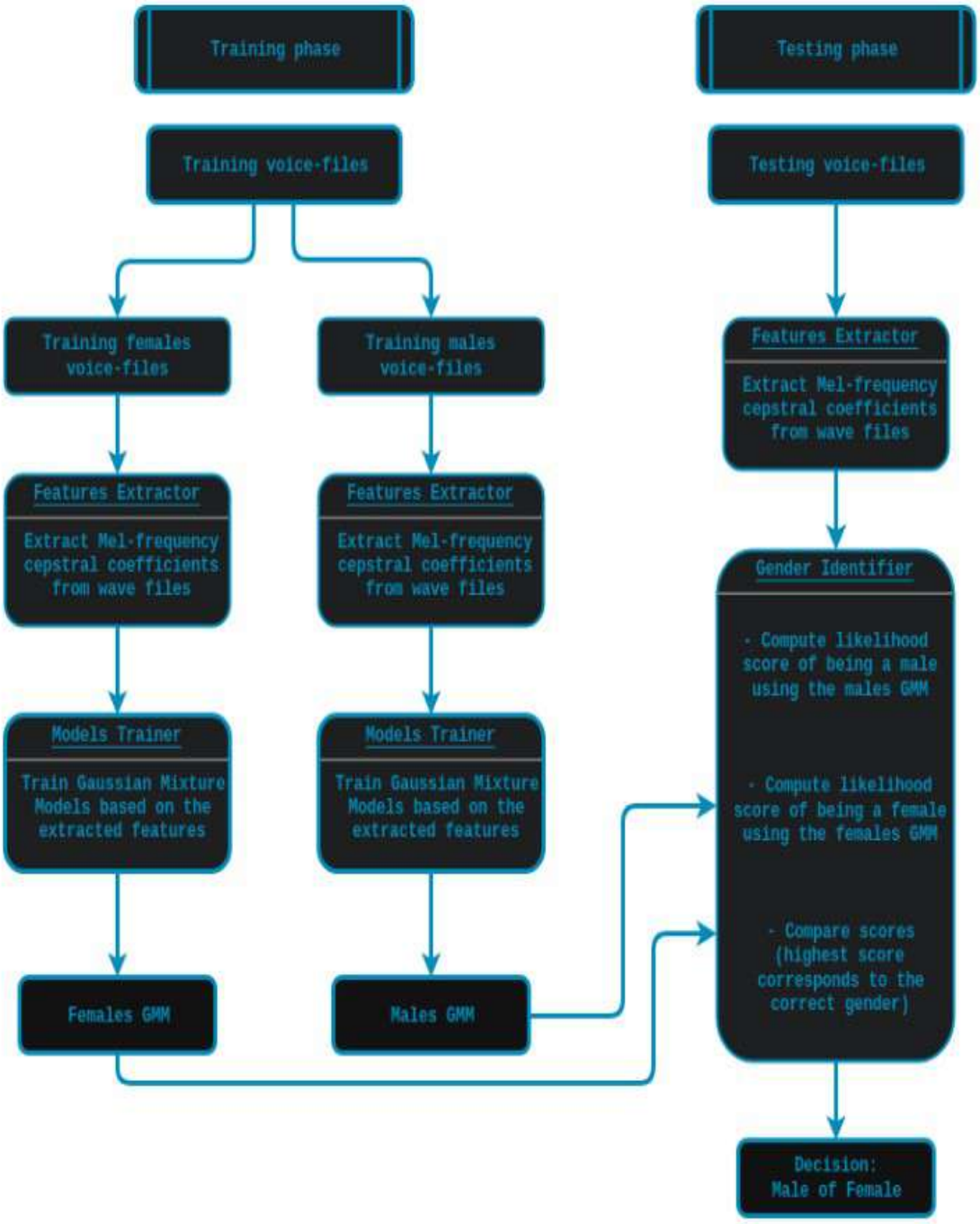


Figure 4.2 Dataflow Diagram for Gender Voice Recognition

METHODOLOGY

5. METHODOLOGY

5.1 MODULES DESCRIPTION

Dataset

The The Free ST American English Corpus dataset (SLR45) can be found on SLR45. It is a free American English corpus by Surfingtech, containing utterances from 10 speakers (5 females and 5 males). Each speaker has about 350 utterances.

5.2 ALGORITHM

1. Mel-Frequency Cepstrum Coefficients

Mel frequency cepstral coefficients (MFCC) was originally suggested for identifying monosyllabic words in continuously spoken sentences but not for speaker identification. MFCC computation is a replication of the human hearing system intending to artificially implement the ear's working principle with the assumption that the human ear is a reliable speaker recognizer [19]. MFCC features are rooted in the recognized discrepancy of the human ear's critical bandwidths with frequency filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to retain the phonetically vital properties of the speech signal. Speech signals commonly contain tones of varying frequencies, each tone with an actual frequency, f (Hz) and the subjective pitch is computed on the Mel scale. The mel-frequency scale has linear frequency spacing below 1000 Hz and logarithmic spacing above 1000 Hz. Pitch of 1 kHz tone and 40 dB above the perceptual audible threshold is defined as 1000 mels, and used as reference point

MFCC is based on signal disintegration with the help of a filter bank. The MFCC gives a discrete cosine transform (DCT) of a real logarithm of the short-term energy displayed on the Mel frequency scale [21]. MFCC is used to identify airline reservation, numbers spoken into a telephone and voice recognition system for security purpose. Some modifications have been proposed to the basic MFCC algorithm for better robustness, such as by lifting the log-mel-amplitudes to an appropriate power (around 2 or 3) before applying the DCT and reducing the impact of the low-energy parts

The Mel-Frequency Cepstrum Coefficients (MFCC) are used here, since they deliver the best results in speaker verification. 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.

2.Gaussian Mixture Model

According to D. Reynolds in Gaussian_Mixture_Models: A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as vocal-tract related spectral features in a speaker recognition system. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori(MAP) estimation from a well-trained prior model.

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as vocal-tract related spectral features in a speaker recognition system. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori (MAP) estimation from a well-trained prior model. The Gaussian mixture model for speech representation assumes that a M component mixture model with windowing function weights P(ωm) and the mixture components in the input voice sample contains Gaussian components.

The complete Gaussian mixture model is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities. GMMs are often used in biometric systems, most notably in speaker recognition systems, due to their capability of representing a large class of sample distributions. Given training vectors consisting of MFCC feature vectors and a GMM configuration, we wish to estimate the parameters of the GMM, λ, which in some sense best Signal & Image Processing : An International Journal (SIPIJ) Vol.4, No.4, August 2013 115 matches the distribution of the training feature vectors. There are several techniques available for estimating the parameters of a GMM [4]. By far the most popular and well-established method is maximum likelihood (ML) estimation. The aim of ML estimation is to find the model parameters which maximize the likelihood of the GMM given the training data. For a sequence of T training vectors X = {x1, . . . , xT }, the GMM likelihood, assuming independence between the input data vectors, can be written as, Unfortunately, this expression is a non-linear function of the parameters λ and direct maximization is not possible and hence the iterative Expectation-Minimization algorithm (EM) was used that was then used for training as well as matching purposes.

IMPLEMENTATION

6. IMPLEMENTATION

6.1 CODE

```
from google.colab import drive

drive.mount('/content/drive/')


class DataManager:

def __init__(self, dataset_path):

self.dataset_path = dataset_path


def extract_dataset(self, compressed_dataset_file_name, dataset_directory):

try:

# extract files to dataset folder

tar = tarfile.open(compressed_dataset_file_name, "r:gz")

tar.extractall(dataset_directory)

tar.close()

print("Files extraction was successfull ...")


except:

print("Ecxception raised: No extraction was done ...")


def make_folder(self, folder_path):

try:

os.mkdir(folder_path)

print(folder_path, "was created ...")

except:

print("Ecxception raised: ", folder_path, "could not be created ...")
```

```
def move_files(self, src, dst, group):
for fname in group:
os.rename(src + '/' + fname, dst + '/' + fname)

def get_fnames_from_dict(self, dataset_dict, f_or_m):
training_data, testing_data = [], []

for i in range(1,5):
length_data       = len(dataset_dict[f_or_m +"000" + str(i)])
length_separator  = math.trunc(length_data*2/3)

training_data += dataset_dict[f_or_m + "000" + str(i)][:length_separator]
testing_data  += dataset_dict[f_or_m + "000" + str(i)][length_separator:]

return training_data, testing_data

def manage(self):

# read config file and get path to compressed dataset
compressed_dataset_file_name = self.dataset_path
dataset_directory = compressed_dataset_file_name.split(".")[0]

# create a folder for the data
try:
os.mkdir(dataset_directory)
except:
pass
```

```
# extract dataset

self.extract_dataset(compressed_dataset_file_name, dataset_directory)


# select females files and males files

file_names   = [fname for fname in os.listdir(dataset_directory) if ("f0" in fname or "m0" in
fname)]

dataset_dict = {"f0001": [], "f0002": [], "f0003": [], "f0004": [], "f0005": [],

"m0001": [], "m0002": [], "m0003": [], "m0004": [], "m0005": [], }


# fill in dictionary

for fname in file_names:

dataset_dict[fname.split('_')[0]].append(fname)


# divide and group file names

training_set, testing_set = {},{}

training_set["females"], testing_set["females"] = self.get_fnames_from_dict(dataset_dict,
"f")

training_set["males"  ], testing_set["males"  ] = self.get_fnames_from_dict(dataset_dict,
"m")


# make training and testing folders

self.make_folder("TrainingData")

self.make_folder("TestingData")

self.make_folder("TrainingData/females")

self.make_folder("TrainingData/males")

self.make_folder("TestingData/females")

self.make_folder("TestingData/males")
```

```
# move files
self.move_files(dataset_directory, "TrainingData/females", training_set["females"])
self.move_files(dataset_directory, "TrainingData/males",   training_set["males"])
self.move_files(dataset_directory, "TestingData/females",  testing_set["females"])
self.move_files(dataset_directory, "TestingData/males",    testing_set["males"])


if __name__ == "__main__":
data_manager = DataManager("/content/drive/MyDrive/SLR45.tgz")
data_manager.manage()

import os
import sys
import math
import tarfile

from google.colab import drive
drive.mount('/content/drive')

pip install python_speech_features

import numpy as np
from sklearn import preprocessing
from scipy.io.wavfile import read
from python_speech_features import mfcc
from python_speech_features import delta
```

```
from sklearn import mixture


class FeaturesExtractor:

def __init__(self):

pass


def extract_features(self, audio_path):

"""

Extract voice features including the Mel Frequency Cepstral Coefficient (MFCC)

from an audio using the python_speech_features module, performs Cepstral Mean

Normalization (CMS) and combine it with MFCC deltas and the MFCC double

deltas.


Args:

audio_path (str) : path to wave file without silent moments.

Returns:

(array) : Extracted features matrix.

"""

rate, audio  = read(audio_path)

mfcc_feature = mfcc(# The audio signal from which to compute features.

audio,

# The samplerate of the signal we are working with.

rate,

# The length of the analysis window in seconds.

# Default is 0.025s (25 milliseconds)

winlen       = 0.05,

# The step between successive windows in seconds.
```

```
# Default is 0.01s (10 milliseconds)
winstep      = 0.01,
# The number of cepstrum to return.
# Default 13.
numcep       = 5,
# The number of filters in the filterbank.
# Default is 26.
nfilt        = 30,
# The FFT size. Default is 512.
nfft         = 512,
# If true, the zeroth cepstral coefficient is replaced
# with the log of the total frame energy.
appendEnergy = True)


mfcc_feature  = preprocessing.scale(mfcc_feature)
deltas        = delta(mfcc_feature, 2)
double_deltas = delta(deltas, 2)
combined      = np.hstack((mfcc_feature, deltas, double_deltas))
return combined

import pickle

class ModelsTrainer:

def __init__(self, females_files_path, males_files_path):
self.females_training_path = females_files_path
```

```
self.males_training_path   = males_files_path

self.features_extractor    = FeaturesExtractor()


def process(self):

females, males = self.get_file_paths(self.females_training_path,

self.males_training_path)

#print(females)

# collect voice features

female_voice_features = self.collect_features(females)

male_voice_features   = self.collect_features(males)

#print(female_voice_features)


# generate gaussian mixture models

females_gmm = mixture.GaussianMixture(n_components = 16, max_iter = 200,
covariance_type='diag', n_init = 3)

males_gmm   = mixture.GaussianMixture(n_components = 16, max_iter = 200,
covariance_type='diag', n_init = 3)

# fit features to models

print("training*******************************")

females_gmm.fit(female_voice_features)

males_gmm.fit(male_voice_features)

# save models

self.save_gmm(females_gmm, "females")

self.save_gmm(males_gmm,   "males")


def get_file_paths(self, females_training_path, males_training_path):

print(females_training_path)

# get file paths
```

```
females = [ os.path.join(females_training_path, f) for f in os.listdir(females_training_path) ]
males   = [ os.path.join(males_training_path, f) for f in os.listdir(males_training_path) ]
#print(females)
return females, males

def collect_features(self, files):

features = np.asarray(())
# extract features for each speaker
for file in files:
print("%5s %10s" % ("PROCESSNG ", file))
# extract MFCC & delta MFCC features from audio
vector    = self.features_extractor.extract_features(file)
# stack the features
if features.size == 0:  features = vector
else:                   features = np.vstack((features, vector))
return features

def save_gmm(self, gmm, name):
""" Save Gaussian mixture model using pickle.

Args:
gmm        : Gaussian mixture model.
name (str) : File name.
"""
filename = name + ".gmm"
with open(filename, 'wb') as gmm_file:
```

```
pickle.dump(gmm, gmm_file)

print ("%5s %10s" % ("SAVING", filename,))


if __name__== "__main__":

models_trainer = ModelsTrainer("/content/drive/MyDrive/TrainingData/females",
"/content/drive/MyDrive/TrainingData/males")

models_trainer.process()


import os

import pickle

import warnings

import numpy as np


warnings.filterwarnings("ignore")


class GenderIdentifier:


def __init__(self, females_files_path, males_files_path, females_model_path,
males_model_path):

self.females_training_path = females_files_path

self.males_training_path   = males_files_path

self.error                 = 0

self.total_sample          = 0

self.features_extractor    = FeaturesExtractor()

# load models

self.females_gmm = pickle.load(open(females_model_path, 'rb'))

self.males_gmm   = pickle.load(open(males_model_path, 'rb'))
```

```
def process(self):
files = self.get_file_paths(self.females_training_path, self.males_training_path)
# read the test directory and get the list of test audio files
for file in files:
self.total_sample += 1
print("%10s %8s %1s" % ("--> TESTING", ":", os.path.basename(file)))

vector = self.features_extractor.extract_features(file)
winner = self.identify_gender(vector)
expected_gender = file.split("/")
expected_gender=expected_gender[5]


print("%10s %6s %1s" % ("+ EXPECTATION",":", expected_gender))
print("%10s %3s %1s" %  ("+ IDENTIFICATION", ":", winner))

if winner != expected_gender: self.error += 1
print("----------------------------------------------------")



accuracy     = ( float(self.total_sample - self.error) / float(self.total_sample) ) * 100
accuracy_msg = "*** Accuracy = " + str(round(accuracy, 3)) + "% ***"
print(accuracy_msg)

def get_file_paths(self, females_training_path, males_training_path):
# get file paths
```

```
females = [ os.path.join(females_training_path, f) for f in os.listdir(females_training_path) ]

males   = [ os.path.join(males_training_path, f) for f in os.listdir(males_training_path) ]

files   = females + males

return files


def identify_gender(self, vector):

# female hypothesis scoring

is_female_scores         = np.array(self.females_gmm.score(vector))

is_female_log_likelihood = is_female_scores.sum()

# male hypothesis scoring

is_male_scores         = np.array(self.males_gmm.score(vector))

is_male_log_likelihood = is_male_scores.sum()


print("%10s %5s %1s" % ("+ FEMALE SCORE",":", str(round(is_female_log_likelihood,
3))))

print("%10s %7s %1s" % ("+ MALE SCORE", ":", str(round(is_male_log_likelihood,3))))


if is_male_log_likelihood > is_female_log_likelihood: winner = "males"

else                                    : winner = "females"

return winner



if __name__== "__main__":

gender_identifier = GenderIdentifier("/content/drive/MyDrive/TestingData/females",
"/content/drive/MyDrive/TestingData/males", "/content/drive/MyDrive/females.gmm",
"/content/drive/MyDrive/males.gmm")

gender_identifier.process()
```

```
import numpy as np
from sklearn import preprocessing
from scipy.io.wavfile import read
from python_speech_features import mfcc
from python_speech_features import delta
from sklearn import mixture
import os
import pickle
import warnings

warnings.filterwarnings("ignore")
import sys
import math
import tarfile

females_gmm = pickle.load(open("/content/drive/MyDrive/females.gmm", 'rb'))
males_gmm   = pickle.load(open("/content/drive/MyDrive/males.gmm", 'rb'))

def extract_features(audio_path):
rate, audio  = read(audio_path)
mfcc_feature = mfcc(# The audio signal from which to compute features.
audio,
# The samplerate of the signal we are working with.
rate,
# The length of the analysis window in seconds.
# Default is 0.025s (25 milliseconds)
winlen       = 0.05,
```

```
# The step between successive windows in seconds.
# Default is 0.01s (10 milliseconds)
winstep      = 0.01,
# The number of cepstrum to return.
# Default 13.
numcep       = 5,
# The number of filters in the filterbank.
# Default is 26.
nfilt        = 30,
# The FFT size. Default is 512.
nfft         = 512,
# If true, the zeroth cepstral coefficient is replaced
# with the log of the total frame energy.
appendEnergy = True)



mfcc_feature  = preprocessing.scale(mfcc_feature)
deltas        = delta(mfcc_feature, 2)
double_deltas = delta(deltas, 2)
combined      = np.hstack((mfcc_feature, deltas, double_deltas))
return combined

def identify_gender(vector):
# female hypothesis scoring
is_female_scores         = np.array(females_gmm.score(vector))
is_female_log_likelihood = is_female_scores.sum()
# male hypothesis scoring
```

```
is_male_scores         = np.array(males_gmm.score(vector))

is_male_log_likelihood = is_male_scores.sum()

if is_male_log_likelihood > is_female_log_likelihood: winner = "male"

else                                  : winner = "female"

return winner


vector =
extract_features('/content/drive/MyDrive/TestingData/males/m0003_us_m0003_00011.wav')

winner = identify_gender(vector)

print(winner)
```

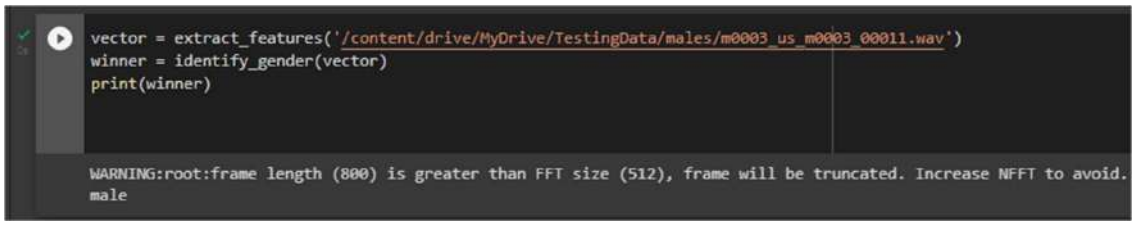
RESULTS AND DISCUSSION

7.1 RESULTS AND DISCUSSION

RESULT:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

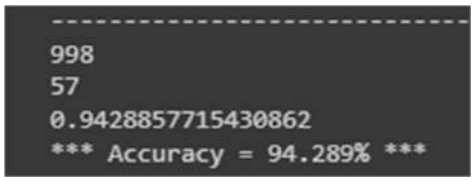
- The system results in a **95%** accuracy of gender detection.
- The code can be further optimized using multi-threading, acceleration libs and multi-processing.
- The accuracy can be further improved using GMM normalization aka a UBM-GMM system



CONCLUSION

8.1 Conclusion:

The proposed methodology is to detect the gender of the speaker with real time test data a new solution to detect the gender of the speaker. The Fast Fourier Transform with Machine Learning model has achieved accuracy of 95% on test data which is more as compared to MFCC gender voice recognition model



8.2 Future Scope:

From experimentations, it can be seen that further exploration of potentially interesting features could also be pictured out. At the same time, the set of selected features seem to be enough for the tested models to achieve good accuracy.

Adding more voice data and other gender data helps to detect more genders and store the speaker's voice in a database for analysis purpose.

Can be added to recognition the child voice and they/them voice as we add more data set and apply it.

BIBILOGRAPHY

BIBILOGRAPHY

REFERENCES:

[1] J. M. Hilbe, Logistic Regression Models, CRC Press, 2009.

[2] M. Araya-Salas, G. Smith-Vidaurre, warbleR: an R package to streamline analysis of animal acoustic signals. Methods Ecol Evolution, 2016, doi:10.1111/2041-210X.12624

[3] Voice Gender Recognizer Recognition of Gender from Voice using Deep Neural Networks By Lakhan Jasuja, Akhtar Rasool, Gaurav Hajela at 2020 International Conference on Smart Electronics and Communication (ICOSEC).

[4] Gender recognition system using speech signal by Md. Sadek Ali, Md. Shariful Islam and Md. Alamgir Hossain

WEBSITE

- https://docs.python.org/3/faq/general.html
- https://ieeexplore.ieee.org/document/9215254
- https://www.researchgate.net/publication/276197281_Gender_Recognition_System_Using_Speech_Signal
- https://realpython.com
- https://github.com/175M6NRJ/MAJOR-PROJECT-175M6

JOURNAL

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VI June 2022- Available at www.ijraset.com

Gender Voice Recognition Using Machine Learning Algorithms

Arun Pal Bommaji[1], Neeraj Shiwani[2], Arushi Shandilya[3]
[1, 2, 3]*Dept.Of CSE, CMR Technical Campus*

Abstract: ***Gender identification is one of the major problems in the area of signal processing. The system deals with finding the gender of a person using oral features. One of the most persnickety problems faced is feature selection from wide range of features, which is distinguishing factor in classifying the gender of aperson. The ideal of this design is to design a system that determines the speaker gender using the pitch of the speaker's voice. relating the gender from the plots of voice data set i.e., pitch, median, frequency etc. can be possible by using machine learning. In this design, we're trying to classify gender into male or female based on the data set containing varied attributes related to voice like pitch, frequency etc. The data set have features with explanation data points recorded samples of male and female voices. The data set can be trained with different machine learning algorithms. The proposed system can determine the gender of the speaker with real time test data a new result to discover the gender of the speaker using Fast Fourier Transform with Logistic Regression***
Keywords: Voice recognition, machine learning- Frequency Cepstrum Coefficients, Gaussain mixture model.

I. INTRODUCTION

Human based gender voice recognition system is used to recognize the gender of the person whether the person is male or female. These gender voice recognition machines have currently received symbolic attention in computer vision community. The capability to do automatic recognition of human gender is essential for several systems that process human- source information like information retrieval, human- robot intercommunication etc. Gender voice recognition has great significance in games, business intelligence, demographic scan and forensics.

II. LITERATURE SURVEY

An automatic male-female voice discrimination system has two steps extraction of audio features like ZCR, STE, Pitch, Tempo, MFCC etc from the input speech signal and discrimination based on the uprooted feature. After extracting features, using supervised classifier(like k- NN classifier) voice division will be erected which is computationally affordable and able of differentiating a male and female voice. The performance of the systems will be estimated for a wide range of speech quality. As our main ideal is to determine to which class or gender a particular speech sample belongs to, with the help of point birth and posterior bracket. We propose a methodology for answering the problem. Raw data collected would be pre-processed for missing data, anomalies and outliers. furthermore an algorithm would be trained on this data to produce a model. This model would be used for forecasting the final results. ETL stands for Extract, Transform and load. It's a tool which is a combination of three functions. It's used to get data 8 from one database and transform it into a suitable format. Data prepossessing is a data mining technique used to transform sample raw data into an accessible format. Real world collected data may be inconsistent, incomplete or contains an error and hence data preprocessing is required.

III. PROPOSED SYSTEM

Human based gender voice recognition system is used to recognize the gender of the person whether the person is male or female. Where as in proposed system we're adding the(GMM) Gaussian mixture models, A Gaussian Mixture Model(GMM) is a parametric probability consistence function represented as a weighted sum of Gaussian component consistence. GMMs are generally used as a parametric model of the probability distribution of nonstop measures or features in a biometric system, similar as spoken- tract related spectral features in a speaker recognition system. GMM parameters are estimated from training data using the iterative Expectation- Maximization(EM) algorithm or Maximum A Posteriori(MAP) estimation from a well- trained previous model.

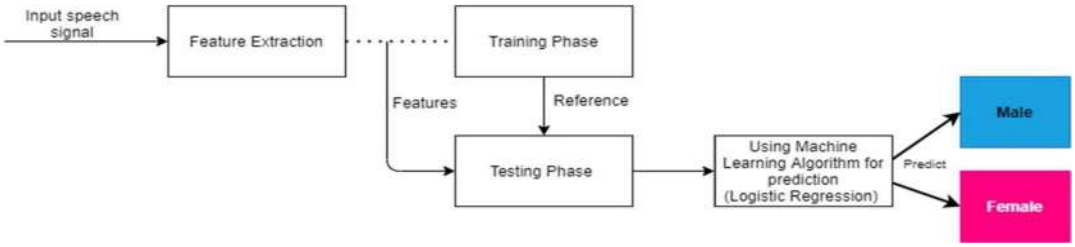


Figure1.1 Architecture of the Model.

IV. RESULT

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of factors,sub-assemblies, assemblies and/ or a finished product It's the process of exercising software with the intent of icing that the

1) The system results in a 95 preciseness of gender discovery.
2) The law can be further optimized usingmulti-threading, acceleration libs and multi- processing.
3) The delicacy can be further bettered using GMM normalization aka a UBM- GMM system.

```
vector = extract_features('/content/drive/MyDrive/TestingData/males/m0003_us_m0003_00011.wav')
winner = identify_gender(vector)
print(winner)
```

```
WARNING:root:frame length (800) is greater than FFT size (512), frame will be truncated. Increase NFFT to avoid.
male
```

FIG 1.2 Result

```
vector = extract_features('/content/drive/MyDrive/TestingData/females/f0001_us_f0001_00018.wav')
winner = identify_gender(vector)
print(winner)
```

```
WARNING:root:frame length (800) is greater than FFT size (512), frame will be truncated. Increase NFFT to avoid.
female
```

FIG 1.3 Result

```
class FeaturesExtractor:
    def __init__(self):
        pass

    def extract_features(self, audio_path):
        """
        Extract voice features including the Mel Frequency Cepstral Coefficient (MFCC)
        from an audio using the python_speech_features module, performs Cepstral Mean
        Normalization (CMS) and combine it with MFCC deltas and the MFCC double
        deltas.

        Args:
            audio_path (str) : path to wave file without silent moments.
        Returns:
            (array) : Extracted features matrix.
        """
        rate, audio  = read(audio_path)
        mfcc_feature = mfcc(# The audio signal from which to compute features.
                            audio,
                            # The samplerate of the signal we are working with.
                            rate,
                            # The length of the analysis window in seconds.
                            # Default is 0.025s (25 milliseconds)
                            winlen       = 0.05,
                            # The step between successive windows in seconds.
                            # Default is 0.01s (10 milliseconds)
                            winstep      = 0.01,
                            # The number of cepstrum to return.
                            # Default 13.
                            numcep       = 5,
                            # The number of filters in the filterbank.
                            # Default is 26.
                            nfilt        = 30,
                            # The FFT size. Default is 512.
                            nfft         = 512,
                            # If true, the zeroth cepstral coefficient is replaced
                            # with the log of the total frame energy.
                            appendEnergy = True)


        mfcc_feature  = preprocessing.scale(mfcc_feature)
        deltas        = delta(mfcc_feature, 2)
        double_deltas = delta(deltas, 2)
        combined      = np.hstack((mfcc_feature, deltas, double_deltas))
        return combined
```

FIG 1.4 Code Execution

V. CONCLUSION

The proposed methodology is to descry the gender of the speaker with real time test data a new result to descry the gender of the speaker. The Fast Fourier transfigure with Machine Learning model has achieved delicacy of 95 on test data which is more as compared to MFCC gender voice recognition model

REFERENCES

[1] J.M. Hilbe, Logistic Regression Models, CRC Press, 2009.
[2] M. Araya- Salas,G. Smith- Vidaurre, warbleR an R package to streamline analysis of beast aural signals. styles Ecol Evolution, 2016, doi10.1111/ 2041-210X.12624
[3] Voice Gender Recognizer Recognition of Gender from Voice using Deep Neural Networks By Lakhan Jasuja, Akhtar Rasool, Gaurav Hajela at 2020 International Conference on Smart Electronics and Communication(ICOSEC).
[4] Gender recognition system using speech signal by Md. Sadek Ali, Md. Shariful Islam and Md. Alamgir Hossain


ISSN No. : 2321-9653
iJRASET
International Journal for Research in Applied Science & Engineering Technology
IJRASET is indexed with Crossref for DOI-DOI : 10.22214
Website : www.ijraset.com, E-mail : ijraset@gmail.com
ISRA Journal Impact Factor: 7.429
45.98
INDEX COPERNICUS
THOMSON REUTERS
Researcher ID: N-9681-2016
10.22214/IJRASET
doi
cross ref
TOGETHER WE REACH THE GOAL
SJIF 7.429
Certificate
It is here by certified that the paper ID : IJRASET44347, entitled
Gender Voice Recognition Using Machine Learning Algorithms
by
Arun Pal Bommaji
after review is found suitable and has been published in
Volume 10, Issue VI, June 2022
in
International Journal for Research in Applied Science & Engineering Technology
Good luck for your future endeavors
Editor in Chief, iJRASET


ISSN No. : 2321-9653
iJRASET
International Journal for Research in Applied Science & Engineering Technology
IJRASET is indexed with Crossref for DOI-DOI : 10.22214
Website : www.ijraset.com, E-mail : ijraset@gmail.com
ISRA Journal Impact Factor: 7.429
45.98
INDEX COPERNICUS
THOMSON REUTERS
Researcher ID: N-9681-2016
10.22214/IJRASET
doi
cross ref
TOGETHER WE REACH THE GOAL
SJIF 7.429
Certificate
It is here by certified that the paper ID : IJRASET44347, entitled
Gender Voice Recognition Using Machine Learning Algorithms
by
Neeraj Shiwani
after review is found suitable and has been published in
Volume 10, Issue VI, June 2022
in
International Journal for Research in Applied Science & Engineering Technology
Good luck for your future endeavors
Editor in Chief, iJRASET

ISSN No. : 2321-9653

iJRASET

International Journal for Research in Applied Science & Engineering Technology

IJRASET is indexed with Crossref for DOI-DOI : 10.22214

Website : www.ijraset.com, E-mail : ijraset@gmail.com

ISRA Journal Impact Factor: 7.429

45.98 INDEX COPERNICUS

THOMSON REUTERS Researcher ID: N-9681-2016

10.22214/IJRASET doi cross ref

TOGETHER WE REACH THE GOAL SJIF 7.429

Certificate

It is here by certified that the paper ID : IJRASET44347, entitled

Gender Voice Recognition Using Machine Learning Algorithms

by

Arushi Shandilya

after review is found suitable and has been published in

Volume 10, Issue VI, June 2022

in

International Journal for Research in Applied Science & Engineering Technology

Good luck for your future endeavors

Editor in Chief, iJRASET