International Journal of Multidisciplinary Research and Growth Evaluation
ISSN: 2582-7138
Received: 17-09-2021; Accepted: 20-10-2021
www.allmultidisciplinaryjournal.com
Volume 2; Issue 6; November-December 2021; Page No. 349-363

Speech to multi-language text conversion

Dr. M Upendra Kumar
Professor of CSE MJCET OU Hyderabad, Telangana, India

Corresponding Author: **Dr. M Upendra Kumar**

DOI: https://doi.org/10.54660/.IJMRGE.2021.2.6.349-363

Abstract
Right from the beginning of previous century, researchers have shown interest in areas like Automatic Speech Recognition, Image Processing and Natural Language Processing. The area of Automatic Speech Recognition (ASR) has received attention over the past five decades due to its application in both commercial and military. In the recent times this can be attributed to the advancements in Artificial Intelligence and Advanced Algorithms. ASR takes speech as input and converts it in to text. ASR is employed in electronic dictionaries, Customer Call Centers, Voice Dictation and Query based Information Systems, Speech Transcription, Avionics, Smart Houses and Access Systems and many more areas. ASR can also be used to interact with handicapped people. ASR enables human beings interact with computers using speech rather than using keyboards & mouse (Vimalaand Radha V., 2012). ASR aims to provide natural machine interface where in speech acts input to the machine.

Generally, ASR is based on two tasks viz. Identification of Phoneme and Whole-Word Decoding. A relationship between speech signal and speech segment that has dissimilar physical or perceptual features usually termed as phones is established in two steps. The first step deals with dimensionality reduction and second step deals with the estimation of likelihood of each phoneme. In the dimensionality reduction phase, the volume of the speech signal is decreased by extracting the relevant information using task-specific knowledge. In the next phase, the system recognizes the word sequence using a discriminative program. Traditionally ASR systems preferred the Mel frequency Cepstral coefficients (MFCC) for the first phase and Discriminative techniques for the second phase. Over the years ASR systems have evolved from being an integration of multiple trained components to “end-to-end” Deep neural architectures that link speech to text directly. The proposed work implements an MLP with AdaBoost Classifier. The MLP will be used to extract discriminative features from the speech data. Later AdaBoost classifier will map these features to the relevant set of words.

Keywords: Speech, multi-language, ASR

1. Introduction

Right from the beginning of previous century, researchers have shown interest in areas like Automatic Speech Recognition, Image Processing and Natural Language Processing. The area of Automatic Speech Recognition (ASR) has received attention over the past five decades due to its application in both commercial and military. In the recent times this can be attributed to the advancements in Artificial Intelligence and Advanced Algorithms. ASR takes speech as input and converts it in to text. ASR is employed in electronic dictionaries, Customer Call Centers, Voice Dictation and Query based Information Systems, Speech Transcription, Avionics, Smart Houses and Access Systems and many more areas. ASR can also be used to interact with handicapped people. ASR enables human beings interact with computers using speech rather than using keyboards & mouse (Vimalaand Radha V., 2012). ASR aims to provide natural machine interface where in speech acts input to the machine.

Generally, ASR is based on two tasks viz. Identification of Phoneme and Whole-Word Decoding. A relationship between speech signal and speech segment that has dissimilar physical or perceptual features usually termed as phones is established in two steps. The first step deals with dimensionality reduction and second step deals with the estimation of likelihood of each phoneme. In the dimensionality reduction phase, the volume of the speech signal is decreased by extracting the relevant information using task-specific knowledge. In the next phase, the system recognizes the word sequence using a discriminative program. Traditionally ASR systems preferred the Mel frequency Cepstral coefficients (MFCC) for the first phase and Discriminative techniques for the second phase. Over the years ASR systems have evolved from being an integration of multiple trained components to “end-to-end” Deep neural architectures that link speech to text directly.

The proposed work implements an MLP with AdaBoost Classifier. The MLP will be used to extract discriminative features from the speech data. Later AdaBoost classifier will map these features to the relevant set of words.

1.1. Background

Over the years, the scale and complexity of investigations and developments in ASR for a wide range of applications has increased gradually. Around 1970s most ASR systems were developed using isolated speech recognition methodology (Baker. J 1975). They used Hidden Markov Models (HMMs) and steadily improved the recognition accuracy (Baker. J 2009).

Around late 1980s the focus shifted to continuous speech recognition methodology. Due to this the vocabulary size of recognition increased by many folds. For example, vocabulary size of in the Resource Management (RM) Task had about 900 words which increased to 20,000 words in the Wall Street Journal (WSJ) task (P. Douglas and J. M. Baker 1992). Current ASR Systems like voice-based search engines require virtually unlimited vocabulary size (Bacchiani. M *et al*. 2008). The difference between real time operating conditions and standard laboratory conditions also impacted the recognition accuracy of the ASR systems. Before 1995 testing and analysis of ASR systems were performed using speech recorded in noiseless environment (C.H.Lee 1996). These systems lost their accuracy as soon as they were applied to real time applications because in real time ASR Systems had to deal with noisy data. From this point on the focus of research in ASR Systems shifted to recognition of spontaneous and conversational speech in noisy environments. Currently End-to-End Systems that perform Acoustic Frame to Phone mapping in one go are being developed. In these types of systems all the modules are trained simultaneously using Advanced Machine Learning algorithms. These models take raw speech as input and generate phoneme class conditional probabilities as output. Applications like Broadcast News Transcription, Telephone Conversation Analytics gained from this new approach. Recent Commercial ASR Systems have to tackle with challenging tasks like voice search, short message dictation, YouTube transcription, Multilingual Conversion and so on. These systems work on data recorded from natural conversations which do not follow any predefined data collection procedures. Moreover, they have to operate in diverse acoustic environments catering the needs of millions of users. The ASRs have to adapt to these challenges and provide the user accurate transcriptions. This adaptation can be performed by feature-based methods or model-based methods. In feature-based methods the feature vectors of test data are normalized whereas in model-based methods acoustic model attributes are modified to better model the trail conditions.

1.2 Rationale of the study

What is the research issue?

Automatic Speech Recognition Systems have to deal with challenges like acoustic diversity of speech data, Environmental noise. Voices spoken by people of different backgrounds, accents and differing styles contribute to Acoustic Diversity. Channel Characteristic differences add up to the variations in speech signals. Speech services for global users require the ASR systems to have the ability to adapt to both acoustic and channel diversity. These non-speech diversities usually confuse the ASR system and significantly degrade its recognition accuracy. To solve these issues Adaptation techniques that can separate speech diversity from non-speech diversities and then absorb these non-speech diversities have to be developed.

1.3 Research Gap

Although there are many studies pertaining to Automatic speech Recognition, there is little work present in the literature that accurately recognizes the speech and converts it into text of Multiple Languages by employing hierarchical training for MLP and AdaBoost for testing.

There is still a need to develop models that are robust to reverberation and variable Acoustics. This thesis will focus on the model-based approach to address the issues of reverberation and variable acoustics.

1.4. Purpose of Study

The purpose of the work presented in this thesis is to contribute to the improvement of speech recognition. It aims to deal with issues like acoustic diversity and environment diversity. The objective is to build an ASR that reads speech as input and gives text as output. The text generated can be later translated in to other languages of the user's choice. This study will concentrate on model based acoustic adaptation technique for speech recognition. The research question that this thesis answers is

1. How to optimize the training time of MLP on speech data?
2. Can AdaBoost improve the accuracy of an ASR system?

1.5. Report Outline

This thesis is divided in to six chapters. First chapter introduces the topic, explains the background of the problem and outlines the research issues. It also discusses the purpose and objectives of this study. The second chapter conducts a detailed review of available literature related to Automatic Speech Recognition Systems. It also highlights the achievements and pitfalls of the existing works in this area. The third chapter will throw light on the software required to answer the research questions raised. The programming language, packages and necessary libraries that are required to conduct this research is discussed broadly. Model Design, Experimental Setup and Test strategies implemented for achieving accurate results in emotion classification is explained in the fourth chapter of this thesis. The fifth chapter gives a detailed analysis of results obtained. It performs a comparative analysis of the proposed technique with existing techniques present in the literature. The final chapter draws meaningful conclusions from the result analysis and suggests how this study can be further improved.

2. Literature Review

2.1 Investigation

Kuldip K. Paliwal and *et al* in the year 2004 had discussed that without being affected by their popularity for front end parameter in speech recognition, the cepstral coefficients which had been obtained from linear prediction analysis is sensitive to noise. Here, the use of spectral sub band centroids had been discussed by them for robust speech recognition. They discussed that performance of recognition can be achieved if the centroids are selected properly as in comparison with MFCC. to construct a dynamic centroid feature vector a procedure had been proposed which

essentially includes the information of transitional spectral information (Jingdong Chen, 2004).

Esfandier Zavarehei and *et al* in the year 2005, studied that a time-frequency estimator for enhancement of noisy speech signal in DFT domain is introduced. It is based on low order auto regressive process which is used for modeling. The time-varying trajectory of DFT component in speech which has been formed in Kalman filter state equation. For restarting Kalman filter, a method has been formed to make alteration on the onsets of speech. The performance of this method was compared with parametric spectral subtraction and MMSE estimator for the increment of noisy speech. The resultant of the proposed method is that residual noise is reduced and quality of speech in improved using Kalman filters (Hakan Erdogan, 2005).

Ibrahim Patel and *et al* in the year 2010, had discussed that frequency spectral information with mel frequency is used to present as an approach in the recognition of speech for improvement of speech, based on recognition approach which is represented in HMM. This approach of Mel frequency utilizes the frequency observation in speech within a given resolution resulting in the overlapping of resolution feature which results in the limit of recognition. In speech recognition system which is based on HMM, resolution decomposition is used with a mapping approach in a separating frequency. The result of the study is that there is an improvement in quality metrics of speech recognition with respect to the computational time and learning accuracy in speech recognition system (Ibrahim Patel, 2010).

Kavita Sharma and Prateek Hakar in the year 2012 has represented recognition of speech in a broader solution. It refers to the technology that will recognize the speech without being targeted at single speaker. Variability in speech pattern, in speech recognition is the main problem. Speaker characteristics which include accent, noise and co-articulation are the most challenging sources in the variation of speech. In speech recognition system, the function of basilar membrane is copied in the front-end of the filter bank. To obtain better recognition results it is believed that the band subdivision is closer to the human perception. In speech recognition system the filter which is constructed for speech recognition is estimated of noise and clean speech (Kavita Sharma, 2012).

2.2. Proposed Idea

There is a gap is found after analyzing the previous works in speech recognition. The speech recognition can be done in two steps. In this project the same is proposed. In this project first the audio is converted in to text then in the text the keywords are searched. The audio files can be easily converted into to text using python speech recognition library function. Once the audio file is converted in to text then it is easy to search the keywords in the files.

3. System Analysis & Feasibility Study

This chapter includes, existing system, proposed system, methologies (or) algorithms, software development life cycle, feasibility study.

3.1. Existing System

- Existing System did not used any Machine Learning Techniques
- Previous Systems are not efficient
- The Systems are developed for One-to-One Language Conversion
- The behavior of the System is not user friendly

3.2. Disadvantages of Existing System

- Lack of Accuracy and Misinterpretation
- Time Costs and Productivity
- Accents and Speech Recognition
- Background Noise interference

3.3. Proposed System

- There is a gap is found after analyzing the previous works in speech recognition. The speech recognition can be done efficiently by using machine-learning technique. The machine learning techniques increase the efficiency and accuracy of the model.
- In this project, we are proposing a novel technique of speech recognition based on Machine learning.
- In this project the Audio data set is collected from the kaggle repository and the audio file are used for training the model. Adaboost algorithm is used in this project. Then the audio file, which we want, is fed to the model then the audio file will be translated to English text. Then after using the google translator, the text will be converted in to any other recognized language.
- By removing the noise and for smoothening of the speech while converting the Speech to Text Conversion we use AdaBoost Algorithm

3.3.1. Advantages Of Proposed System

- Accurate Speech to Text Conversion
- User friendly Interface
- Speed in Conversion

4. Methodologies (Or) Algorithms

4.1. Adaboost Algorithm

AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique that is used as an Ensemble Method in Machine Learning. It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instances. Boosting is used to reduce bias as well as the variance for supervised learning. It works on the principle where learners are grown sequentially. Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones. Adaboost algorithm also works on the same principle as boosting, but there is a slight difference in working. Let's discuss the difference in detail.

Working

First, let us discuss the working of boosting. It makes n number of decision trees during the training period of data. As the first decision tree/model is made, the record which is incorrectly classified during the first model is given more priority. Only these records are sent as input for the second model. The process will go on until we specify a number of base learners we want to create. Remember, the repetition of records is allowed with all boosting techniques.

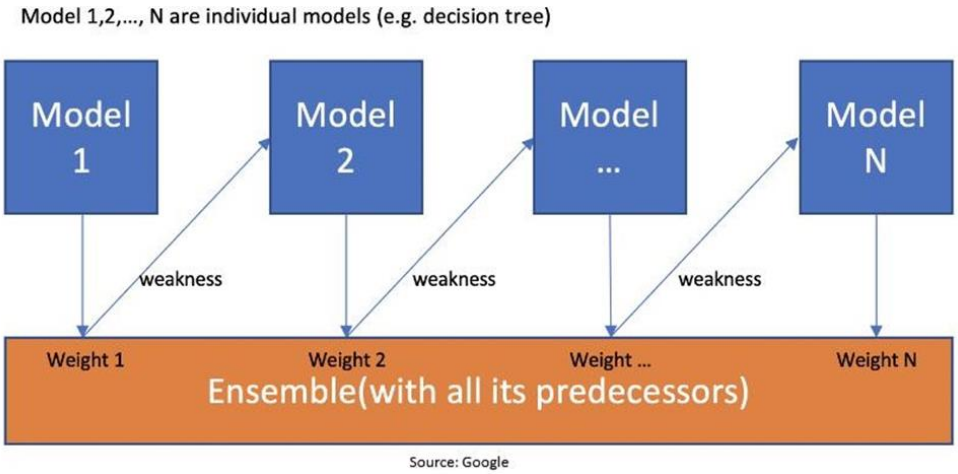

Fig 1: Working of Adaboost Algorithm

This figure shows that when the first model is made and the errors from the first model are noted by the algorithm, the record which is incorrectly classified is given as the input for the next model. This process is repeated until the specified condition is met. As you can see in the figure, there are n number of models made by taking the errors from the previous model. This is how boosting works. The models 1,2, 3, N are individual models that can be known as decision trees. All types of boosting models work on the same principle.

Since we know the boosting principle, it will be easy to understand the AdaBoost algorithm. Let's deep dive into the working of Adaboost. When the random forest is used, the algorithm makes n number of trees. It makes proper trees that consist of a start node with several leaves nodes. Some trees might be bigger than others, but there is no fixed depth in a random forest. But with Adaboost, that's not the case. In AdaBoost, the algorithm only makes a node with two leaves, and this is known as Stump.

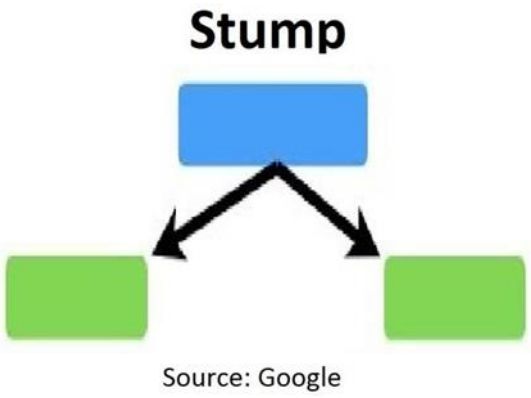

Fig 2

The figure here represents the stump. It can be seen clearly that it has only one node with only two leaves. These stumps are weak learners, and boosting techniques prefer this. The order of stumps is very important in AdaBoost. The error of the first stump influences how the other stump is made. Let's understand this with an example.

Table 1

Row No.	Feature 1	Feature 2	Feature 3	Output	Sample Weight
1				Yes	1/5
2				Yes	1/5
3				No	1/5
4				No	1/5
5				Yes	1/5

Here I have created a sample dataset that consists of only three features, and the output is in categorical form. The image shows the actual representation of the dataset. As the output is in binary/categorical form, it becomes a classification problem. In real life, the dataset can have any number of records and features in it. Let us consider 5 datasets for explanation purposes. The output is in categorical form and, here it's Yes or No. All these records will get a sample weight. To assign some sample weight, the formula used is, W=1/N where N is the number of records. In this dataset, there are only 5 records, so the sample weight becomes 1/5 initially. Every record gets the same weight. In this case, it's 1/5.

Step 1: Creating First Base Learner

Now it's time to create the first base learner. The algorithm takes the first feature, i.e., feature 1, and creates the first stump f1. It will create the same number of stumps as the number of features. Here, it will create 3 stumps as there are only 3 features in this dataset. From all these stumps it will create three decision trees and can be called stumps base learner model. Out of these 3 models, the algorithm selects only one. For selecting a base learner, there are two properties, those are, Gini and Entropy. We must calculate Gini or Entropy the same way it is calculated for decision trees. The stump that has the least value will be the first base learner. In the below figure, all the 3 stumps can be made with 3 features. The number below the leaves represents the correctly and incorrectly classified records. By using these records, the Gini or entropy index is calculated. The stump that has the least entropy or Gini will be selected for the base learner. Let's assume that the entropy index is the least for stump 1. So, let's take stump 1, i.e., feature 1 as our first base learner.

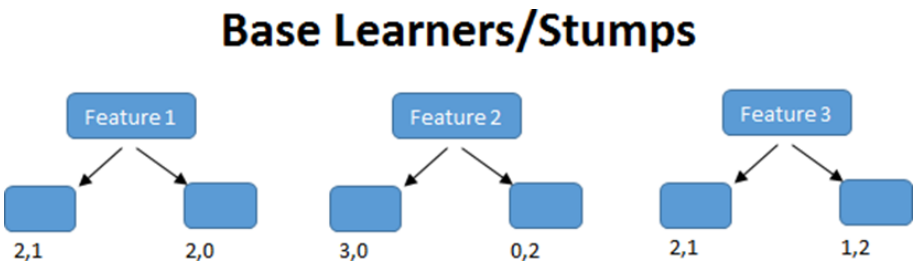

Fig 3

Table 2

Row No.	Feature 1	Feature 2	Feature 3	Output	Sample Weight
1				Yes	1/5
2				Yes	1/5
3				No	1/5
4				No	1/5
5				Yes	1/5

Here, feature (f1) has classified 2 records correctly and 1 incorrectly. The row in the figure that is marked red is incorrectly classified. For this, we will be calculating the total error.

Step 2: Calculating the Total Error (TE)
The total error is the sum of all the errors in the classified record for sample weights. In our case, there is only 1 error, so Total Error (TE) = 1/5.

Step 3: Calculating Performance of Stump
Formula for calculating Performance of Stump is:

$$\text{Performance of Stump} = 1/2 \ln \left[\frac{1-TE}{TE} \right]$$

Where, ln is natural log and TE is Total Error.

In our case, TE is 1/5. By putting the value of total error in the above formula and after solving, we get the value for the performance of Stump as 0.693. You must be wondering why it's necessary to calculate the TE and performance of stump? The answer is, we must update the sample weight before proceeding for the next model or stage because if the same weight is applied, we receive the output from the first model. In boosting, only the wrong records/incorrectly classified records got more preference than the correctly classified records. Thus, only the wrong records from the decision tree/stump are passed on to another stump. While in AdaBoost, both records were allowed to pass, the wrong records are repeated more than the correct ones. We must increase the weight for the wrongly classified records and decrease the weight for the correctly classified records. In the next step, we will be updating the weights based on the performance of the stump.

Step 4: Updating Weights
For incorrectly classified records the formula is:
New Sample Weight = Sample Weight * e^(Performance)
In our case Sample weight = 1/5 so, 1/5 * e^ (0.693) = 0.399

And for correctly classified records, we use the same formula with a negative sign with performance, so that the weight for correctly classified records will reduce compared to the incorrect classified ones. The formula is:

New Sample Weight = Sample Weight * e^-(Performance)

Putting the values, 1/5 * e^-(0.693) = 0.100

Table 3

Row No.	Feature 1	Feature 2	Feature 3	Output	Sample Weight	Updated Weight
1				Yes	1/5 ↓	0.1
2				Yes	1/5 ↑	0.399
3				No	1/5 ↓	0.1
4				No	1/5 ↓	0.1
5				Yes	1/5 ↓	0.1
-	-	-	-	-	1	0.799

Table 4

Row No.	Feature 1	Feature 2	Feature 3	Output	Sample Weight	Updated Weight	Normalized Weight
1				Yes	1/5 ↓	0.1	0.13
2				Yes	1/5 ↑	0.399	0.50
3				No	1/5 ↓	0.1	0.13
4				No	1/5 ↓	0.1	0.13
5				Yes	1/5 ↓	0.1	0.13
-	-	-	-	-	1	0.799	1

The updated weight for all the records can be seen in the figure. As is known, the total sum of all the weights should be 1. But in this case, one can see that the total updated weight of all the records is not 1, it's 0.799. To make the total sum 1, one must divide every updated weight by the total sum of updated weight. For example, if our updated weight is 0.399

and we divide this by 0.799, i.e., 0.399/0.799=0.50.
0.50 can be known as the normalized weight. In the below figure, we can see all the normalized weight and their sum is approximately 1.

Step 5: Creating New Dataset

Now, it's time to create a new dataset from our previous one. In the new dataset, the frequency of incorrectly classified records will be more than the correct ones. While considering these normalized weights, we have to create a new dataset and that dataset is based on normalized weights. It will probably select the wrong records for training purposes. That will be the second decision tree/stump. To make a new dataset based on normalized weight, the algorithm will divide it into buckets.

Table 5

Normalized Weight	Buckets
0.13	0 - 0.13
0.50	0.13 - 0.63
0.13	0.63 - 0.76
0.13	0.76 - 0.89
0.13	0.89 - 1.02

So, our first bucket is from 0 – 0.13, second will be from 0.13 – 0.63(0.13+0.50), third will be from 0.63 – 0.76(0.63+0.13), and so on. After this the algorithm will run 5 iterations to select different-different records from the older dataset. Suppose, in 1st iteration, the algorithm will take a random value 0.46, then it will go and see in which bucket that value falls and selects those records in the new dataset, then again it will select a random value and see in which bucket it is and select that record for the new dataset and the same process is repeated for 5 times.

There is a high probability for the wrong records to get selected several times. This will be the new dataset. It can be seen in the below image that row number 2 has been selected multiple times from the older dataset as that row is incorrectly classified in the previous dataset.

Table 6

Row No.	Feature 1	Feature 2	Feature 3	Output
2				Yes
3				No
2				Yes
5				Yes
2				Yes

Based on this new dataset, the algorithm will again create a new decision tree/stump and it will repeat the same process from step 1 till it sequentially passes through all stumps and finds that there is less error when compared with normalized weight that we had in the initial stage.

Deciding Output

Suppose with the above dataset, the algorithm constructed 3 decision trees or stumps, the test dataset will pass through all the stumps which have been constructed by the algorithm. While passing through the 1st stump, it gives the output as 1, passing through 2nd stump it again gives the output as 1, and while passing through 3rd stump it gives the output as 0. So, in AdaBoost algorithm also, the majority of votes take place between the stumps, the same as in random trees. And in this case, the final output will be 1. This is how the output with test data is decided.

Coding AdaBoost in Python

In Python, it is easy with only 3-4 lines of code for AdaBoost algorithm. We must import the AdaBoost classifier from the sci-kit learn library. Before applying AdaBoost to any dataset, one should split the data into train and test. After splitting the data into train and test, the training data is ready to train the AdaBoost model. This data has both the input as well as output. After training the train data, our algorithm will try to predict the result on the test data. Test data only consists of the inputs. The output of test data is not known by the model. So, test data is given to the model. One can check the accuracy by comparing the actual output of the test data and the predicted output by the model. This can help us conclude how our model is performing. How much accuracy can be considered depends on the problem statement. If it's a medical problem, then accuracy should be above 90%. Usually, 70% accuracy is considered good. Accuracy also depends on more factors apart from the type of model. The below figure shows the code to implement AdaBoost.

```
[ ] from sklearn.ensemble import AdaBoostClassifier
    ad=AdaBoostClassifier()

[ ] pred=ad.fit(xtrain, ytrain).predict(xtest)

[ ] accuracy_score(ytest, pred)

    0.7533333333333333
```

Fig 4: Code for AdaBoost in Python

At last, I would like to conclude that Adaptive Boosting is a good ensemble technique and can be used for both Classification and Regression problems. But in most cases, it is used for classification problems. It is better than any other model as it improves model accuracy, one can check this by going in sequence. First try decision trees and then go for the random forest, next apply to boost and finally go for AdaBoost. We can see that the accuracy keeps increasing as we follow the above sequence. The weight assigning technique after every iteration makes AdaBoost algorithm different from all other boosting algorithms. And that's the best thing about the AdaBoost algorithm.

System Requirements
Software Requirements

- Operating System : Windows 7+
- Server-side Script : Python 3.6+
- Platforms : Jupiter Notebook
- Libraries used : Pandas, Scikit-learn, Matplotlib, Google trans and Librosa
- Dataset : AN4 dataset from Kaggle

Design
System design

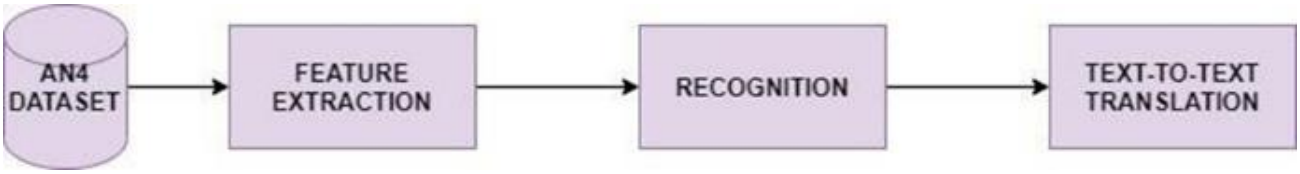

Fig 5: Speech to Multilingual Text System Architecture

The design of the project is described as, the user has the predefined data set which contains all the information related to audio files. Later, Python libraries are imported for the data set. The libraries include NumPy, Panda, Sklearn, Matpoltlib. Pandas is an open-source, Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Implementation
Introduction

This chapter explains the implementation of the proposed system. Setting up the environment, selection of dataset, preprocessing of dataset, modeling and conclusions are included with appropriate code snippets and output screen shots. Besides this chapter focus on explanation of functions which are used in various stages of implementation.

This chapter reviews the hierarchy components required to design an Automatic Speech to Multilingual Text Convertor. The goal of this model is to convert speech into text automatically.

This project uses a workflow model to develop the proposed model. This model offers a robust solution for the problem of building software models from initiation to completion. The work flow model is a pipeline of various phases that are to be executed for developing a software project.

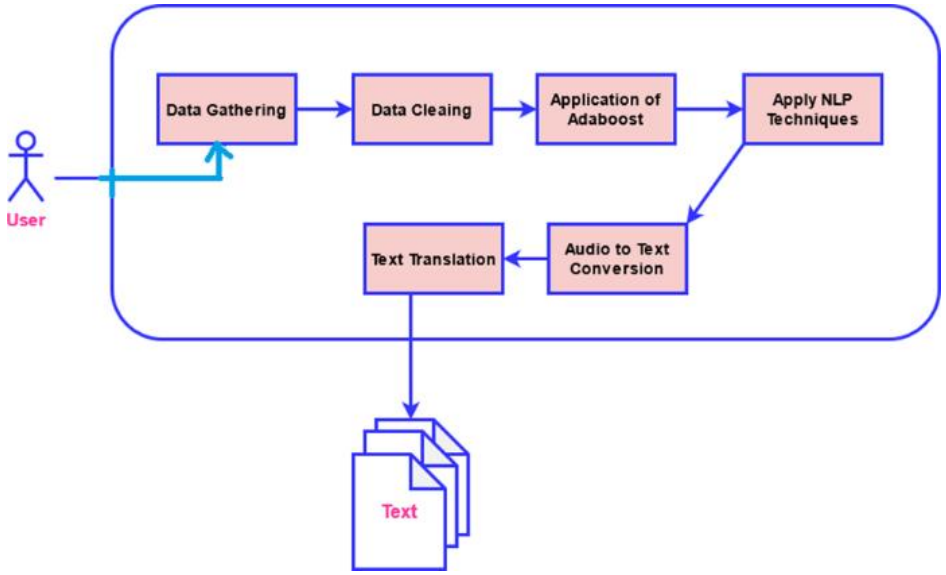

Fig 6: Proposed System Architecture

This model design contains five phases namely, data Collection, data preprocessing, data analysis, feature selection, speech recognition, speech to text conversion and text-to-text conversion. The workflow model is depicted in figure 6.1.

Installation required

- Python Speech Recognition module: pip install speechrecognition Speech Input Using a Microphone and Translation of Speech to Text
- Allow Adjusting for Ambient Noise: Since the surrounding noise varies, we must allow the program a second or too to adjust the energy threshold of recording so it is adjusted according to the external noise level.
- Speech to text translation: This is done with the help of Google Speech Recognition. This requires an active internet connection to work. However, there are certain offline Recognition systems such as PocketSphinx, but have a very rigorous installation process that requires several dependencies. Google Speech Recognition is one of the easiest to use.

Data Collection and Preprocessing

Data is very important in any project. Data plays a major role in drawing the correct results. The collection and data preprocessing are discussed in the chapter 3. The audio file is read using a function rate, data = wavfile.read(audio file path).

Generally. Wav format audio files are accepted in the instruction. If the audio file is not in. Wav format is should be converted then its path is given as parameter for the function. The noise can be removed from the audio files by using filters. The filters can be designed to eliminate the noise. Noise is removed using function removenoise () which is supported by librosa.

Data Analysis

The data analysis can be used to study the data type and patterns. After analyzing, the data if required changing of data types is applied for the parameters, which are not suitable for processing. If the data type is objects or character then the data is converted to suitable formats.

The amplitude, frequency and pitch of the audio data is analyzed using 'wave' library functions.

To analyze the amplitude of the audio the function 'np.fromstring(soundInfo, np.int16)' is used.

Speech Recognition

Speech recognition is one more important phase in this project. This is done with the help of Speech Recognition library. The speech recognition can be done using Recognizer function.

Import speech recognition as sr
From guessing_game.py import recognize_speech_from_mic r = sr.Recognizer()

Audio to text conversion

Audio to text conversion is last and crux of the thesis. In this phase the audio file is given as input. That audio file is read using a function AudioFile() which is a predefined function present in the Speech Recognition library. After Reading the file that audio is recorded using function record(). The audio file is converted to text data using a function recognize_google(audio_data). This function returns the text for the given audio file. The code snippet for conversion of speech to text is given in the figure 6.6.

```
with sr.AudioFile(filename) as source:
        audio_data = r.record(source)
        text = r.recognize_google(audio_data)
        print(text)
```

Fig 7: Speech to Text Conversion Code

Text To Text Conversion

In Text-to-text conversion phase the text translation in to other languages is done. This is language translation phase. The language translation can be done using 'googletrans' library. In googletrans library supports a function to translate the text from one language to other language. The language form which the text to convert is known as source language. The language to which that text has to be translated is known as destination language. The function translate('text", src='la1', dest='la2') is used to convert the text from one

language to other language.

Results

Introduction

In this chapter the results projected using appropriate visualization diagrams. The results are compared with the systems which are existed.

The main aim of this thesis is to convert the speech to text and language translation. The speech to text converting is done using speech recognition supporting library functions. Text translation is done using google translator library functions.

Results Discussion

The Audio analysis results are given in the figure 7.2.1 and 7.2.2

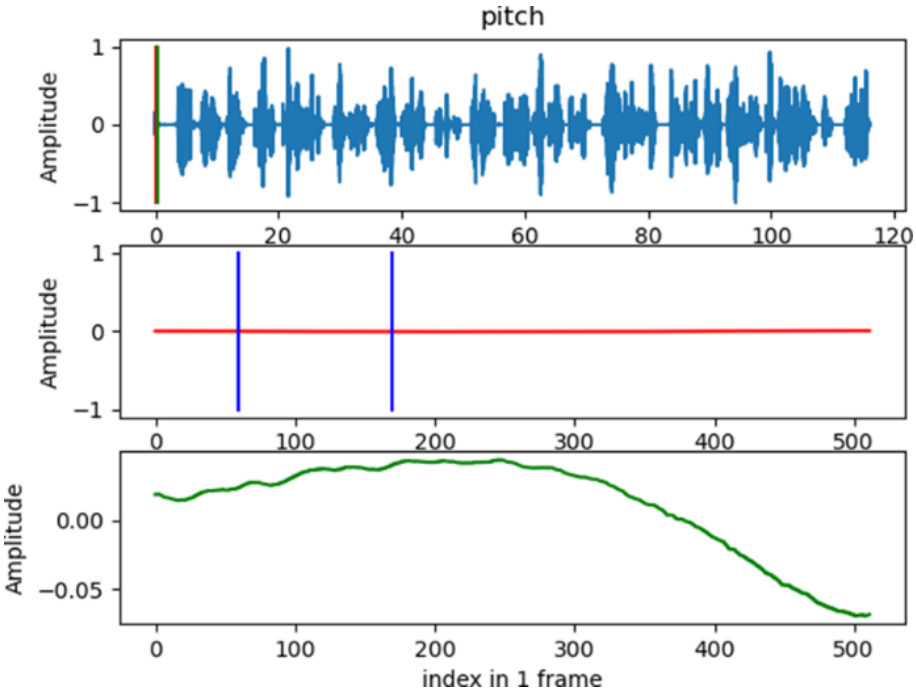

Fig 8: Pitch of the given audio

The Figure 7.2.1 shows the pitch of the audio file, which we want to convert in to text in amplitude domain. By observing the figure we can say that pitch of the audio file not constant.

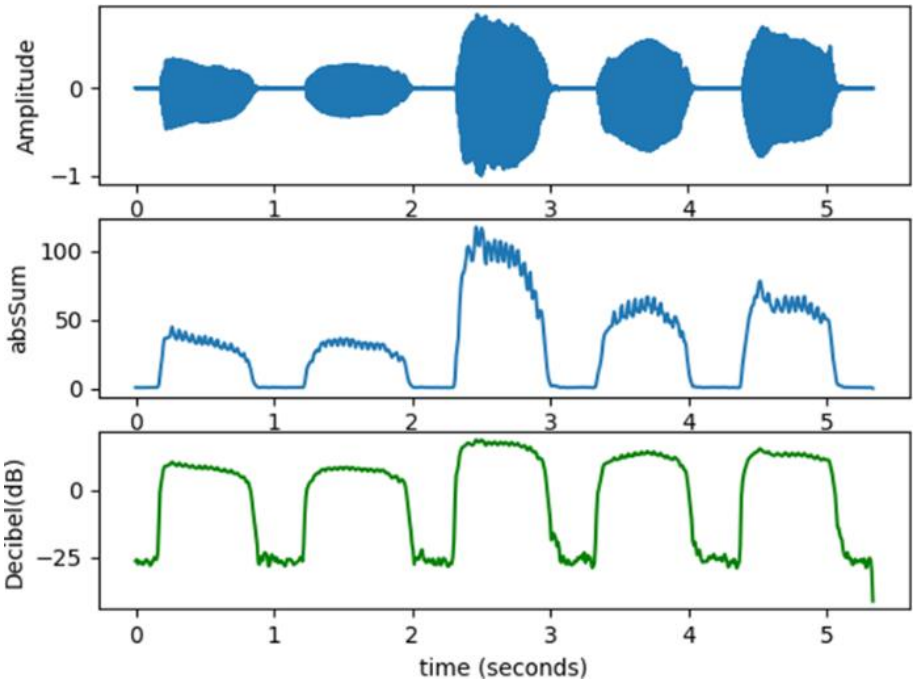

Fig 9: Amplitude of the given audio

The Figure 7.2.2 shows the volume of the audio file, which we want to convert in to text in amplitude domain and in terms of Decibels also. By observing the figure we can say that volume of the audio file not constant.

Speech to Text Conversion Result

For speech conversion recognizer function is used, the function recognizer is available in speech recognition library.

The given input audio file: 4.wav The output text:

```
[[-36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339]
 [-36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339 -36.04365339 -36.04365339 -36.04365339 -36.04365339
  -36.04365339]]
```

Fig 10: MFCC

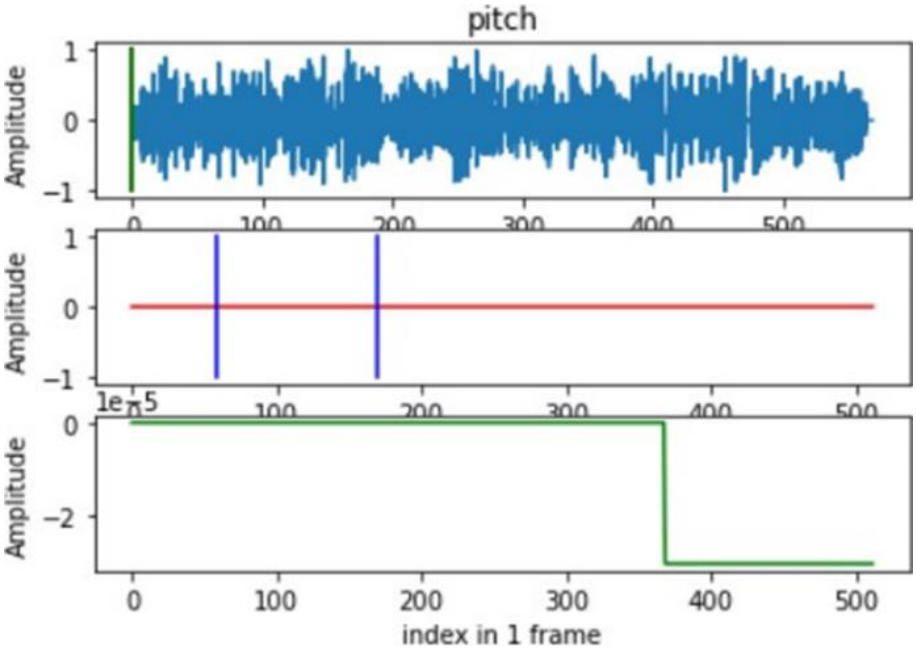

Fig 11: Pitch

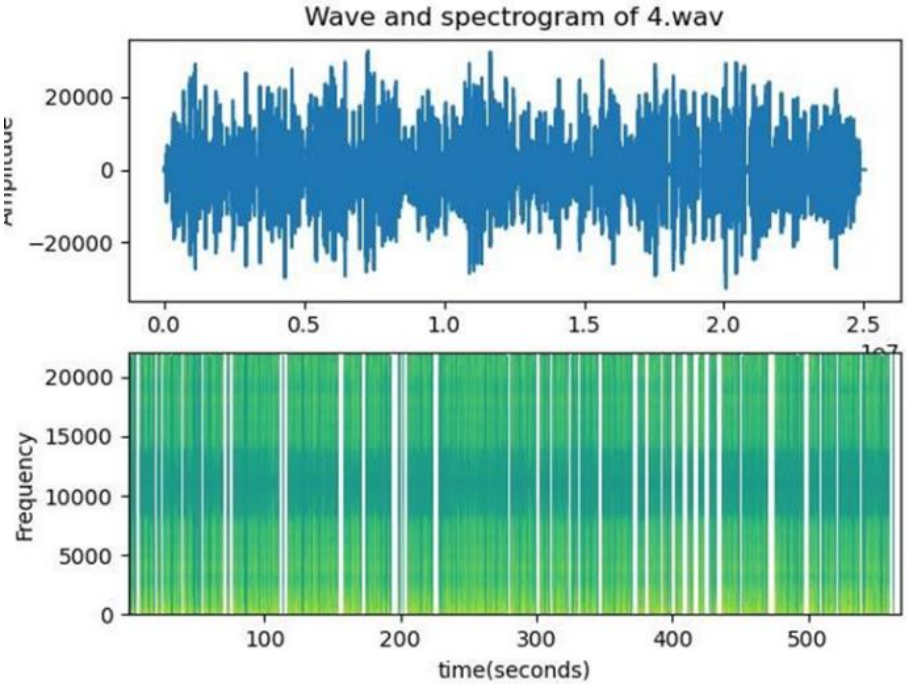

Fig 12: Spectrogram

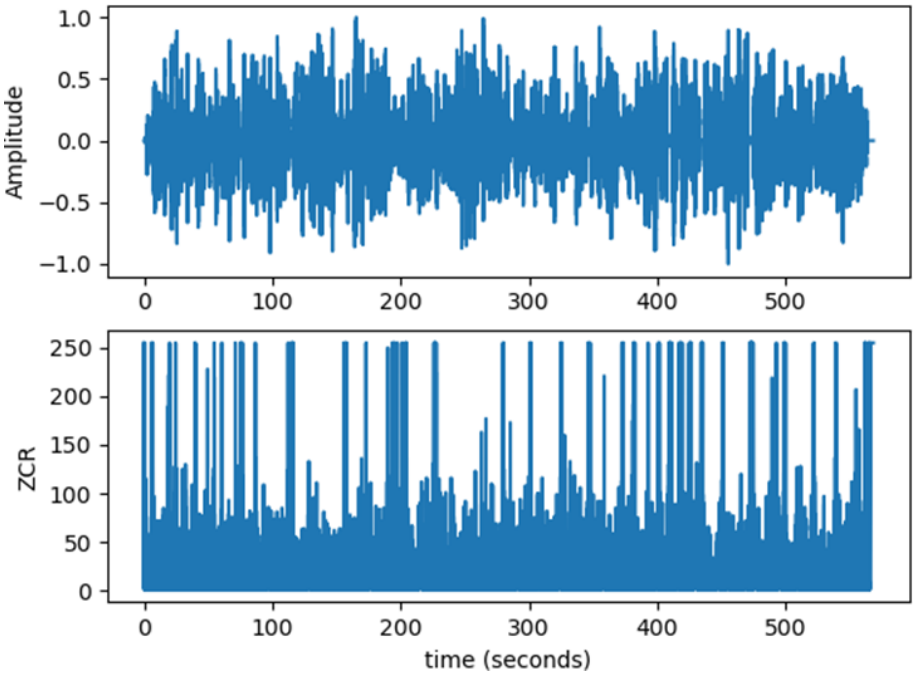

Fig 13: ZeroCR

Fig 14: English to Hindi

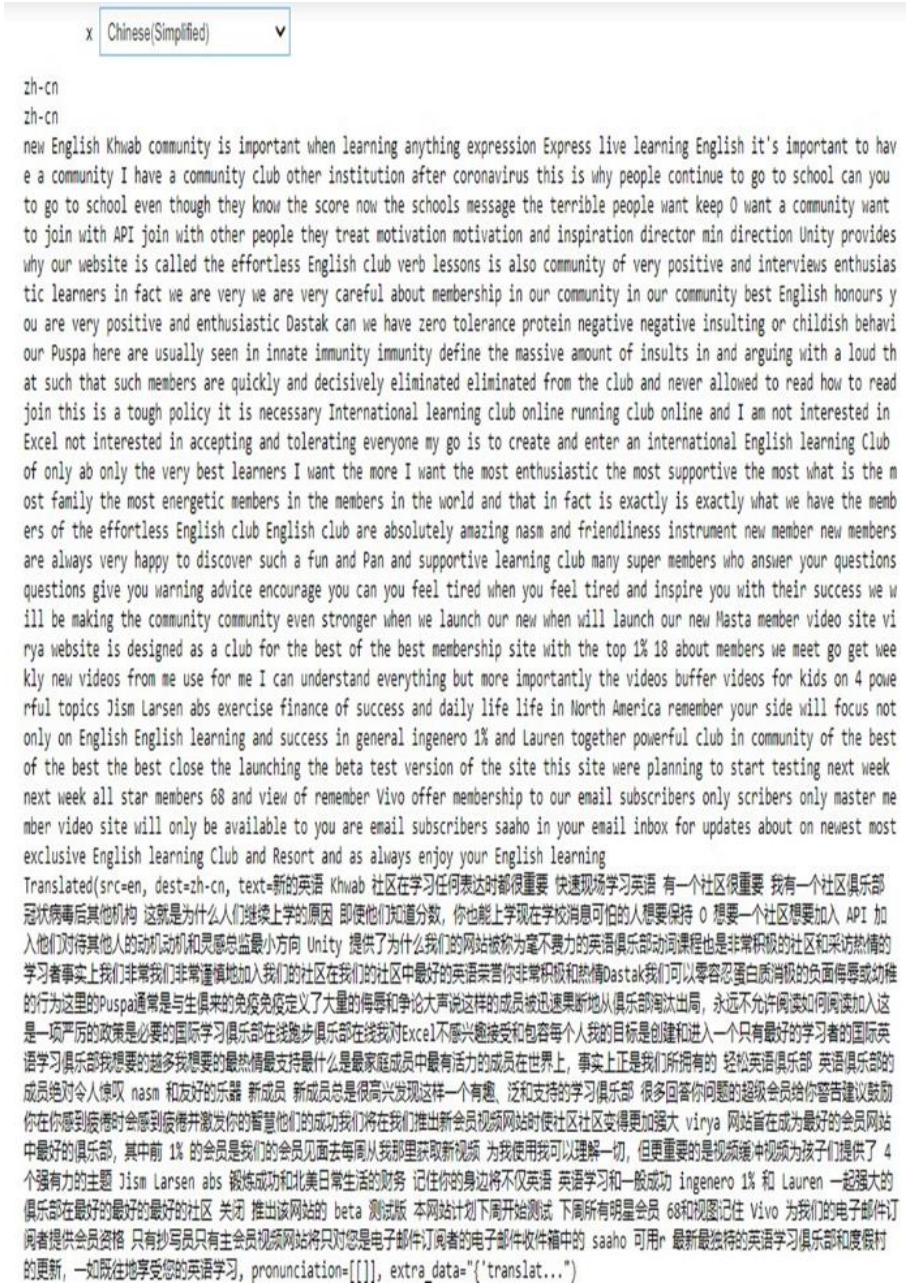

Fig 15: English to Chinese

The given input audio file: 5.wav
The output text:

```
[[10.94713983  9.99398292 12.18790791 12.8504111  13.62553786 12.29264385
  10.92248824 11.55679057 12.25664498 11.51486177  8.38093661  8.81039205
   8.46320476  9.27358183  8.84892385  8.84642652  9.43254083  8.7842305
   8.91336212  6.71144106  6.21862968  6.32807392  6.42951027  8.40391954
   8.34886804  8.45447038]
 [11.12858872 11.2736033  13.86072626 14.51233543 15.49314012 12.93437362
  10.75359312 13.08680835 14.08713444 13.01643775  9.467426    8.9429215
   9.78233427 10.60088878 11.54396398  9.99562533 10.919647   11.20341505
  11.79934337  7.21605933  4.05816941  4.21901235  6.81349061 10.92982219
  10.12483045 10.10499301]]
```

Fig 16: MFCC

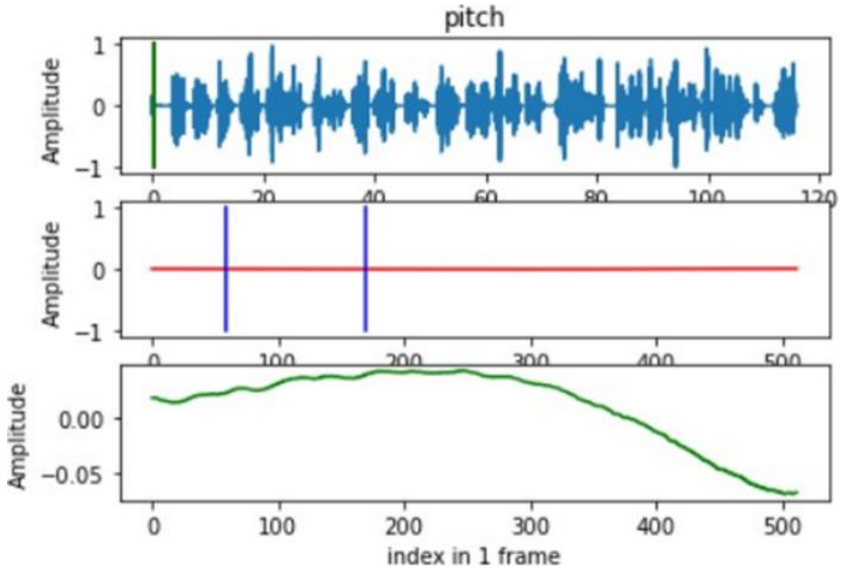

Fig 17: Pitch

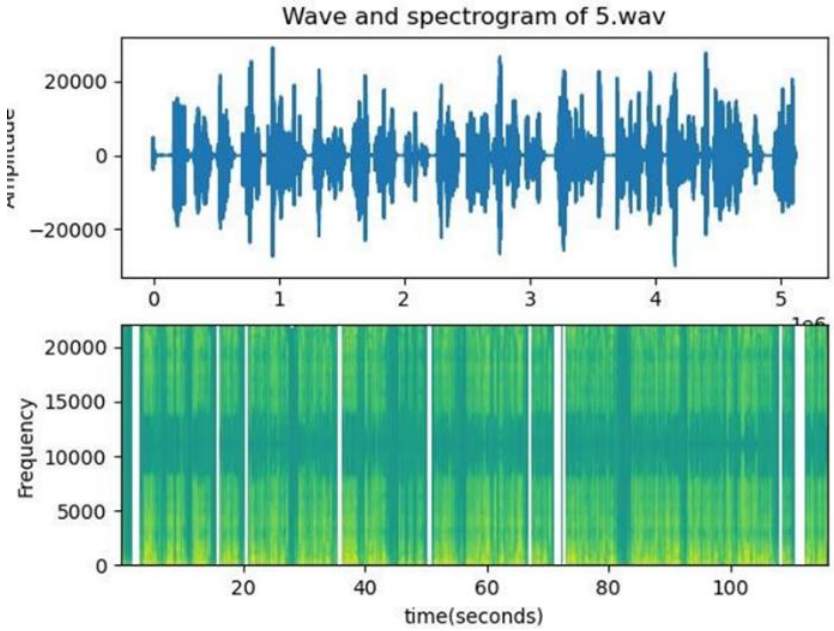

Fig 18: Spectrogram

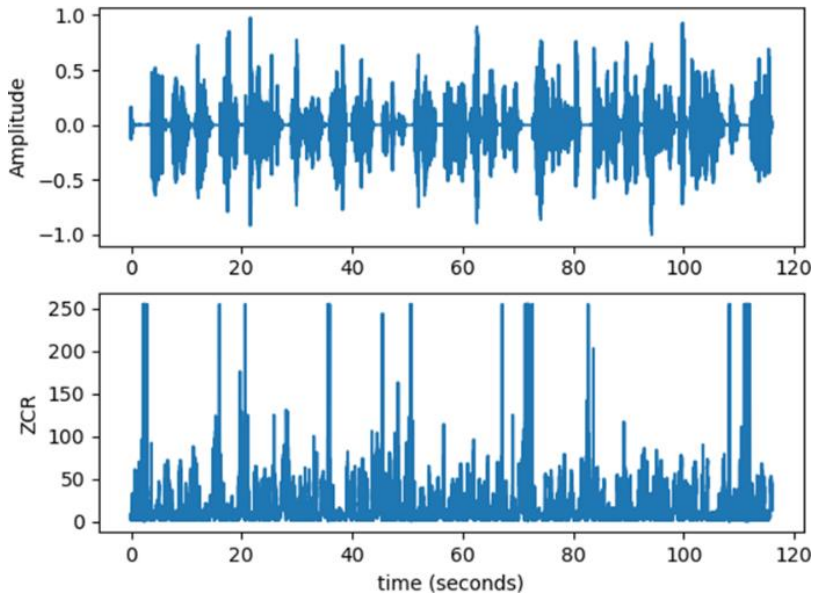

Fig 19: ZeroCR

Fig 20: English to Chinese

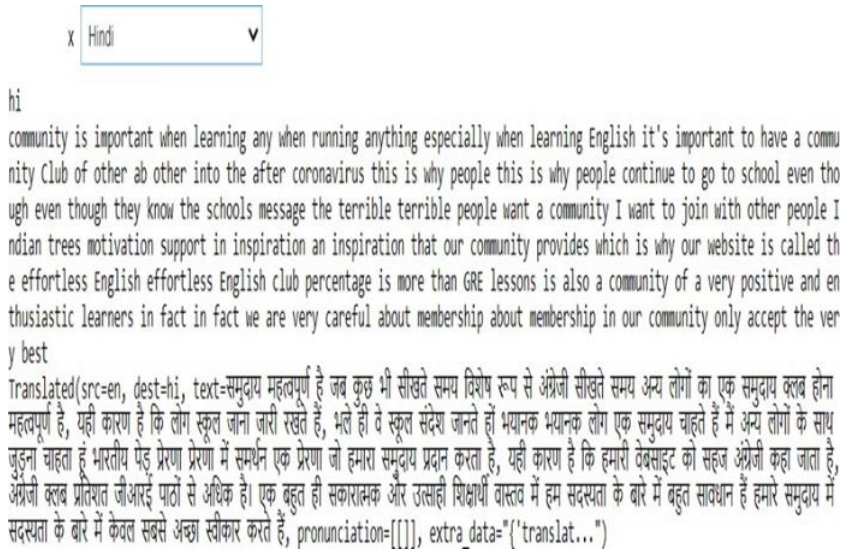

Fig 21: English to Hindi

Text to Text Conversion

In text-to-text conversion google translator is used. The function translator consists of three parameters one those are

1. Text: the text which is to be translated is specified
2. SRC: this is the source language
3. DST: this is destination language The text is converted English to Telegu:

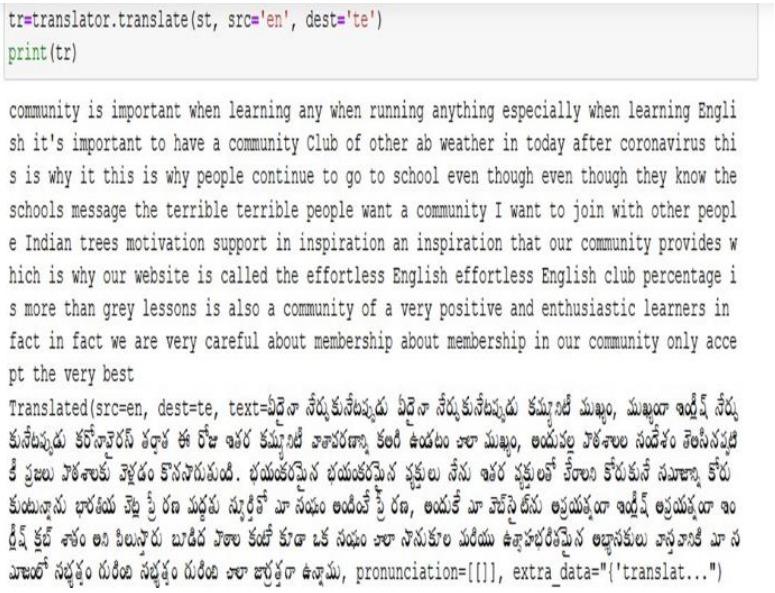

Fig 22: English to Telugu

The text is converted English to Irish:

```
community is important when learning any when running anything especially when learning Engli
sh it's important to have a community Club of other ab weather in today after coronavirus thi
s is why it this is why people continue to go to school even though even though they know the
schools message the terrible terrible people want a community I want to join with other peopl
e Indian trees motivation support in inspiration an inspiration that our community provides w
hich is why our website is called the effortless English effortless English club percentage i
s more than grey lessons is also a community of a very positive and enthusiastic learners in
fact in fact we are very careful about membership about membership in our community only acce
pt the very best
Translated(src=en, dest=ga, text=tá an pobal tábhachtach nuair atá siad ag foghlaim rud ar bi
th agus iad ag rith aon rud go háirithe agus iad ag foghlaim Béarla tá sé tábhachtach go mbea
dh Club pobail ab aimsire eile ann inniu tar éis coronavirus agus sin an fáth go leanann daoi
ne orthu ag dul ar scoil cé go bhfuil teachtaireacht na scoile ar eolas acu tá na daoine uafá
sacha uafásacha ag iarraidh pobail ba mhaith liom a bheith páirteach le daoine eile Tacaíocht
spreagtha crainn Indiach mar inspioráid inspioráid a sholáthraíonn ár bpobal agus sin an fáth
ar a dtugtar ár suíomh Gréasáin mar chéatadán an chlub gan iarracht Béarla Béarla gan iarrach
t ná gur ceachtanna liath é pobal de foghlaimeoirí an-dearfacha agus díograiseacha i ndáiríre
i ndáiríre táimid an-chúramach faoi bhallraíocht faoi bhallraíocht inár bpobal ach glacadh le
is an gcuid is fearr, pronunciation=[[]], extra_data="{'translat..."
```

Fig 23: English to Irish

Conclusion and future work

Conclusion

In this project, we have implemented Speech to text conversion. In this project the Google API used to convert speech to text using the microphone in English audio file. This can be useful in natural language processing projects for handling audio files and transcripts as well.

The purpose of the project is achieved with expected results and the speech to text conversion is also done in efficient manner.

Future Work

This project is mainly concentrated in converting the English audio files to other recognized languages. This project can be extended to convert any audio file to any text. This project can be implemented using other techniques like deep learning and Multilayer perceptron's etc.

One more important extension for this project is instead of audio to text conversion, this con be extended for converting video file to text file.

References

1. Vimala C, Radha V. A Review on Speech Recognition Challenges and Approaches. World of Computer Science and Information Technology Journal (WCSIT). 2012; 2(1):1-7. 2221-0741.
2. J Baker. The DRAGON system–An overview. IEEE Transactions on Acoustics, Speech, and Signal Processing. 1975; 23(1):24-29.
3. J Baker, L Deng, J Glass, S Khudanpur, CH Lee, N Morgan, D O'Shaughnessy. Developments and directions in speech recognition and understanding. IEEE Signal Processing Magazine. 2009; 26(3):75-80.
4. P Douglas, JM Baker. The design for the Wall Street Journal-based CSR corpus. In Proceedings of the Workshop on Speech and Natural Language, 1992, 357-362.
5. M Bacchiani, F Beaufays, J Schalkwyk, M Schuster, B Strope. Deploying goog411: early lessons in data, measurement, and testing. In Proceedings of ICASSP. 2008, 5260-5263.
6. CH Lee, FK Soong, KK Paliwal. Automatic speech and speaker recognition: advanced topics. Springer, 1996.
7. N Sharma, S Sardana. "A real time speech to text conversion system using bidirectional Kalman filter in Matlab," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, 2016, 2353-2357, doi: 10.1109/ICACCI.2016.7732406.
8. https://www.analyticsvidhya.com/blog/2019/07/learn-build-first-speech-to-text-model-python/
9. Vlado Delić, *et al*. "Speech Technology Progress Based on New Machine Learning Paradigm" Volume Computational Intelligence and Neuroscience, 2019.
10. Ahad A, Fayyaz A, Mehmood T. Speech recognition using multilayer perceptron. 2002; 103-109:1. 10.1109/ISCON.2002.1215948.
11. https://heartbeat.fritz.ai/a-2019-guide-for-automatic-speech-recognition-f1e1129a141c
12. MA Anusuya, SK Katti. Speech Recognition by Machine: A Review (IJCSIS) International Journal of Computer Science and Information Security, 2009, 6(3).
13. R Sultana, R Palit. A survey on Bengali speech-to-text recognition techniques, 2014 9th International Forum on Strategic Technology (IFOST), Cox's Bazar, 2014, 26-29, doi: 10.1109/IFOST.2014.6991064.
14. K Nguyen, T Ng, L Nguyen. Adaptive boosting features for automatic speech recognition, 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, 2012, 4733-4736, doi: 10.1109/ICASSP.2012.6288976.