

Speech Based Animal Name Translator

Mahasin Hossen Munshi(224101035)

Mayukh Das(224101036)

Rahul Asati(224101042)

Introduction

We have developed a Speech recognition Project on English to Hindi Animal name translator. This application is intended for children users especially them who are not aware of Hindi translation of common English Words. For our application we have taken animal names which will be given as voice input and our application will respond with a **voice(hindi translate),image,little paragraph** about the animal. The media response of our application is used for attracting more children users.

Technology Used

We have used the HMM based modelling for our word recognition. For Universe building we have used Durbin's Algorithm of Auto-Correlation and used Cepstral vectors as the feature vector. Then We have used LBG algorithm to build the codebook and finally vector quantization to produce the observation sequence. This observation sequences are training sequences for HMM.

In Front End We have tried to make it very simple and used Visual C++ to design UI based on Windows Forms Applications.

Vocabulary and Parameters used

Vocabulary size = 10

Vocabulary = ["cat", "dog", "horse", "lion", "tiger", "monkey", "rabbit", "goat", "cow", "zebra"]

Training utterance count of each word = 40

(Cepstral vector dimension) $p = 12$

Frame Size = 320

Overlapped frames = 80

Codebook Size(M) = 32

T (Maximum observation sequence length) = 160

N (Number of hidden states) = 5

Working Flow

-For Pre-Training

- Record 40 utterances for each of 10 words . This is our training data
 - Build universe of cepstral vectors using durbin's algo auto correlation method from our Training data
 - build codebook from Universe using LBG algorithm
 - Perform Vector quantization on each of digit and it's 40 utterances to form the observation sequence using the codebook and universe and using tokhura distance as distance measure
 - After having the observation sequences of each digit utterances we can do the training using HMM
 - We use 40 word utterances for training
 - for each word we will iterate for training_iterations=3 number of times
 - In 1st iteration we will use the initial biased model for every 40 utterances
 - In subsequent iterations we will use the converged model as initial model from previous iterations
 - After every iteration we will get average of 40 converged models
 - In the final iteration our final model for a word will be the average of the 40 converged models of final iteration

-For Live Testing

Cropping(Start and End marker)

First we have calculated the room's silence **ste**. And used it as threshold.

For the recorded audio(of 3 sec) we have cropped out the clip having ste greater than the threshold by some factor and stored it in another file. We have used the same procedure as training for building the

observation sequence for live utterance of cropped data . Finally we used **forward procedure** to calculate **$P(O/\lambda)$** for each word's model and output the word for which **$P(O/\lambda)$** is **maximum**.

-For Live Training

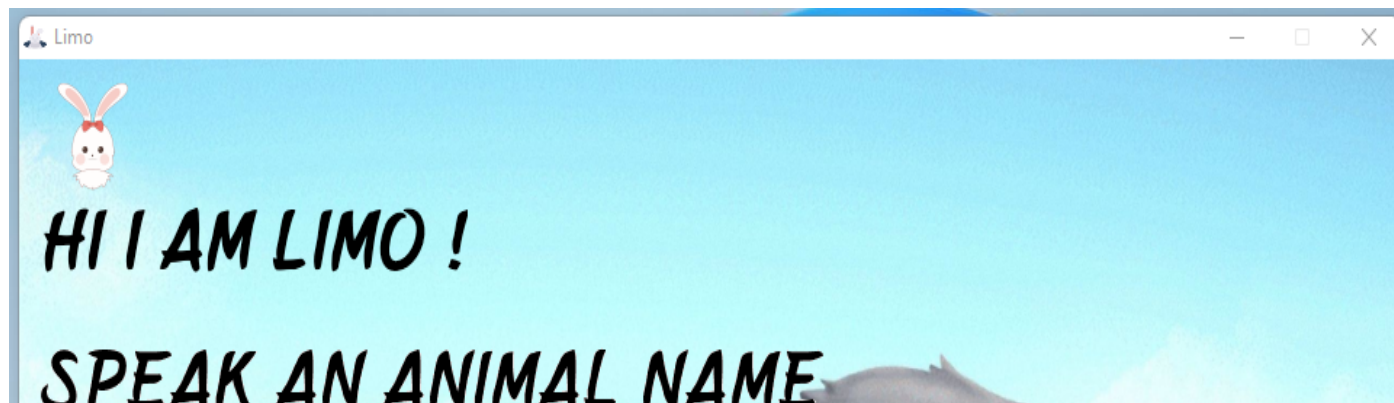
In the front end we accept Speaker name ,number of words(vocabulary size) and number of utterances for each word in a form.A model with the Speaker name is build and stored and he/she has to provide “**word label**” for each recorded word. Once the dataset is build. We train the model in similar fashion as we did for Pre-Training here number of words, word labels and number of utterances is variable and given by user.

Challenges Faced

We were facing issues to properly distinguish the speech data from the noise due to which our live test accuracy was falling down although We have a training accuracy of more than 95%.
But Still we managed to get a live test accuracy of near about 65-75% after applying our cropping algorithm

Snap Shots of our Application

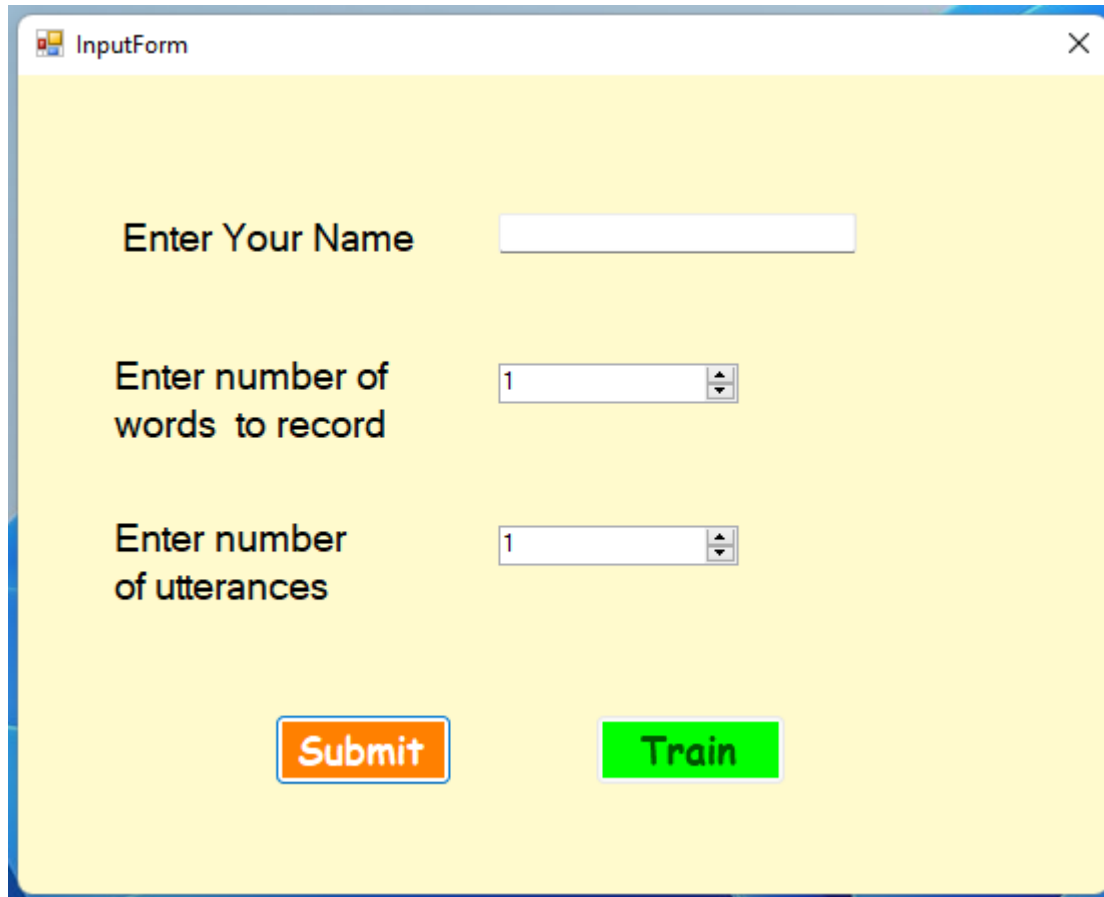
Welcome Page



Application Response



Live Training

A screenshot of a software window titled "InputForm" with a close button (X) in the top right corner. The window has a yellow background and a blue border. It contains three input fields: a text box for "Enter Your Name", a spinner box for "Enter number of words to record" (set to 1), and another spinner box for "Enter number of utterances" (set to 1). At the bottom, there are two buttons: an orange "Submit" button and a green "Train" button.

InputForm

Enter Your Name

Enter number of words to record

Enter number of utterances

Submit **Train**