Speech recognition

by Adithya Aravind

Submission date: 23-Oct-2021 10:20AM (UTC+0530)
Submission ID: 1681704531
File name: PW22UD02_Phase_Two_report.pdf (1.64M)
Word count: 8512
Character count: 43361

CHAPTER 1

INTRODUCTION

In the new age of the digital era, the majority of the world has access to smartphones but not all are able to fully utilise the functionalities these devices provide-one of them being a personal voice assistant. This has become a prevailing issue in India among people who aren't well versed in English. With the help of our product, we aim to bridge the communication gap between this community and a voice assistant that can understand only a monolingual query.

The domain of recognising and translating multilingual speech is still a very less researched topic but there are a few top contenders who lead the market in translating and recognising monolingual sentences. Apart from the fact that these tools are not currently able to efficiently and accurately handle a mixture of multiple languages in a single speech or text query, there are other drawbacks to the current models and tools.

Here is a look at the top models for speech recognition and the limitations of them.

1.1 Top 5 Speech recognition APIs currently in use:

1. Google Speech API
2. IBM Watson API
3. SpeechAPI
4. Speech to Text API
5. Rev.AI API

1.2 Google API working and drawbacks

When the input language is chosen as English, the translator aims to translate the words using the English dictionary vocabulary. Thus, when another language (for example Kannada) is used in the sentence, that foreign word is mapped to the closest sounding English word irrespective of the meaning.

For example: When the input speech is ‘How to go to Shaale’ all the English words are recognised correctly but Kannada word ‘Shaale’ is mapped to the closest sounding English word ‘Charlotte’.

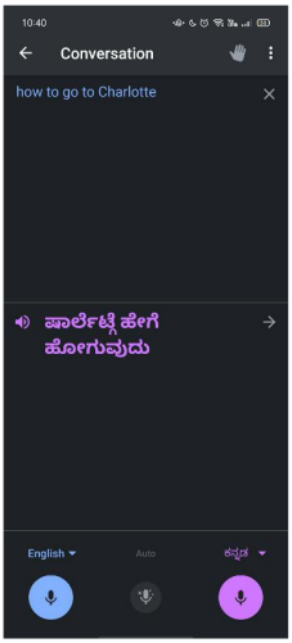

Figure 1 English to Kannada translation

When the input language chosen is a regional language (other than English), the translator recognizes all the words of that particular language correctly but if there is any other word of a different language present in the sentence, that foreign word is mapped to English or input regional language’s vocabulary.

Thus, when two regional languages are used for example if Kannada and Telugu are present in the speech and Kannada is selected as an input language, then all the Kannada words are recognised correctly whereas Telugu words are mapped to the closest English or Kannada word. For example: The given input speech was ‘Saaku nayigalu nange ekkada dorukutundi’ , a sentence consisting of Kannada and Telugu. Since input language is chosen as Kannada all the Kannada words are correctly recognised but the Telugu words are translated incorrectly.

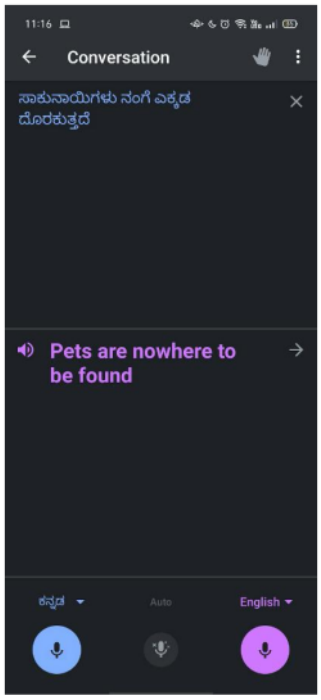

Figure 2 Multilingual translation containing two regional language

1.3 Testing existing APIs

To analyse the performance of speech to text APIs following tests were run.

1.3.1 Google Translate API

The input provided for this API was an audio file containing the multilingual sentence and also the list of languages present in the statement.

Expected output was the converted multilingual text from the audio file but however google cloud requires the user to provide their credit card credentials, hence the test case failed to execute. Another drawback for the API to work perfectly is that it is necessary to give the languages present in the sentence which is not ideal in the real-world scenario.

```python
speech.py > ...
from google.cloud import speech_v1p1beta1 as speech
import speech_recognition as sr
recognizer = sr.Recognizer()

client = speech.SpeechClient()

speech_file = "Recording.wav"
first_lang = "en-US"
second_lang = "kn-IN"

with open(speech_file, "rb") as audio_file:
    content = audio_file.read()

audio = speech.RecognitionAudio(content=content)
config = speech.RecognitionConfig(
    encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
    sample_rate_hertz=44100,
    audio_channel_count=2,
    language_code=first_lang,
    alternative_language_codes=[second_lang],
)

print("Waiting for operation to complete...")
response = client.recognize(config=config, audio=audio)
```

Figure 3 Google Translate API

1.3.2 Inbuilt Python Module

For speech to text conversion of the multilingual sentence inbuilt python libraries were used. The output could properly recognise only English words in the given sentence.

```
In [1]: import speech_recognition as sr

In [11]: filename = "54_3.wav"

In [12]: r = sr.Recognizer()

In [13]: with sr.AudioFile(filename) as source:
             # listen for the data (load audio to memory)
             audio_data = r.record(source)
             # recognize (convert from speech to text)
             text = r.recognize_google(audio_data)
             print(text)

         MG Road reach Agar ks2 time aguthe
```

Figure 4 Python Module

1.3.3 Detecting languages present in the multilingual sentence

Python functions were used to detect the languages present in the textual multilingual sentence. This function could properly recognise English words however in most cases failed to recognise regional languages.

```
In [4]: kannada
Out[4]: ['How to hang painting on gode',
         'Gode mele hege painting hang maadodhu',
         'Gode mele painting hege nethaakadhu',
         'Wall mele painting hege nethaakadhu',
         'How to grow apple mara',
         'Apple mara hege belsadhu',
         'Apple tree hege belsadhu',
         'Sebu mara hege grow maadodhu',
         'Sebu tree hege tree maadodhu',
         'How to weave a butti',
         'Butti na hege weave maadodhu',
         'Basket na hege weave maadodhu',
         'Parivala ge hege feed maadodhu?',
         'how to feed Parivala',
         'Pigeon ge hege thinsodhu']

In [7]: lang=TextBlob('goode')
        print('Seebu',lang.detect_language())

        Seebu en

In [5]: for i in kannada:
            i=i.split()
            for j in i:
                try:
                    lang = TextBlob(j)
                    print(j,lang.detect_language())
                except:
                    print('')

        How en

        hang en
        painting en

        gode da
        Gode da
        mele ro
        hege zh-CN
        painting en
        hang en
        maadodhu kn
        Gode da
        mele ro
        painting en
        hege zh-CN
        nethaakadhu ml
        Wall en
        mele ro
        painting en
        hege zh-CN
        nethaakadhu ml
        How en
```

Figure 5 Language Detection

To address these drawbacks and issues to bridge the multilingual communication gap, we need a model that can accurately recognise the different languages used in the speech, translate it to a single language for convenience and feed the query to a tool that will provide the appropriate output. We aim to do this by building an app for all these requirements.

CHAPTER TWO

PROBLEM STATEMENT

The growing need to fully utilise the functionalities of smart devices such as voice assistants in phones is very essential in this tech-savvy era. To bridge the communication gap between the non-English speaking people and the new technologies, the use of multilingual recognition for voice assistants is a must.

The full functionality of the assistant can be utilised by all people regardless of their fluency in English or one particular language and thus expands the use of this technology and improves the lives of many.

The models and tools currently available in the market are not efficiently bridging the communication gap in the multilingual domain. Some of the situations where a need for a better tool is required are as follows:

- Translating multilingual sentences from professors who are non-native or who do not speak fluent common languages, so that students can better understand the lecture and explanation offered by the faculty in universities abroad and in India. This use case works both ways where a student who cannot speak the common language fluently can speak with preferred language with ease and translated to professors to better understand the student's difficulties.
- The other use case is translating instructions such as directions, requests, etc between non- locals.

In order to accurately solve the above issues and drawbacks, we have formulated a problem statement to tackle these limitations and solve the problem of multilingual speech recognition which is mentioned below.

Recognizing the various languages used in a multilingual sentence and translating it into a single language sentence.

Developing a voice assistant that comprehends multilingual voice navigation queries by recognizing the various languages used in a multilingual sentence and reply accordingly with a single language sentence.

Our Project Scope involves:

- Taking input as an audio file containing a query sentence with a mixture of languages including English with Kannada.
- Recognising the different languages used in the query.
- Converting the query to a single language of convenience.
- Feeding the translated sentence to an existing API which will generate appropriate output for the query.

- Integrating all the above functionalities into a user-friendly app.

CHAPTER 3

LITERATURE REVIEW

3.1 Paper 1:

Multilingual Speech Recognition with a single end-to-end model - Shubham Toshniwal, 15th February 2018

3.1.1 Introduction

The paper describes a new method for Acoustic Speech Recognition (ASR) using a sequence-to-sequence model rather than the traditional models, as the conventional models highly depend on the lexicons, word units and the word inventories of the languages.

The paper trains a single sequence ASR model in 9 different Indian languages, which have very little overlap scripts. There is a joint ASR model which learns through the outputs of each of these single ASR models and it is found that without external information being provided, the model improves recognition of different languages by 21% compared to the analogous ASR models.

3.1.2 Methodology

The paper talks about the ASR sequence to sequence Listen, Attend and Spell (LAS) model.

The model consists of three main parts - the encoder, decoder and an attention network which are trained together and fed with the audio frame sequence input and analyses graphemes from the input sequence.

The model collects audio features every ten milliseconds in a window frame of twenty-five milliseconds, thus leading to a eighty-dimensional logarithmic audio factor. The eight sequences

of continuous features are collected and framed together to serve as an input. This sequence frame is stepped down by a factor of three to serve a simpler computation purpose.

The encoder part of the LAS model acts similar to a typical ASR model where it is fed with an audio stack sequence of inputs a = (a1,a2,a3…. ak) and gives the output as a set of high level hidden variables o = (o1,o2,o3...ok). The encoder is a Recurrent Neural Network model which takes audio sequence stack as the input.

The next part of the model is the decoder which is a monodirectional stacked RNN which computes the probability of a stack of variables z.

3.1.3 Drawbacks and Weaknesses

Since the LAS model is trained separately on each language and then compared with the joint LAS model for all languages, the self-learning model gets confused between the languages. The lack of script overlapping Indian languages is a very huge set-back to this model.

In theory, the joint LAS model should be able to switch codes for other languages, but the model is specified to learn certain traits of the acoustic only which tends to overfit the model as a whole.

This model learns on its own using internal representations of languages and uses it to distinguish among different languages, this is a problem as it overrides the fundamental difference between the languages in reality.

3.1.4 Conclusion

The paper gives us a methodology which is better than the standard sequence to sequence audio language recognition models. The paper also talks about variants of the LAS model which can be used to better recognize monolingual audio sentences without the explicit mentioning of the language being used. The paper has some more scope in terms of research regarding code

switches and their use in further enhancing single language recognition models. But overall, the paper has found conclusive evidence of a better LAS model which rarely gives the wrong result.

3.2 Paper 2:

Multilingual Text-to-Speech Software Component for Dynamic Language Identification and Voice Switching, September 2016 Paul Fogarassy, Costin Pribeanu

3.2.1 Introduction and summary

This paper aims in the process of implementation of a software that will help in the text-to-speech feature for applications that are developed for people who are visually challenged or have reading disabilities. Language recognition is an important feature that is required for multilingual text-to-speech conversion. The algorithms used in this process are different from those used in automatic language detection, since the recognition is done non synchronously on a continuous stream of texts.

3.2.2 Methodology

N-gram frequencies are statistically analysed for language identification. The main idea behind this is that for any given particular language there are n-grams that are present more often than others and based on these languages can be identified. However, for this to work the language corpus must be analogous and grammatically correct. The quality of the corpus is really important for great accuracy of language identification.

The multilingual text to speech is distinguished into four categories which are mixed lingual with pre-defined language,monolingual,polyglot with language detection and simple multilingual.

In the first case, the system first detects non-native words, then will understand the process of pronunciation and intonation.

In the next case, non-native words are built with available voices.

The third case involves detection of language with the help of multilingual text analysis and then generating speech using voiced and toned models.

The last case is where language change is detected and the model switches to appropriate voice.

3.2.3 Implementation

Software component

The development cycle involves steps, the alpha version which will help to understand proof of the concept, commercial version, functional version and implementation of different existing applications.

Alpha version

The main aim of the alpha version was to test the language identification algorithms. The text here will be written in one language and the method is based on tri-grams frequency.

Beta version

Beta version allowed users to select candidate language and desired synthetic voices. The trained model is then used to perform analysis for the given input corpus and it will then output tri-gram frequencies.

Commercial version

The application interface was used as a bridge between speech engine modules and custom applications. The iT2V component is used to identify the language of the input text and later correctly select the speech engine for that particular language.

Implementation version

The implementation of this model was successfully done in two assistive applications: the Digital organizer Pronto and Reading machine POET.

3.2.4 Conclusion

In this paper, the software component of multilingual text-to-speech has been presented. The results of the beta version showed that for language detection algorithms, fragmentation of a piece of text is an important parameter. Tri grams provided better accuracy in language recognition as compared to single or bigram. Since most of the users of this application are visually challenged, manually changing voice in the audio menu by following voice guidance was difficult and really time-consuming.

3.3 Paper 3 :

AUTOMATIC LANGUAGE IDENTIFICATION USING DEEP NEURAL NETWORKS,2016, Ignacio Lopez-Moreno , Javier Gonzalez-Dominguez, Oldrich Plchot

3.3.1 Introduction

This paper examines the performance of deep neural networks on the problem of Language identification. This deep neural network works on the features extracted from short speech utterances. This paper also describes how the proposed model outperforms the state-of-the-art i-vector based acoustic model. It is found out that when the data-set is large, the deep neural networks perform the language identification better

3.3.2 Methodology

I-vector based language identification:

State-of-the art language identification acoustic model systems are built based on I-vectors. Feature vectors are obtained for every window of length 25ms with a frame rate of 10ms. Energy based voice activity detectors are used to filter out the silent framers. Different classification techniques were used after the I-vectors were extracted for every language. Linear logistic regression, which is a discriminative model, is tested. Generative models were also tested such as linear discriminative analysis and cosine distance is used as a similarity measure. The Gaussian model with 1G and 2G components is used to fit the I-vectors of every language.

DNN based language identification:

The proposed architecture of the deep neural network is a fully connected feed forward network. Rectified linear units (ReLU) are used as activation functions in the hidden layers. The output layer uses softmax activation function and cost function used is cross-entropy loss. Each hidden layer consists of 2560 neurons and the number of neurons in the output layer is equal to the number of languages and an extra neuron is added as an out of set language. This deep neural network works with frames.21 frames which are formed by stacking the current processed frame and ±10 left-right contexts are fed to the input layer. Asynchronous stochastic gradient descent is used with a batch size of 200 samples and the learning rate is fixed as 0.001.

Two different performance metrics were used in-order to compare the proposed DNN to the I-vector based approach. C(avg) (Average cost) which is defined by LRE 2009 and Equal Error Rate (EER) are used metrics. It is found out that on the Google 5m Language Identification Corpus, proposed 8-hidden layer DNN architecture achieved close to 70% improvement on the C(avg) compared to the I-vector based gaussian modelling. On the Language Evaluation 2009 Data-set, the proposed model showed a relative improvement in equal error rate with respect to linear discriminant analysis based on I-vectors.

3.3.3 Drawbacks

It is found out that if the training amount of data is less than 10 hours per language, the I-vector models perform better than the deep neural network. DNN has many hyper-parameters and it is difficult to obtain the best values for these.

3.3.4 Conclusion

In this paper, the capability of DNN's to learn discriminative language information from speech signals is explored .The DNN outperforms the state-of-the-art models in most cases. This is when the training data for each language is more than 20 hours. The performance of I-vector based approaches will start to saturate when the training data starts to exceed 20 hours.

CHAPTER FOUR

SOFTWARE REQUIREMENTS SPECIFICATIONS

4.1 Product Perspective

4.1.1 Product Features

The end deliverables and features of the product include the following :

- Recognising the different languages present in the speech query with the help of ML and NLP techniques and algorithms.

- Converting the comprehended words into a single language and framing a semantically correct sentence.
- Feeding the single language query to an existing API which will generate the appropriate output.
- Converting the obtained output into a single language voice output of user's convenience.

4.1.2 Operating Environment

The operating environment required for a regional voice assistant are :

- Database for storing the queries
- API for generating output for the queries.
- ML and NLP model to recognise the different languages used in the query.
- Text to speech API to generate audio output.

4.1.3 General Constraints, Assumptions and Dependencies

Following are the depends, constraints and the assumptions attributed to the regional voice assistant:

- Using existing APIs to generate the query's output once it is translated to a single language sentence.
- ML/NLP algorithms such as character N-grams
- There is an assumption that API will generate an appropriate output to the query.
- Constraint on daily requests the existing API can handle.

4.1.4 Risks

The risks involved in the working of the regional voice assistant are as follows :

- The potential error can be caused by having noise in the audio data input which can affect the accuracy of the model.
- No governance over the usage of unethical language as an input for the model

- The accuracy of the output entirely depends on the working and accuracy of the existing API used to feed the input query to.

4.2 Functional Requirements

4.2.1 Validity tests on input :

- The speech shall be recorded using the embedded speaker on the smartphone.
- It is assumed that the user speaks in languages that the model can decode.
- The input audio is expected to contain some speech rather than random noises.
- The input speech shall be a simple command or a query in the form of only a single sentence.
- The speech query shall be less than 10 sec.
- It is expected that the user pronounces the words accurately.

4.2.2 Error Handling and recovery :

- In case if the model cannot understand the words, the user is prompted to speak again.
- Suppose if the necessary output can not be generated, google search results shall be displayed accordingly as per the user's query.
- If any request to the cloud API is failed, it shall be re-attempted.
- If the user specifies that the desired output is not obtained, an option to query again shall be given.

4.2.3 Consequences of parameters:

- Random disturbances and background noise may result in undesirable working of the model.
- Good internet connectivity is required for fast processing of query.

4.2.4 Relationship of output to input:

- Appropriate speech output is given to user as per the speech input query

4.3 External Interface Requirements

4.3.1 User Interfaces

A simple interface is presented to the user, where the user makes use of the 'record' option to allow the device to start capturing the speech query. A processing/loading animation shall be displayed while the input is being processed.

After the processing, the output shall be given out in the form of audio. The text for the same will also be displayed on the screen. User is also presented with a record again option and the same process continues. Appropriate error messages will be displayed whenever necessary, such as instances where the model can't decode the audio properly, connection timeouts etc.

4.3.2 Hardware Requirements

- Devices that run Android version 8.0 or above and iOS 12.0 or above are supported.
- Communication with the cloud API's is through standard HTTP/HTTPS protocols.

4.3.3 Software Requirements

Flutter:

This is an open-source UI software development kit that allows for easy development of applications for Android and iOS version : 1.20.6

Google Cloud Platform:

Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products. This shall provide API for necessary translations and the model can be hosted on this platform.

4.3.4 Communication Interfaces

- Mobile microphones are used to give audio input where the audio from the user is converted to an electric signal and transferred to the processor through serial ports.
- The multilingual audio data from the user is converted to bytes data and given as an input to cloud APIs through standard HTTP/HTTPS protocols.
- The single language output from the cloud API is communicated to the mobile application through HTTP/HTTPS protocols in the form of byte data
- The final byte data is converted to mp3 format and transferred from mobile's processor to speaker with a serial port and outputted to the user

4.4 Non-Functional Requirements

4.4.1 Performance Requirement

- The model must be deployed in cloud infrastructure since it must always be available to the users.
- The endpoints in the mobile application must be such that it can accommodate any new changes in the technology for adaptability reasons.
- The cloud API must be able to handle multiple requests at the same time.This is to provide reliable services and to improve efficiency of the application.
- If there is any error in the cloud function the mobile application must request the cloud API again having saved the users inputs to provide reliability.
- Same code and database must be used to build applications in all OS platforms.

4.4.2 Safety Requirements

- The output from the cloud APIs must not contain any words,phrases or sentences that can likely harm any person's emotional sentiment or be offensive to the user.
- A filter check must also be provided to the input such that there is no usage of unethical language

4.4.3 Security Requirement

- The privacy policies of external APIs must be carefully read so that the user's audio data is not at risk.
- The application must ask the user's permission to record audio input and to give output in their personal devices.
- The application must also record audio input only when the user performs the required action(such as pressing record audio button in the application) and not record any audio data otherwise.

4.5 Other Requirements

- The model must be self-learning. As new input queries are asked the model must be able to update itself and improve its accuracy.
- The data must be sufficient enough to train the model such that there is minimum bias and variance.
- The mobile application must get all the required permissions so that the cloud API can get the required input.

CHAPTER FIVE

SYSTEM DESIGN

5.1 Design Considerations

5.1.1 Design Goals

The current existing voice assistant systems cannot recognise regional languages.Thus by integrating regional languages to voice assistant helps in providing these services to non English speaking communities.These personal assistants also fail in recognising multiple language present in a sentence.This newly proposed system will help the assistant to recognise and translate the multiple languages into a single language sentence.

Since the entire backend including the trained models will be deployed in the cloud architecture the services will always be available to use and also be reliable.Security is provided by ensuring that only valid users can have access to the backend.With the help of HTTP/TCP connections and parallel computing quick response time is achieved.Interoperability is ensured using flutter which will help in communication with both android and IOS devices.

5.1.2 Architecture Choices

For the development of mobile application native JAVA along with android studio was initially chosen. However after thorough research it was found that flutter was the better alternative since it provides high reusability of using the same code base to both android and IOS applications. It also provides another feature where any changes made to the code can be

immediately seen in the application without having to rebuild the entire application again thus saving a lot of computing power.

For the database ,Firestore, a NoSQL non relational database was chosen rather than its other SQL database alternatives like SQLite. This was done to ensure easier integration between flutter and firestore since firestore provides inbuilt functions for this process. It also helps in easier querying of data rather than having to code complicated SQL queries.However since this is a non relational database multiple copies of data need to be maintained across various entities.

The cloud architecture will be used to deploy the trained models instead of the traditional client server architecture since cloud provides a reliable service compared to client server models. The cloud can always be made available however that process becomes difficult in the client architecture. The processing and communication speed is also higher in the cloud as compared to the client server architecture. This works more efficiently even in a heavy traffic environment than the client server model.

5.1.3 Constraints, Assumptions and Dependencies

- Using existing APIs to generate the query's output once it is translated to a single language sentence.
- Ml/NLP algorithms such as character N-grams
- There is an assumption that API will generate an appropriate output to the query.
- Constraint on daily requests the existing API can handle.
- The potential error can be caused by having noise in the audio data input which can affect the accuracy of the model
- No governance over the usage of unethical language as an input for the model

5.2 Design Description

5.2.1 Application Model

5.2.1.1 Description

This module describes the functionalities involved after the user records the audio and how the corresponding results are displayed to the user. Once the recording is completed,the following recording is uploaded to firebase storage with its timestamp. The backend then downloads the audio file with the most recent timestamp since that will be the most recent query from the user, deletes that audio file from the firebase storage and preprocess the audio file as required by the model. This preprocessed file is later sent to a speech-to-text model and output query is generated. The query is now converted to a single language with the help of google translate and then the links of top 5 results are taken from google search engine and this is saved in the firestore database. The app now displays this query along with the results for the user and the user can go to the corresponding result link or can delete that result. If the user is not satisfied with the results then they can now select the correct words predicted by the model to later further train the model and the corresponding query will be generated by sending the audio query directly to google translate.

5.2.1.2 Use case diagram

The following use case diagram shows different use cases the user is involved with while using the app. This includes all the elements that are displayed in the app as seen by the user

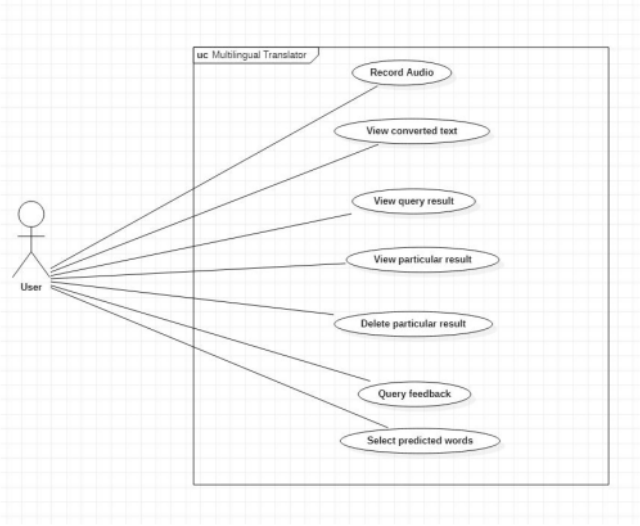

Fig 6 Use Case Diagram for application model

Class Diagram

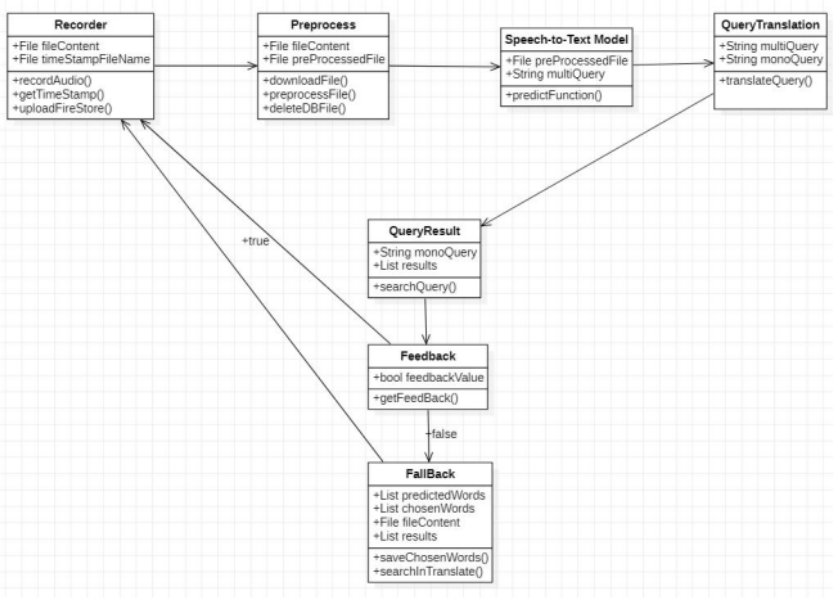

Fig 7 Class Diagram for application model

Class Name 1 :Recorder

Recorder class helps to get the multilingual audio input from the user and save it to the database.

Class Name 2: PreProcess

This class helps in changing the audio file into required format as specified by the model

Class Name 3: Speech-To-Text Model

This class helps in getting the trained model and finding the output of the model

Class Name 4: QueryTranslation

QueryTranslation translates the multilingual query of the user onto single language

Class Name 5:QueryResult

This class shows the top 5 results of the query given by the user

Class Name 6:Feedback

This class helps to get feedback from the user to identify if the user is satisfied with the results or not.

Class Name 7:Fallback

Following class implements the fallback procedure that will happen when the feedback is negative from the user. The predicted words will be shown to the user where the user can select the correct words. Then these words will be sent to the google search engine. If the user isn't satisfied with the predicted words then the entire audio file will be sent to google API.

5.3 Sequence Diagram

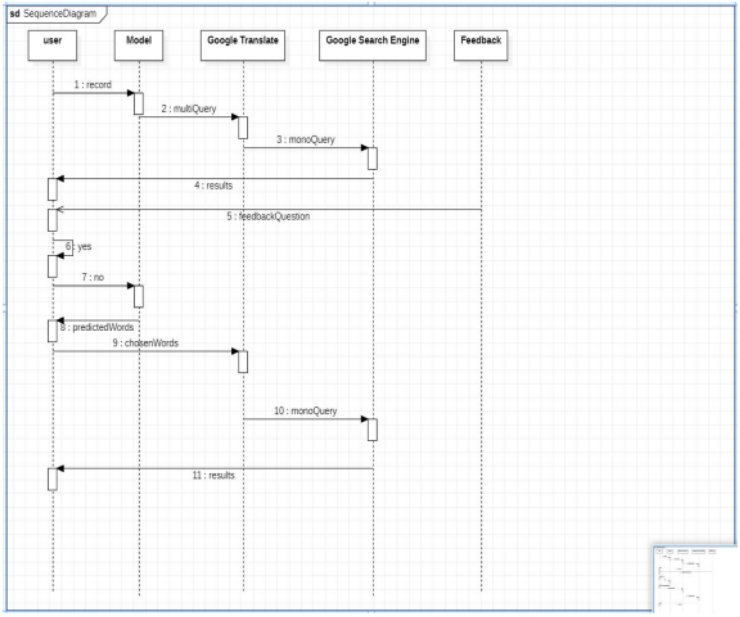

Fig 8 Sequence Diagram for application model

5.4 Packaging diagram

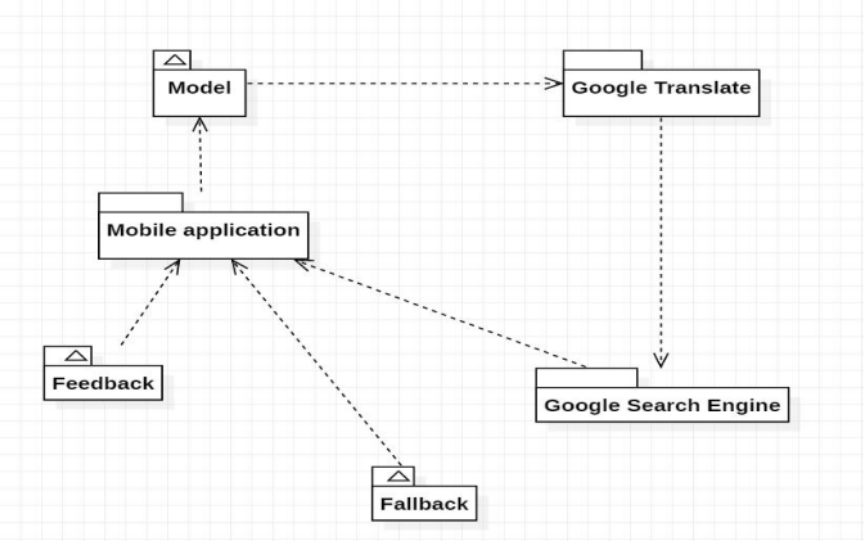

Fig 9 Packaging Diagram for application model

5.5 Deployment Diagram

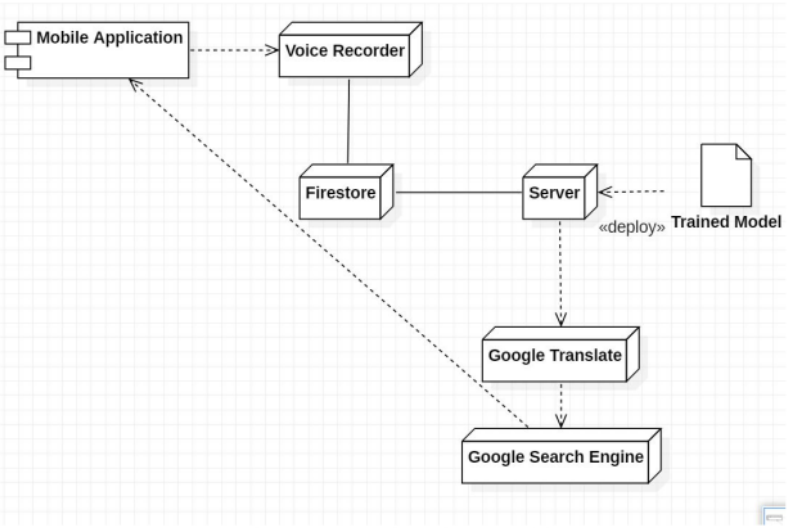

Fig 10 Deployment Diagram for application model

5.6 Splitting Module

5.6.1 Description

This module takes in a speech query as input and splits it into chunks based on word beginning and ending.So the number of chunks is nothing but the number of words in the speech query. These chunks will later be used to predict the uttered word.

5.6.2 Use case Diagram

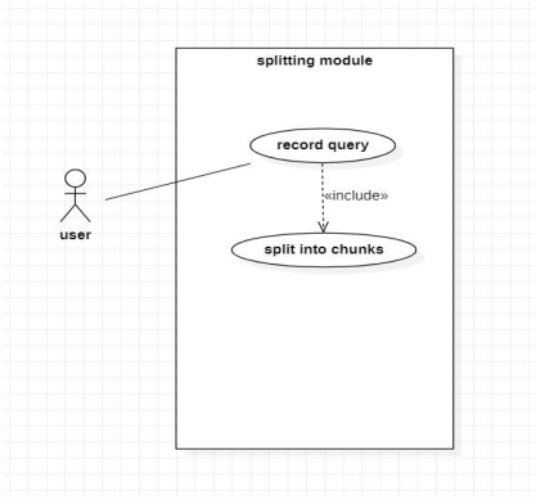

Fig 11 Use Case Diagram for splitting module

5.6.3 Class Diagram

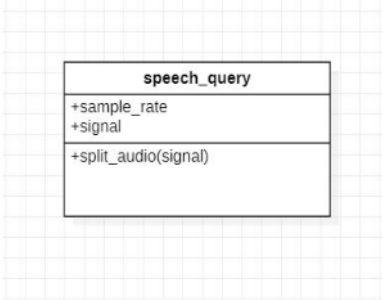

Fig 12 Class Diagram for splitting module

Class Name 1 : speech_query

The objects of this class depicts the signal which is obtained by reading the wav file using librosa library.

5.6.4 Sequence Diagram

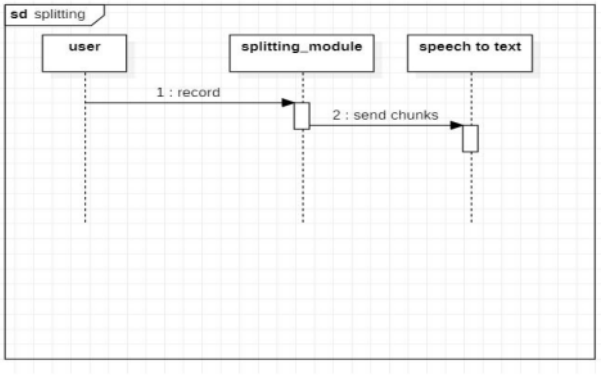

Fig 13 Sequence Diagram for splitting module

5.6.5 Packaging Diagram

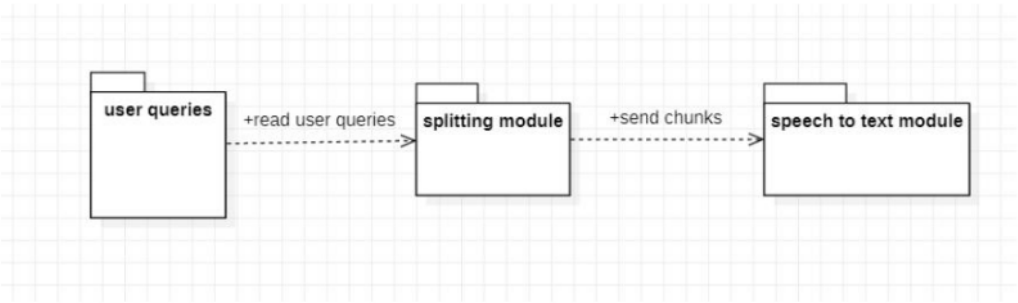

Fig 14 Packaging Diagram for splitting module

5.7 Speech to Text Module

5.7.1 Description

This module is responsible for converting the speech query into text.

5.7.2 Use Case Diagram

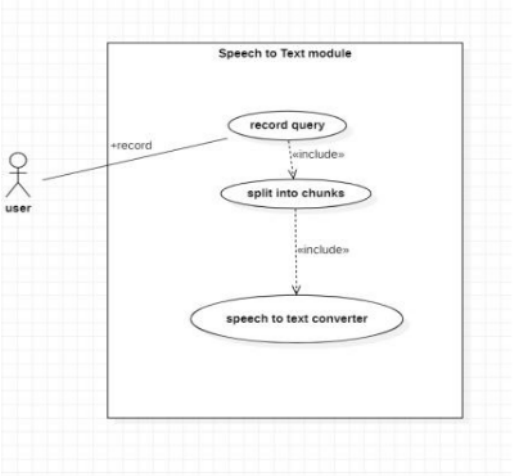

Fig 15 Use Case Diagram for speech to text module

5.7.3 Class Diagram

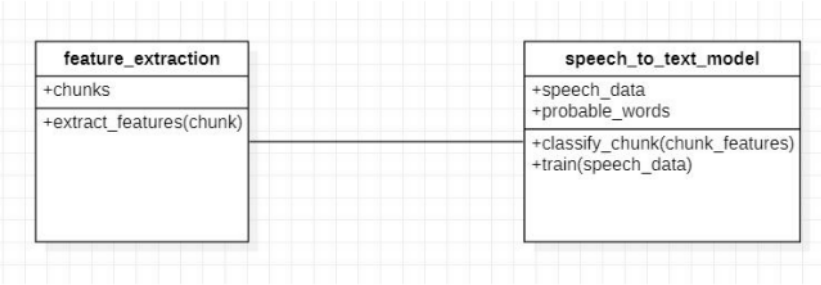

Fig 16 Class Diagram for speech to text module

Class Name 1: feature extraction

Methods

extract_features(chunk)

This method takes each individual chunk from the list of chunks and extracts features for the same. MFCC features for each chunk are extracted using the librosa library.

Input - wav file
Output - list of chunks
Parameters - number of mfcc features
Exceptions - not required
Pseudo-code :

```
for chunk in chunks:
    librosa.extract_MFCC_features(chunk)
```

Class Name : speech to Text model

Methods:

train(speech_data)

Input - all speech recordings
Output - list of chunks
Parameters - None
Exceptions - not required
Pseudo-code :

```
Model = create_NN_model()
Model.train(path_to_directory)
```

classify_chunk(chunk_features)

Input - chunk_features

Output - text format of the chunk

Parameters - None

Exceptions - not required

Pseudo-code :

X = Model.predict(chunk_features)

Return X

5.7.4 Sequence Diagram

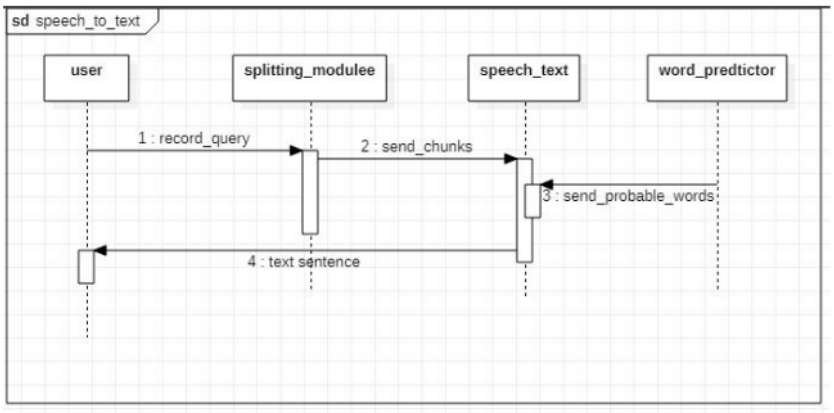

Fig 17 Sequence diagram for speech to text module

5.7.5 Deployment Diagram

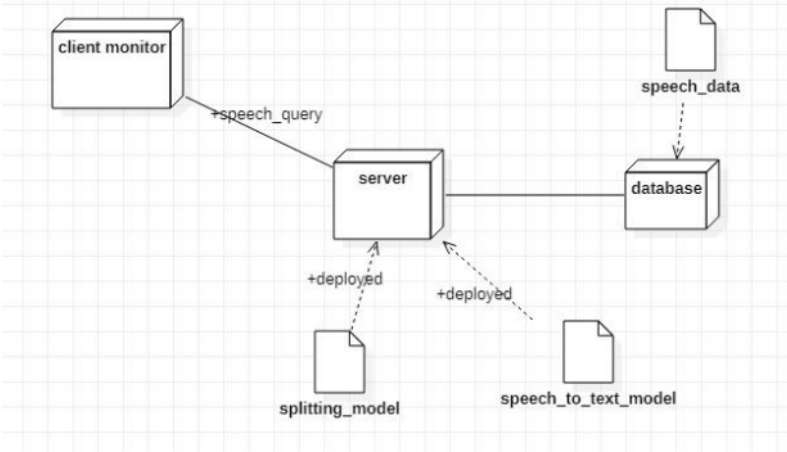

Fig 18 deployment diagram for speech to text module

5.8 Word Prediction Model

5.8.1 Class Diagram

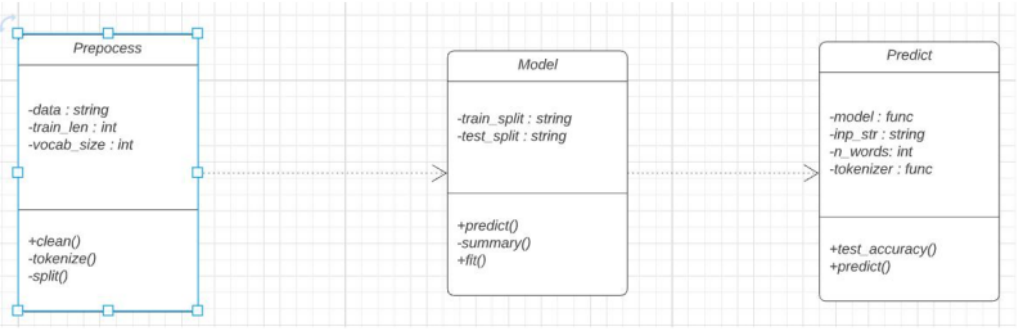

Fig 19 Class Diagram for word prediction model

Class Name 1 : Pre-process

Uses raw input data provided by the user and generates pre-processed, clean and tokenized text.

Class Name 2: Model

This class is the creation of a DNN using packages from keras that will be used to learn and train from the tokenized sequences to find and add weights to important words following prior words and form an accurate learning from the patterns.

Class Name 3 : Predict

This class is used to generate the final output and test the model for accuracy with test data.

5.9 POS Tag Prediction Model

5.9.1 Class Diagram

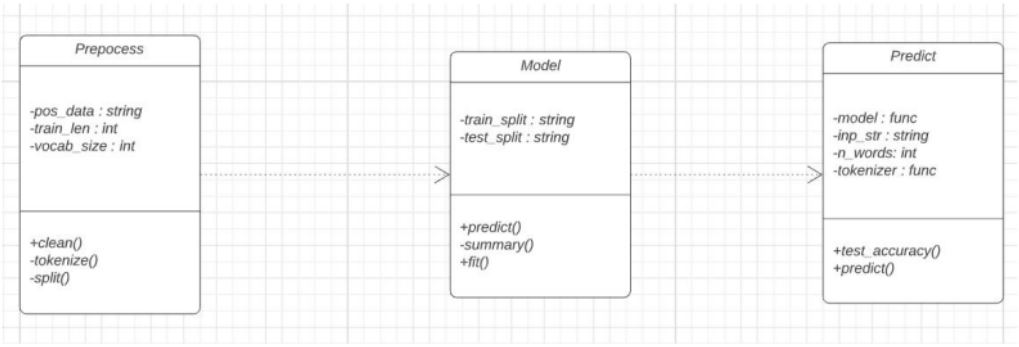

Fig 20 Class diagram for POS tag prediction module

Class Name 1 : Preprocess

Uses POS tagged input data provided by the user and generates preprocessed, clean and tokenized text.

Class Name 2: Model

This class is the creation of a DNN using packages from keras that will be used to learn and train from the tokenized sequences to find and add weights to important words following prior words and form an accurate learning from the patterns.

Class Name 3 : Predict -

This class is used to generate the final output and test the model for accuracy with test data.

5.9.2 Sequence Diagram

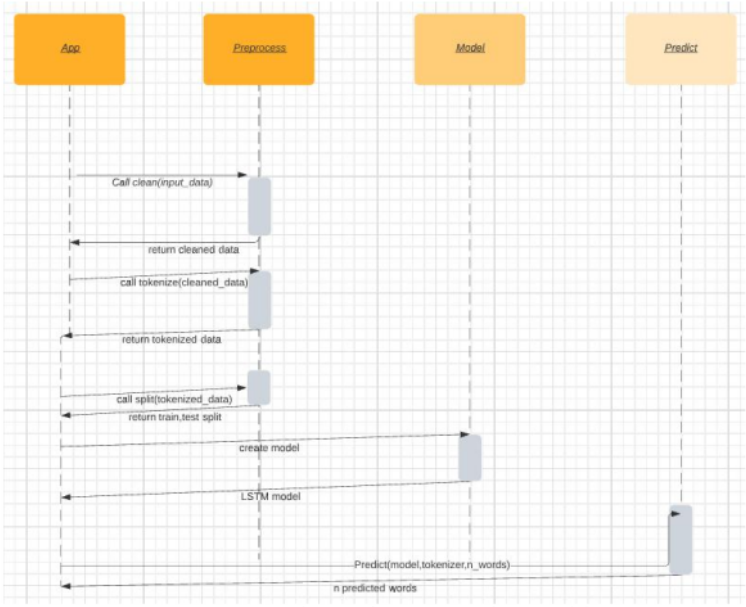

Fig 21 Sequence diagram for POS tag prediction module

5.9.3 Deployment Diagram

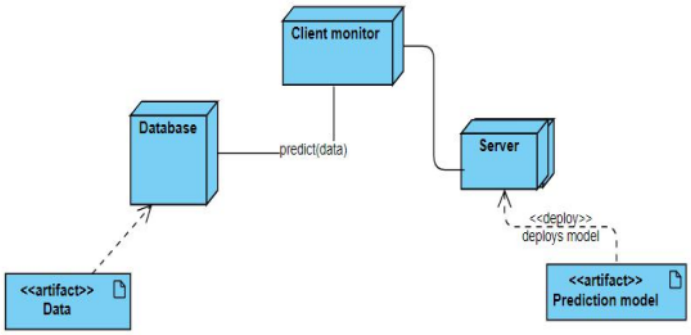

Fig 22 deployment diagram for POS tag prediction module

CHAPTER SIX

PROPOSED METHODOLOGY

The approach used for recognition and translation of multilingual audio query is as follows:

- The user query sentence is split into individual wav files which represent each word in the sentence and these chunks are then pre-processed.
- These pre-processed wav files are then passed to a deep learning model that uses a deep learning model to map the audio to text and generate the text output for the corresponding words.
- The accuracy of the speech-to-text model is further increased using a next-word prediction model and a POS tag prediction model. The deep learning model hence classifies each chunk among the next possible words

- The multilingual text query is passed through to Google Translation API to get the corresponding monolingual query which is then passed to the Search Engine to get the appropriate output.
- The entire process of recording the audio query to display the results is integrated into a user-friendly application.
- To make the prediction faster another approach was developed based on similarity of the waveform of the words. However the accuracy of prediction was less compared to deep learning models.

Proposed Methodology of each module

6.1 Data used for training:

- 384 commonly queryed monolingual questions were generated and for each sentence all possible multilingual sentences (English and Kannada) were created.
- Among 1152 multilingual sentences, 412 navigational queries were selected and tagged with their corresponding Parts of Speech.
- 64 most commonly spoken words among these sentences were chosen to be recorded 10 times by 3 different voices.
- These 1920 voice recordings were used to build the model.

6.2 Preprocessing:

Preprocessing of the audio files were done to obtain clean data and to improve the accuracy of the model. The wav files were converted to a numpy array using librosa and the array was normalised between -1 to 1. The silence factor that existed in the beginning and end of the audio file was removed. The speed of audio files were changed by changing its frame rate to maintain fixed duration across the dataset since each user can speak a particular word at different speeds.

6.3 Splitting of sentence:

After careful analysis of a few recorded sentences, it was found that each individual word utterance was between 15,000 and 25,000 array length. It was observed that the amplitude is low between each utterance of words. Hence amplitude is used as a factor to split the audio file. Moving Average with a window size of 10,000 is used to smoothen the wav file and to clearly identify the minimas. Then, minimas were found in the smoothened signal at a window size of 15,000 array length. Thus when the original signal is sliced at these minimas, individual word utterances are obtained.

6.4 Word Prediction:

Each sentence was appended a starting <s> and an ending <e> tag after which they were tokenized and all n-gram sequences (n = 4) were generated. The first three tokens were considered as features which help in predicting the fourth token. This data was then passed to a LSTM model which can predict the next top 'k' words.

With n = 4, there was ambiguity in predicting the first and second word of a sentence and hence two other models were built in similar manner but with n = 2 and n = 3 respectively.

6.5 Speech to text:

6.5.1 Methodology 1 : Deep Learning

This module receives a set of next possible words(classes) from the word predictor model and classifies the input chunk into one of these words. Pre-processed wav files of these classes were selected to train the model. The extracted mfcc (Mel Frequency Cepstral Coefficient) features were used for training the model.

6.5.2 Methodology 2 : Based on similarity

Similar to the deep learning model, this model also receives a set of next possible words(classes) from the word predictor model and classifies the input chunk into one of these words. Similarity is found between the input chunk and among all the recordings of next possible

words. Highest occurring class among the top 20 most similar recordings is then predicted as the next occurring word. An alternative approach was also tested where average similarity of each class is also found and the class with highest average is then predicted as the next occurring word.

Method to find similarity between two wav files:

- Read the wav files using librosa which converts it into a numpy array.
- Calculate the moving average with a window size of 1500 to smoothen the signals.
- Cosine similarity between these two smoothened signals is calculated and returned.

The main constraint of this method is that the training audio and textual multilingual query dataset needed for the speech-to-text conversion model is very high(in terms of hundreds of thousands), which is not feasible with the team size and the time constraint. But this can be overcome, by expanding the dataset by generating recordings of different age groups,gender and language dialects.

This methodology also depends on the performance of the Google translation API and the Search Engine for the accuracy of the translation and output. The application depends on the storage constraints of the Database used as well as the limit on the amount of audio and textual queries that can be stored.

CHAPTER SEVEN

IMPLEMENTATION AND PSEUDO CODE

7.1 Preprocessing

The wav files were converted to a numpy array using librosa and the array was normalised between -1 to 1. The silence factor that existed in the beginning and end of the audio file was removed. The speed of audio files were changed by changing its frame rate.

class preProcess:

```
def __init__(self,fileName):
        self.fileName=fileName
def normalise(self):    #Normalising the array
        signal,sample_rate=librosa.load(self.fileName)
        signal.scaleArray(-1,1)
        return signal
def trimmed(self):
        signal,sample_rate=librosa.load(self.fileName)
        signal.removeSilence(beginning,end)
        return signal

def speedChange(self):
        signal,sample_rate=librosa.load(self.fileName)
        signal.changeFrameRate(targetLen=20000)
        return signal
```

7.2 Splitting of sentence

The moving average for the signal is calculated and minimas at windows of each 10000 were found out which would be the points at which the sentence will be splitted to chunks where each represents a single word utterance.

def split(fileName)

Window_size = 10000

Signal,sample_rate = librosa.read("path to wav file")

X = Moving_average(winodw_size)

result = split_at_minimas(X)

return result

7.3 Word Predictor

Sentences were tokenized and all n-gram (n=4) sequences were generated.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
model = Sequential()
model.add(Embedding(vocabulary_size, seq_len, input_length=seq_len))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
model.add(Dense(50,activation='relu'))
model.add(Dense(vocabulary_size, activation='softmax'))
print(model.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 3, 3)              201
_________________________________________________________________
lstm_4 (LSTM)                (None, 3, 50)             10800
_________________________________________________________________
lstm_5 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense_4 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_5 (Dense)              (None, 67)                3417
=================================================================
Total params: 37,168
Trainable params: 37,168
Non-trainable params: 0
_________________________________________________________________
None
```

Fig 23 Model summary for word prediction using LSTM

def wordPredictor(sentence):

clean = sentence.lower()

tokenize = clean.word_tokenize()

vocab_size = len(tokenizer[train_len])

split() = Tokenizer().fit_text_to_sequences(tokenize)

7.4 Speech to text

7.4.1 Methodology 1: Deep learning

Extracted MFCC features were used to train the neural model and the model and it is a classification model which classifies the input chunk into one of the next possible words returned by the word predictor model.

```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_24 (Dense)             (None, 100)               4100
_________________________________________________________________
activation_24 (Activation)   (None, 100)               0
_________________________________________________________________
dropout_18 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_25 (Dense)             (None, 200)               20200
_________________________________________________________________
activation_25 (Activation)   (None, 200)               0
_________________________________________________________________
dropout_19 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_26 (Dense)             (None, 100)               20100
_________________________________________________________________
activation_26 (Activation)   (None, 100)               0
_________________________________________________________________
dropout_20 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_27 (Dense)             (None, 5)                 505
_________________________________________________________________
activation_27 (Activation)   (None, 5)                 0
=================================================================
```

Fig 24 speech to text deep learning model summary

def speech_to_text(audio_query):

```
chunks = []
chunks = split(audio_query)
chunks.preprocess()
cur_sentence = "<s>"
for chunk in chunks:
        classes = word_predictor(cur_sentence)
        next_word = model.predict(chunk,classes)
        cur_sentence = cur_sentence + next_word
return cur_sentence
```

7.4.2 Methodology 2 : Based on similarity

Cosine similarity is found between the input chunk and among all the recordings of next possible words and highest occurring class among the top 20 most similar recordings is then predicted as the next occurring word.

def find_similarity(file1,file2):

```
audio1 = librosa.read(file1)
audio2 = librosa.read(file2)
audio1 = moving_average(audio1, window = 1500)
audio2 =  moving_average(audio2, window = 1500)
return cosine_similarity(audio,audio2)
```

def predict(chunk,classes):

```
similarity=[]
for i in trainingData:
```

```
            if i in classes:
                    similarity.append(find_similarity(i,chunk))
    similarity.sort(reverse=True)
    similarity=similarity[:20]
    predictedWord=mostFrequentClass(similarity)
    return predictedWord
```

CHAPTER EIGHT

RESULTS AND DISCUSSION

8.1 Splitting of sentence

This is the visualization of a wav file for the recording of the sentence “how to hang painting on gode”.

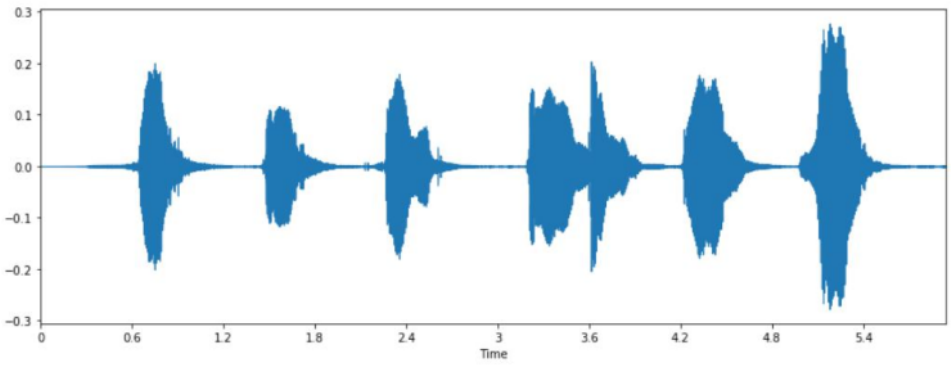

Fig 25 signal for the recording of a sample sentence

After smoothing, the minimas are defined clearly and hence split at these points :

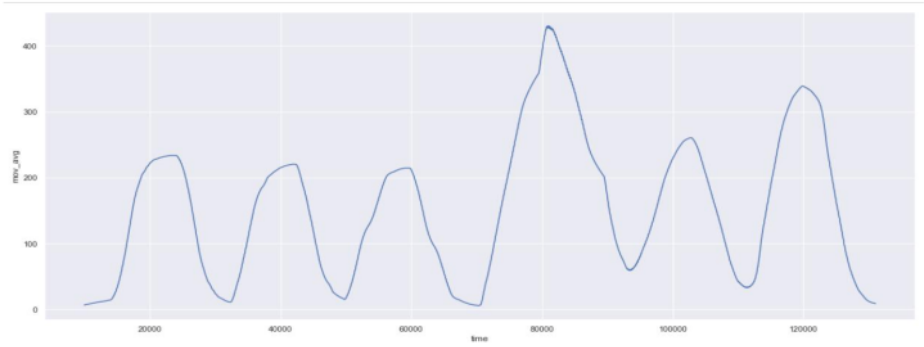

Fig 26 smoothened signal

	Splitting Module	
Tested Sentence	**Actual Number of Words**	**Predicted Number of Words**
nanna next turn yenu	4	4
what is my next saradi	5	5
what is my mundina saradi	5	5
how is the datthane ahead	5	5
munde traffic hegidhe	3	3
how is the datthane munde	5	5
what are the nearby anila stations	6	6
what are the nearby gas kendragalu	6	6
what are the nearby anila kendragalu	6	6
hathira gas stations yavdu	4	4
hathira gas kendragalu yavdu	4	4
hathira anila stations yavdu	4	4
are there any kredaa mydana near me	7	7
nanna hathira yavude sports ground idhya	6	6
are there any sports mydana near	7	7

me		
are there any kredaa grounds near me	7	7
what are some good nearby stalagalu to visit	8	8
what are some uttama hathiradha places to visit	8	8
what are some uttama hathiradha stalagalu to visit	8	9
bheti needalu hathiradha good places yavuvu	6	6
hathiradha park yaavaaga thereyuthadhe	4	5
When does the nearest udhyaana open	6	6
hathiradha udhyaanavu yaavaaga open aagothe	5	6
Nanna hathira iro plumbing service	5	5
Hathira iro kolaayi service	4	4
Hathira iro plumbing service	4	4
hathiradha job openings	3	3
Udhyogavakaasha near me	3	3
Kelasa openings near me	4	4

Table 1: Test results of splitting module

The splitting algorithm was tested on 30 sentences out of which 28 were correctly splitted into respective words as shown in the table. The incorrectly splitted sentences were re-recorded with sufficient gaps between words after which the splitting was done accurately.

8.2 Word Prediction:

```
model.fit(train_inputs,train_targets,epochs=500,verbose=1)
model.save("modelOne.h5")
Epoch 491/500
4/4 [==============================] - 0s 4ms/step - loss: 0.2042 - accuracy: 0.8899
Epoch 492/500
4/4 [==============================] - 0s 4ms/step - loss: 0.2360 - accuracy: 0.8300
Epoch 493/500
4/4 [==============================] - 0s 4ms/step - loss: 0.2174 - accuracy: 0.8719
Epoch 494/500
4/4 [==============================] - 0s 5ms/step - loss: 0.2401 - accuracy: 0.8510
Epoch 495/500
4/4 [==============================] - 0s 5ms/step - loss: 0.2281 - accuracy: 0.8731
Epoch 496/500
4/4 [==============================] - 0s 5ms/step - loss: 0.1995 - accuracy: 0.8876
Epoch 497/500
4/4 [==============================] - 0s 5ms/step - loss: 0.2151 - accuracy: 0.8846
Epoch 498/500
4/4 [==============================] - 0s 4ms/step - loss: 0.2099 - accuracy: 0.8877
Epoch 499/500
4/4 [==============================] - 0s 4ms/step - loss: 0.2298 - accuracy: 0.8676
Epoch 500/500
4/4 [==============================] - 0s 4ms/step - loss: 0.2033 - accuracy: 0.9055
```

Fig 27 performance of word prediction model

Word Predictor model gave 90% accuracy and when predicted top 5 possible next words for a current sentence, the desired word was present in this predicted possible words.

8.3 Speech to text:

8.3.1 Methodology 1: Deep learning

DL Module		
Actual Sentence	**Predicted Sentence**	**Average Accuracy**
nanna next turn yenu	nanna next turn yenu	0.83
what is my next saradi	what is my next saradi	0.71
what is my mundina saradi	what aagothe any stalagalu saradi	0.68
how is the datthane ahead	how service the yaavaaga ahead	0.73
munde traffic hegidhe	munde traffic hegidhe	0.78

how is the datthane munde	how service the yaavaaga ahead	0.79
what are the nearby anila stations	what does the nearby anila stations	0.84
what are the nearby gas kendragalu	what aagothe the nearby gas kendragalu	0.84
what are the nearby anila kendragalu	what aagothe the nearby good kendragalu	0.85
hathira gas stations yavdu	what gas stations visit	0.88
hathira gas kendragalu yavdu	hathira gas kendragalu open	0.85
hathira anila stations yavdu	hathira gas stations service	0.85
are there any kredaa mydana near me	hathiradha there any sports plumbing near me	0.89
nanna hathira yavude sports ground idhya	hathira next yavude sports ground idhya	0.87
are there any sports mydana near me	are there any kredaa mydana near me	0.88
are there any kredaa grounds near me	what there any kredaa grounds to me	0.87
what are some good nearby stalagalu to visit	what are the good nearby stalagalu to visit	0.93
what are some uttama hathiradha places to visit	what are hathiradha uttama hathiradha places to visit	0.9
what are some uttama hathiradha stalagalu to visit	what are the uttama hathiradha stalagalu to visit	0.9
bheti needalu hathiradha good places yavuvu	bheti openings hathiradha good places yavuvu	0.89
hathiradha park yaavaaga thereyuthadhe	hathira park yaavaaga thereyuthadhe	0.95
when does the nearest udhyaana open	when there the nearest open openings	0.84
hathiradha udhyaanavu yaavaaga open aagothe	are udhyaanavu yaavaaga open aagothe	0.86
nanna hathira iro plumbing service	hathira traffic iro ground service	0.85
hathira iro kolaayi service	nanna iro kolaayi service	0.87
hathira iro plumbing service	are iro plumbing service	0.84

hathiradha job openings	hathira park openings	0.87
udhyogavakaasha near me	udhyogavakaasha needalu me	0.94
kelasa openings near me	what openings me me	0.86
	Model Accuracy	0.85
	Prediction Accuracy	0.71

Table 2: Test results of speech to text conversion using deep learning

The average accuracy for each sentence was calculated by taking the accuracy of models while predicting each individual word of that corresponding sentence. Model Accuracy is the average accuracy of all the sentences tested. Prediction Accuracy is the number of words correctly predicted divided by the total number of words present.

8.3.2 Methodology 2 : Based on similarity

Speech To Text Module based on similarity				
Actual Sentence	**Predicted Sentence(using average similarity of each class)**	**Accuracy**	**Predicted Sentence(using maximum count among top 20 most similar recordings)**	**Accuracy**
nanna next turn yenu	nanna next turn yenu	1	nanna next turn yenu	1
what is my next saradi	what is any next yenu	0.6	how is my next saradi	0.8
what is my mundina saradi	are is any mundina yenu	0.4	what is my mundina yenu	0.8
how is the datthane ahead	what is some yaavaaga ahead	0.4	what aagothe my nearest ahead	0.2
munde traffic hegidhe	are traffic hegidhe	0.67	are traffic hegidhe	0.67
how is the datthane munde	are is my nearest munde	0.4	what is some nearest munde	0.4
what are the nearby	what is the nearby	0.67	how is some nearby	0.17

anila stations	gas stations		gas kendragalu	
what are the nearby gas kendragalu	what is some nearby gas kendragalu	0.67	what is some nearby gas kendragalu	0.67
what are the nearby anila kendragalu	what is the nearby gas kendragalu	0.67	what is the nearby gas kendragalu	0.67
hathira gas stations yavdu	what iro stations yavdu	0.5	what gas stations yavdu	0.75
hathira gas kendragalu yavdu	hathiradha iro stations yavdu	0.25	hathira gas kendragalu yavdu	1
hathira anila stations yavdu	what gas plumbing yavdu	0.25	what gas kendragalu yavdu	0.25
are there any kredaa mydana near me	are openings yaavaaga kredaa grounds to me	0.43	what does any kredaa grounds to me	0.43
nanna hathira yavude sports ground idhya	nanna next yavude plumbing ground idhya	0.67	nanna next yavude ground ground idhya	0.67
are there any sports mydana near me	are there any mundina plumbing near near	0.57	are there any kredaa mydana near me	0.86
are there any kredaa grounds near me	what there any mundina grounds near near	0.57	how there any mundina grounds near me	0.71
what are some good nearby stalagalu to visit	are are some good nearby mundina to visit	0.75	how are some good nearby places to visit	0.75
what are some uttama hathiradha places to visit	are are the uttama hathiradha places to visit	0.75	are are some good hathiradha places to visit	0.75
what are some uttama hathiradha stalagalu to visit	are are hathiradha uttama kolaayi stalagalu to visit	0.62	are are hathiradha good kolaayi nearby to visit	0.38

bheti needalu hathiradha good places yavuvu	nanna near hathiradha good turn yavuvu	0.5	hathiradha near hathiradha good places yavuvu	0.67
hathiradha park yaavaaga thereyuthadhe	what park anila thereyuthadhe	0.5	hathiradha park any udhyaana	0.5
when does the nearest udhyaana open	when are the iro udhyaana open	0.67	when are the iro udhyaana open	0.67
hathiradha udhyaanavu yaavaaga open aagothe	are udhyaanavu yaavaaga grounds aagothe	0.6	are udhyaanavu yaavaaga grounds aagothe	0.6
nanna hathira iro plumbing service	nanna hathira iro plumbing aagothe	0.8	nanna hathira iro plumbing aagothe	0.8
hathira iro kolaayi service	hathira iro kolaayi yavdu	0.75	hathira iro kolaayi yavdu	0.75
hathira iro plumbing service	hathira iro plumbing service	1	hathira iro plumbing service	1
hathiradha job openings	are udhyaanavu openings	0.33	are job openings	0.67
udhyogavakaasha near me	are near near	0.33	are near me	0.67
kelasa openings near me	nanna openings near near	0.5	nanna openings near me	0.75

Table 3: Test results of speech to text conversion based on similarity

Average similarity of each class is also found and the class with highest average is then predicted as the next occurring word. The second approach was to find similarity between the input chunk and among all the recordings of next possible words. Highest occurring class among the top 20 most similar recordings is then predicted as the next occurring word. The accuracy for both the methods is as shown in the table where it was calculated by taking the ratio of number of correctly predicted words by total number of words in the sentence.

The average accuracy of 0.59 was achieved when prediction was done using average similarity of each class and 0.64 was achieved when using the highest occurring class among the top 20 most similar recordings.

CHAPTER NINE

CONCLUSION

The methodology used by the team is a completely novel approach which uses the self-learning abilities of a model to recognize and translate the multilingual queries to a monolingual query, as accurately as possible. The Deep Learning model uses the top predictions from the Word Predictor model to reduce the search space while identifying and translating each word input from the audio query. Our model, when tested on the generated multilingual dataset, gives an accuracy of 85%. And when it is tested live by the user, it gives an accuracy of 71%.
We can compare this result to the existing multilingual translation methods when tested on the generated dataset as follows :

- CMU Sphinx : Accuracy = 57%
- Similarity measure using Neural Network : Accuracy is 64%
- Google Translate : 35.4% while recognizing Kannada words

We can see our novelty method is able to perform to a good standard, but there are a few improvements that can be added which could further increase the accuracy, these are:

- Increasing the data set, both audio and textual, with a variation in voice such as age, gender, noise level, etc.
- The similarity approach can be improved using similarity index comparison of the spectrograms of the words, instead of the previously mentioned method which compares the wav forms.
- The prediction of words and their parts of speech in a particular sentence by leveraging different and more useful language features.

Although these improvements are beyond the scope of our project, research and implementation of these features can cause a significant boost to the accuracy of multilingual translation.

REFERENCE AND BIBLIOGRAPHY

REFERENCES

- Flutter documentation
- Firebase documentation
- Project Plan
- Cloud API documentation
- Survey of existing models
- Project Requirement Specifications
- Project Plan
- High Level Document
- Low Level Document
- Final Results
- http://www.speech.cs.cmu.edu/tools/lmtool-new.html

BIBLIOGRAPHY

- Multilingual Speech Recognition with a single end-to-end model - Shubham Toshniwal, 15th February 2018
- Multilingual Text-to-Speech Software Component for Dynamic Language Identification

and Voice Switching ,September 2016 Paul Fogarassy,Costin Pribeanu

- AUTOMATIC LANGUAGE IDENTIFICATION USING DEEP NEURAL NETWORKS,2016, Ignacio Lopez-Moreno , Javier Gonzalez-Dominguez, Oldrich Plchot

Speech recognition

ORIGINALITY REPORT

5%	4%	4%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Paul FOGARASSY-NESZLY, Costin PRIBEANU. "Multilingual Text-to-Speech Software Component for Dynamic Language Identification and Voice Switching", Studies in Informatics and Control, 2016 Publication	1%
2	hdl.handle.net Internet Source	1%
3	sigport.org Internet Source	<1%
4	innotech-vn.com Internet Source	<1%
5	www.scilit.net Internet Source	<1%
6	Shubham Toshniwal, Tara N. Sainath, Ron J. Weiss, Bo Li, Pedro Moreno, Eugene Weinstein, Kanishka Rao. "Multilingual Speech Recognition with a Single End-to-End Model", 2018 IEEE International Conference on	<1%

Acoustics, Speech and Signal Processing (ICASSP), 2018
Publication

7	in-time-project.eu Internet Source	<1 %
8	Shubham Toshniwal, Anjuli Kannan, Chung-Cheng Chiu, Yonghui Wu, Tara N Sainath, Karen Livescu. "A Comparison of Techniques for Language Model Integration in Encoder-Decoder Speech Recognition", 2018 IEEE Spoken Language Technology Workshop (SLT), 2018 Publication	<1 %
9	repositorio.uam.es Internet Source	<1 %
10	Ratanon Jantanasukon, Arit Thammano. "Adaptive Learning Rate for Dealing with Imbalanced Data in Classification Problems", 2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering, 2021 Publication	<1 %
11	export.arxiv.org Internet Source	<1 %
12	cmusphinx.sourceforge.net Internet Source	

		<1%
13	sic.ici.ro Internet Source	<1%
14	www.slideshare.net Internet Source	<1%
15	docplayer.net Internet Source	<1%
16	research.aalto.fi Internet Source	<1%
17	sourceforge.net Internet Source	<1%
18	oranguide.com Internet Source	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		