Automatic Subtitle Encoder Using Machine Translation

A PROJECT REPORT

Submitted by

PREMKUMAR A (715517104044)

MUKESHWAR S (715517104041)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH, COIMBATORE - 641062

ANNA UNIVERSITY: CHENNAI - 600 025

APRIL 2021

ANNA UNIVERSITY: CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"Automatic Subtitle Encoder Using Machine Translation"** is the bonafide work of "**PREMKUMAR A (715517104044) and MUKESHWAR S (715517104041)"** who carried out the project work under my supervision.

SIGNATURE

Dr. R. MANIMEGALAI

HEAD OF THE DEPARTMENT

PROFESSOR

Department of Computer Science
and Engineering

PSG Institute of Technology and
Applied Research

Neelambur, Coimbatore-641 062

SIGNATURE

 Dr. S. BHUVANA

SUPERVISOR

ASSOCIATE PROFESSOR

Department of Computer Science
and Engineering

PSG Institute of Technology and
Applied Research

Neelambur, Coimbatore-641 062

Submitted for the project viva-voce Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost, we express our deep sense of gratefulness to our respected Managing Trustee, **Shri. L. GOPALAKRISHNAN** for his provision to utilize all the necessary facilities in the institution.

We express our heartfelt gratitude to our beloved Principal of our institution, **Dr. G. CHANDRAMOHAN B.E(Hons), M.Tech, Ph.D**, for his overwhelming support and encouragement on this project. We would also like to extend our heartfelt thanks to our honorable Secretary **Dr. P. V. MOHANRAM B.E(Hons), M.Tech, Ph.D**, for his moral support.

We are greatly indebted to Head of the Department Computer Science and Engineering, **Dr. R. MANIMEGALAI M.E., Ph.D**, for her guidance and continuous support which was instrumental in the completion of this project.

We extend our thanks to our guide, **Dr. S. BHUVANA M.E., Ph.D** for her guidance and constant supervision without which we could not have completed this project study.

We would further like to thank our project coordinator **Mr. S. THIVAHARAN M.Tech,** for carrying out reviews smoothly and the valuable feedback provided at each step of the project development.

Finally, I take this opportunity to extend my humble gratitude to Almighty without the blessings of whom we would not have been able to overcome the challenges posed by this project study.

PREMKUMAR A
MUKESHWAR S

PLAGIARISM REPORT (VERIGUIDE)

ABSTRACT

In the present scenario, video plays a crucial role to help people understand and comprehend the information. It becomes important to make videos available to the people having auditory problems and even more for the people to remove gaps of their native language. According to the Eighth Schedule of Indian Constitution, India consists of around 22 languages. To remove the above problems, we can make use of subtitles for the respective video. Traditionally, people use to download and load subtitles for any other language videos/movies from the internet. Downloading subtitles from the internet is a monotonous process and at times, people find it difficult to search for subtitles in every language. Automatic Subtitle Generation is an undergoing subject of research. Hence, this project undergoes three distinct modules namely Audio Extraction which converts input video file of any MPEG standard format to audio file of .flac format with the help of Psycho-Acoustic Model. Then Speech Recognition of the extracted audio file and finally, Subtitle Generation in which Sub Ripped Text File is generated which can be synchronized with the respective video file. For now, project is able to generate subtitles for 66 languages with high accuracy. In future, we tend to generate .srt files for other native languages.

Keywords : SubRip Subtitle, Psycho-Acoustic, Machine Translation, Cloud API

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	DESCRIPTION
API	Application Programming Interface
ASR	Automatic Speech Recognition
DDL	Dynamic Link Library
GUI	Graphical User Interface
HMM	Hidden Markov Model
ICACCS	International Conference on Advanced Computing and Communications System
ICAML	International Conference on Applied ML
LSTM	Long Short Term Memory
MPEG	Moving Pictures Expert Group
PCM	Pulse Code Modulation
REST	Representational State Transfer
RNN	Recurrent Neural Networks
SRT	SubRip Subtitle File
TTS	Text to Speech
URL	Uniform Resource Locator
WER	Word Error Rate
XML	Extensible Markup Language
IDE	Integrated Development Environment

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In this Information Age Era, Language has become the information carrier and most significant for humans to communicate and it is also the barrier of communications between people. In the present scenario, video plays a crucial role to help people understand and comprehend the information. It becomes crucial and mandatory to make videos available to the people having auditory problems and also for the people to remove gaps of their native language. According to the Eighth Schedule of Indian Constitution, India consists of around 22 languages. To remove all above problems, the subtitles can be embedded for the respective video. Traditionally, people use to download and load subtitles for any other language videos/movies from the internet. Downloading subtitles from the internet is a monotonous process and at times, people find it difficult to search for subtitles in every language. Subtitles provide information for individuals who have difficulty understanding the speech and auditory components for the visual. This explains a valid area of research.

At present, Users download the sub ripped subtitle file from the internet to synchronize with the respective video. The file format should be .srt, it does not take other formats as it will not synchronize with the video file. The sub ripped subtitle file comprises time stamps and texts that are spoken by the video component. There are no services or software that provide necessary sub ripped subtitle files automatically in user needed language. It will be available only if anyone intends to generate a .srt file manually. This project undergoes three distinct modules namely Audio Extraction which converts input video file of any MPEG standard format to audio file of .flac format with the help of Psycho-Acoustic Model. Then Speech Recognition of the extracted audio file and finally, Subtitle Generation in which Sub Ripped Text File is generated which can be synchronized with the respective video file. Several methods are developed using Long Short Term Memory (LSTM) – a special kind of recurrent neural networks. Implementation of these networks resulted in a need for more processing power and more time.

1.1 Objective

To help people having auditory problems and to make people understand and comprehend the information. Subtitles play a major role in solving these problems. Subtitle creation by humans manually is tedious. Automatic Subtitle Encoder will help to generate subtitles in user requested language and can embed with the video file. The idea of Multithreading is used to provide a high level of synchronization without getting any subtitle delay.

1.2 Scope and Motivation

Without Subtitle Encoder, people find it tedious to search for necessary language subtitles from the internet. Upon implementation of Subtitle Encoder, people can able to generate their target language subtitle whenever they need. Subtitle Encoder will cover all major languages that are spoken by people all over the world. Easy user interface will help people to choose whatever language the user prefers.

1.3 Proposed Methods

The core function of the subtitle encoder is Machine Translation. It can be simplified into three methods: the analysis of source language text, conversion of source language text to target language text and generation of target language. The main issue in Machine translation is the accuracy of respective language translation. In fact, the methodology adopted by the machine translation system will affect the accuracy. Machine Translation can be done in three methods: Human Translation with machine support, Machine translation with human support and fully automated translation.

1.3.1 Different approaches

There are three popular approaches for performing machine translation: (a) Rule Based Machine Translation, (b) Empirical Machine Translation and (c) Hybrid Based Machine Translation as shown in Fig. 1.1
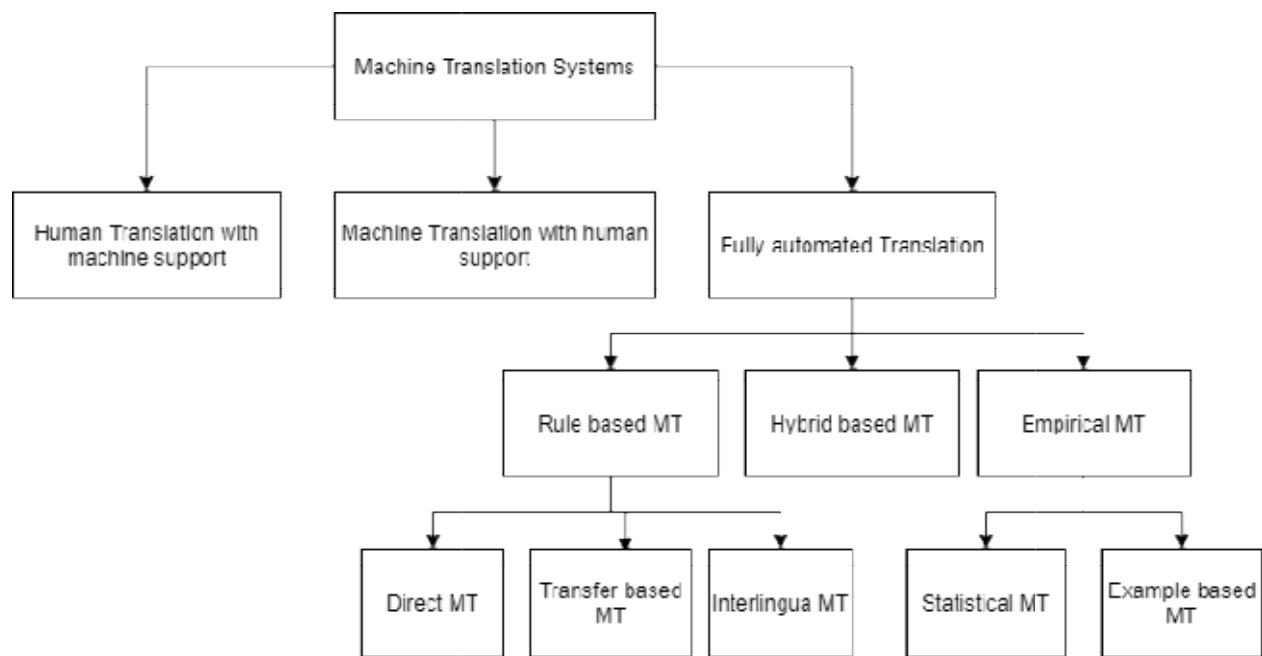
Figure 1.1 Machine Translation Approaches

A. Rule Based Machine Translation

Rule based or Literal Translation or Word Based Translation or Dictionary Based Translation is that the words will be translated as a dictionary does word by word, usually without much correlation of meaning between them. This method is applicable only for a selective pair of languages and is not versatile. This method is further divided into three classifications: Direct Machine Translation, Transfer based Machine Translation and Interlingua Machine Translation. In Direct MT, the source sentence is mapped with corresponding words of target language. In Transfer MT, analysis of a source sentence is done at first and corresponding target sentences are produced. Transfer Machine Translation uses both bilingual dictionaries and grammatical rules to perform conversion. In Interlingua MT, first the source language is analyzed and converted to an Interlingua language, the source language is converted to target language.

B. Empirical Machine Translation

This approach treats machine translation as a process of information transmission, using a channel model to interpret machine translation. The main idea is about translation of an entire sentence into equivalent suitable sentences of the required language. This method is further divided into two classifications: Statistical Machine Translation and Example Based Machine Translation. In Statistical MT, Statistic-primarily based device translation may be divided into the subsequent aspects: version issues, schooling issues and interpreting trouble. The version trouble is to set up a chance version for device translation, that is, to outline the Calculation approach of the chance of sentence translation from the supply language to the goal language. Training trouble is to utilize the corpus to get all of the parameters of this version. The interpreting trouble is to discover the most chance translation for any sentence of supply language on the premise of the acknowledged fashions and parameters. In Example MT, The bilingual case library is the primary source of information for case-based machine translation. The main challenge is to optimize statistics and build a bilingual case library. Case-based machine translation has a major impact for the same or similar text translation, and its function has grown in importance as the size of the sentences library has grown.

C. Hybrid Based Machine Translation

Hybrid Machine Translation that characterizes the use of multiple machine translations to produce respective language sentences. It achieves a satisfactory level of accuracy with the use of multiple machines. It is a combination of statistical and neural to obtain more accuracy. When it comes to hybrid machine translation, the multi-engine translation method involves running several machine translation systems in parallel. The cumulative output of all the subsystems involved in the process is normally used to produce the final output. Multi engine translation approaches are typically used in rule-based and statistical frameworks. With the help of many machine translators, the output efficiency can be achieved greater. This can also leads to decrease in efficiency since failure or poor translation in one translator might affect the

systems entire performance. One has to think properly and choose appropriate translators to build Hybrid based machine translation model. Coming up with complexity and space restrictions, this model might not good suggestion as the complexity gets increases on addition of every new translator into the Hybrid based machine translation model. Thus, if efficiency and accuracy is more important than space and complexity, users can prefer to choose HMT models.

1.4 Contribution of the Project

Automatic Subtitle Encoder will help people who are suffering from auditory problems and help people to understand the concepts crisp and clear. Subtitles help people to solve the gaps between native speakers. Translation by people can't meet the demand of society. Hence, the use of subtitle encoder will generate the needed subtitle whenever the user intends to understand the video concepts. Machine Translation also helps people when a user has more amounts of user generated content that needs to be translated quickly. Some examples of user generated content include, online comments, customer review on certain products and social media posts. With the advent of foreign trade and business among many countries, translators play an important role to communicate among many nations. Encoders are on-demand machines in the upcoming days as these machines are light weighted, minimalistic in design and has user friendly interface. Anyone with no prior knowledge can also use this encoder because of its simplicity.

CHAPTER 2
LITERATURE SURVEY

Machine Translation is the way of converting from source language to target language. It is one of the most active research areas in natural language processing where industries are driving to get useful sights from customer reviews so that they can market their product accordingly. Subtitle Generation is one of the dominant applications of Machine Translation.

2.1 Existing models

Mathur A et al.,[1] proposed a model framework on "Generating Subtitles Automatically Using Audio Extraction and Speech Recognition" and published a paper at IEEE International Conference on Computational Intelligence & Communication Technology. The model does conversion from video file of any MPEG format to audio format of MP3 as first phase followed by generating subtitles in English language. This model also provides synchronization of subtitles along with the video file. This model can convert into various global languages such as English, French, and German. It has achieved an accuracy of about 75%.

Lero R D et al., [11] proposed a model on "Communications using a speech-to-text-to-speech pipeline" and conducted International Conference on Wireless and Mobile Computing, Networking and Communications. This model explains the feasibility of an automated Speech to speech to encode the voice message than encoding the regular voice codecs. To analyze the

Advantages of the above speech to text and text to speech pipeline transcript and compare it against standard Pulse Code Modulation (PCM), the error rate of user transcribed sentences based on Semantically Unpredictable Sentences test. The use of a speech to speech pipeline has been used multiple times to provide translation service and provides assistance in Software as a Service Platform. Pipeline consists of the speech recognition phase and speech synthesis phase. The entire process is modularization as the components can be reused for future use. The concept of modularization also helps

to couple and decouple components based on user need. Due to complexity involved in speech conversion and to achieve time constraint, Users intend to use third party cloud service platforms such as Amazon Web Services, Google Cloud Platform and Microsoft Azure helps to execute speech translation in much easier and time efficient manner.

Idea is to convert the text stream messages into speech samples using a method called Text to Speech API (TTS). The average size of all samples in the set and various encoding standards which illustrates the amount of bandwidth consumed when compressed using TTS method as shown in Table 2.1

Table 2.1 Bit rate and Average size comparison for given sample codec

CODEC	SIZE OF SAMPLE	BIT RATE
SPEEX	2495	7912
PCM	20232	64136
OPUS	3390	10960
STS	32	105

Converted sentences are then measured for their Word Error Rate. It explains how much of the ground truth sentence had to be changed to obtain the hypothesis sentence.

Satpathy et al., [12] proposed a model on "Analysis of Learning Approaches for Machine Translation Systems" in the International Conference on Applied Machine Learning (ICAML). It explains an analytical study on machine translation with lexical and structural ambiguity. It resolves the gap between programmers or developers and linguists can be resolved with this system. While developing these systems, there exists few lexical ambiguity, linguistic ambiguity and structural and synonymic ambiguity. To develop a proper bilingual system one need to analyze the sentence and also the technology behind the machine which translates the source to target language.

Systran is one of the examples for a literal machine translation system. Initially, it converts only language from Russian to English. The simple architecture of literal or direct Machine Translation is shown in Fig 2.1
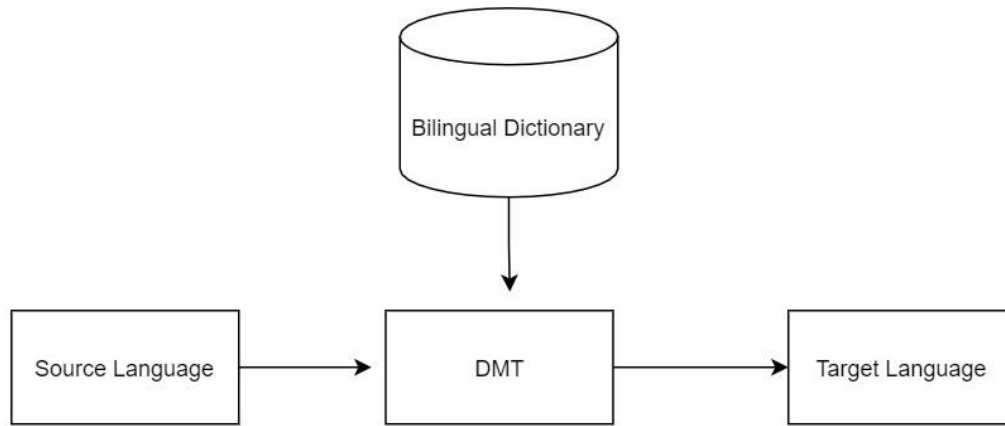
Figure 2.1 Block Diagram for Direct Machine Translation

Ramani A, et al .,[2] proposed a model on "Automatic Subtitle Generation for Videos" and conducted an International conference on Advanced Computing and Communication Systems (ICACCS). The main objective of the project is to choose a suitable Speech Recognition Engine such that while designing a video player to synchronize the subtitle, it will adopt properly. The proposed model comprises three active modules: (a) Audio Extractor, (b) Speech Recognition Engine and (c) Subtitle Synchronization. The chosen data set to test this model is Mozilla Common Voice Data Set as it has a good intermix of different speakers from various countries around the world. The data set comprises video lectures of around 1,000 hours and having a test and train split of 3:7 ratio.

Rizwan Sheikh et al ., [5] extracted audio from the video and converted the video into wav file format. His model has performed a gold size reduction of around 24%. The audio processing is done using chunks and feeds the output into the CMU Sphinx engine for speech recognition. It takes three inputs: psycho acoustic model, language model and dictionary.

K. M. Chaman Kumar et al, [12] "A Survey of Machine Translation Approaches" focus on producing subtitles for hearing impaired people. The flow of processes consist of Object identification, Image Annotation, Video Annotation, Automatic Speech Recognition (ASR) and parallel processing on the audio file. There are various forms of Speech recognition Engines available. Some of the themes include Deep Speech and PocketSphinx. Deep Speech works under recurrent neural networks (RNN). It is an open

source Speech to Text engine which has been implemented using Baidu's Deep Speech research paper. The Recurrent Neural Network works bidirectional with recurrent layers of two groups of hidden layers where one layer is for forward recurrence and another for backward recurrence. Pocket Sphinx uses Hidden Markov Model (HMM) works based on the given audio input. Hidden Markov Model follows the use of transition probabilities to compute an overall probability to reach a particular given state model. In the case of Sphinx, the inputs calculated against the transition probability matrix are evaluated. This evaluated result is called "Phones".

There are three independent models that are used to solve speech recognition tasks which are in tandem.

- Psycho Acoustic model that contains properties of each senome. A detector object to check for the presence of a phone in utterance is called a senome.

- A dictionary that contains a mapping feature between words and sequence of phones.

- A language model for restricting search context whenever a partial sentence is identified which helps in increasing the speed and computation.

The average word error rate for PocketSphinx is 20% and for Deep Speech is about 15.75%. Thus, Sphinx performs better than Deep Speech Translation Engine.

The word Rate metric has been used to help us quantitatively compare various approaches. This approach is similar to the Machine Translation method of error rate calculation or loss function calculation. Word Error Rate (WER) can be calculated as,

$$\text{Word Rate Error} = \frac{I+D+S}{N} \qquad \longrightarrow (1)$$

Where I represents Number of words in a sentence

    D represents Number of words deleted in a sentence

    S represents Number of words substituted in a sentence

    N represents Total number of words in a sentence

Deep Speech requires a large amount of data set and consumes more memory even when it is idle with only the model loaded. Deep Speech offers a tool to map with the memory

and model and it may paint a better picture when compared to other models. A product like this can target regions as it consumes more bandwidth and availability of internet is tandem in some regions.

Kehai Chen et al .,[11] published a paper on "Towards more diverse input representation for Neural Machine Translation" which states how source information plays an important role in transformation based translation system.Word Embedding and positional Embedding are added as input representation. Transformation based translation systems rely mainly on self Attention Networks (SAN).

Ozan, [15] published a paper on "Increasing system performance in machine learning by using multiprocessing," in 26th Signal Processing and Communications Applications Conference where he states that system's performance can be abruptly increased when we have multiple CPU cores working parallely on different instructions in particular given clock cycle.

H. Arif et al., [6] and his colleagues published an paper on "A Comparison between Google Cloud Service and iCloud" in IEEE 4th International Conference on Computer and Communication Systems (ICCCS) stating that there will evolution in cloud technologies. People will get confuse to choose which cloud service provider. Absolutely, Apple product users has much more benefit when they use iCloud. They provide enough synchronization methodologies. But, for android users, they can prefer Google cloud since google has ownership with android. Each users has their own benefits with their own platform service providers.

2.2 Drawbacks on Existing Models

The models explained above have conversion only for their own native languages. The accuracy obtained from the above models is lesser than 80%. For implementation with the help of neural networks, it takes more time than the respective original video length. This will make the users feel uncomfortable in using these models. It will eventually spend time in searching for sub ripped subtitle file.

CHAPTER 3
SYSTEM DESCRIPTION

3.1 Software and Hardware Requirements

Software Requirements

- Operating System : Windows 7 and higher versions,

  Any Linux distributions.

- Language : Python 3.0
- Packages Used : six, subprocess, xlwt, xlrd, ffmpeg, time,

  xlswriter, ntpath, tkinter, shlex, pysrt,PIL,

  google_OAuth

Hardware Requirements

- RAM : 6GB (recommended) and more
- Processor : Intel Core i5 and more
- Disk capacity : 500 KB (Minimum) and more
- Speed : 1GHZ and more

3.2 Packages Used

3.2.1 Six

Six provides simple and minimalistic utilities for wrapping over differences between Python 2.x pipeline and Python 3.x pipeline. It is intended to support various codebases that work on both Python 2.x pipeline and 3.x pipeline without any modification. Six python packages consist of only one python file and it is easy to import into any project whenever it is necessary. The name, "six", comes from normal mathematical multiplication that 2*3 equals 6. Since, "five" has already been snatched away, now all developers are using six tool.

3.2.2 Subprocess

Subprocess package intends to replace several modules in python 2 and 3 are os.system and os.spawn. This module allows users to spawn new processes and also

connect with input or output or error pipes and retain their codes. To invoke any subprocess use run() function for all use cases. This subprocess module also has another method .CompletedProcess which returns process id that has been finished already.

3.2.3 xlwt

xlwt library helps developers to write data and format data and information to older Excel files of format .xls The xlwt package has a Workbook class representing the workbook and all its content. Instantiate this workbook object to create a new workbook. To add a sheet in the object workbook instance, use the add_sheet method which takes two parameters as sheet name and cell_overwrite_ok. To save a file, use the .save method to save it as a workbook file.

3.2.4 ffmpeg

Python can be wrapped along with FFmpeg to work with complex graphs. This package is forked from github repository. Various filters have been included in the FFmpeg standard to produce accurate results.
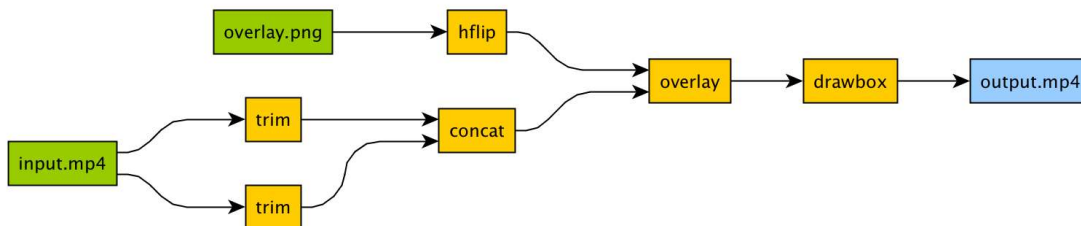

Figure 3.1 FFmpeg package processing input video

3.2.5 time

This module helps various time related functions. It comprises two modules such as datetime and calendar modules. Most of the functions are built in with C library functions. There are several functions available such as .gmtime(), strptime(), struct_time(), local_time(), mktime() etc.

3.2.6 xlsxwriter

This python module is used for writing into Excel files of version above 2007 and above. This uses file extension format as xlsx. It can be used to write numbers, formulae, text and hyperlinks to various worksheets/workbooks. It supports certain features like charts, formatted cells, auto fillers, merged cells, textboxes, support for adding macros, integration with pandas, data validation and drop down lists and provides 100% compatible Excel XLSX files.

3.2.7tkinter

This package is a standard python interface for Tk Graphical User Interface (GUI) toolkit. Both Tk and Tkinter are available for any operating systems. Tk is maintained at ActiveState which is not a part of python. In the Tk interface module, it includes a number of python GUI modules and it is usually a shared library or Dynamic Link Library(DLL). It comprises several modules like canvas, Tk.button, Tk.window, Tk.Text Box etc.., The pack() method is used to couple the entire components within the canvas or window object. This implementation should run indefinitely because it renders the screen based on the user progress.

3.2.8 ntpath

ntpath module helps with the path/location functionality of the respective file in any operating system. To read a file or to write a file, a user can use the open() function and for accessing the operating system's file system we can use os module. The high level path objects can be provided by using the pathlib module. Several other functions include exists(), lexists(), isfile(), ismount() etc… posixpath can be used for UNIX machines whereas ntpath can be used for Windows Machines.

3.2.9 shlex

The shlex package is used to write lexical analyzers. shlex stands for Simple Lexical Analysis. It can be used for simple syntaxes and useful for writing mini languages. It has certain functions like join(), quote() etc… The lexical analyzer object will be the shlex instance or shlex subclass instance.

3.2.10 PIL

PIL stands for Pillow which is an Python Imaging library for processing images to match up with the interpreter capabilities. It can be used for fast access to store data in users' required basic pixel format. It also serves as a general image processing tool. It has more reference mapping like Image, ImageChops, ImageColor, ImageDraw, ImagePalette, ImageMorph, ImageShow etc…

3.2.11 google_OAuth

OAuth is an authentication standard designed exclusively for access delegation, to authenticate the user to grant requests to websites or access services that are exposed through Application Programming Interface (API) without giving them passwords or any credentials. Google_OAuth is used to implement Open Authentication on Google products API and can be linked with developers' python code.

3.3 Google Cloud API

Cloud API allows developers to automate tasks such that developers can focus more on code logic rather than building or designing the software UI. Developers can communicate with the API using RESTful (REpresentational State Transfer) or XML (Xtensible Markup Language) to communicate with the backend core logic.

CHAPTER 4

SYSTEM DESIGN

4.1 Design Overview

The overall abstract view of the project can be seen from the flow diagram as shown in the figure 4.1.
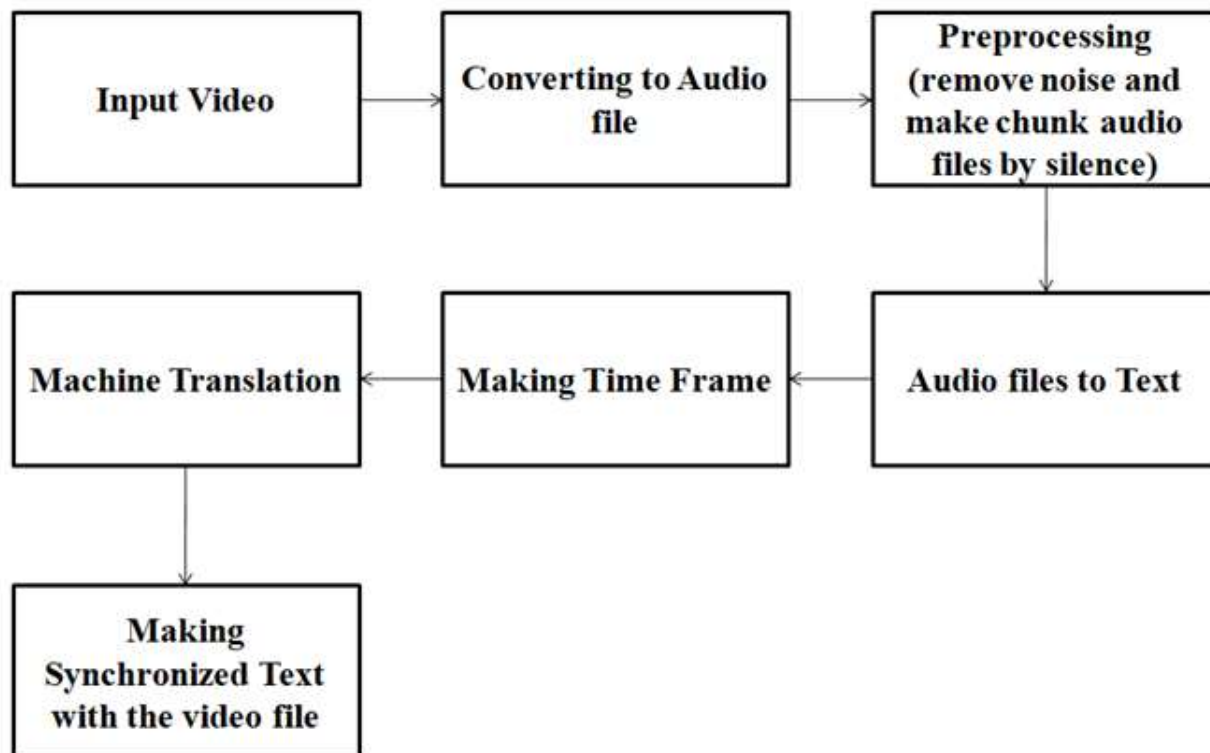

Figure 4.1 Block diagram for subtitle generator

The input of the video file is given by the user, where encoder has to generate a subtitle depends on the user requirements. The subtitle can be of any language.

The input file of the video file is converted into an audio file of any format using FFMPEG where each audio format gives some type of functionalities and also one should also consider the audio format because speech recognition gives accuracy based on the audio format in which it has been converted. In this project , primary goal is to

converting a video file to audio file of FLAC or LINEAR16 format which gives more accuracy during speech recognition.The audio file undergoes various stages during converting from video file to audio file where it has to be taken care of lossy compression, easy to recognize text based on silence and the noise should have been processed by the FFMPEG, this is why this format is suggested for speech recognition, It is not that other formats are not supported for speech recognition.

Now the next phase is to convert from audio file to text which is done using the advanced machine learning algorithm which has been hosted by Google and using the credentials provided by them. Encoder uses cloud speech-text API for speech recognition. Before recognition one have to undergo few steps, they are to establish a connection between our project and Google cloud speech-text API and need to get configured the request based on the user requirements some of the important parameters are type of the audio format, the language in which video file is given as input, whether the video should produce a time offset of start time and end time and audio channel count which represents the header of the audio format. It varies depending on each audio format, the header file should reach the audio format which has been specified.

After getting the response from the API encoder have to process the data because it contains the response of confidentiality, start time, end time, word ,metadata etc. remove unnecessary data and form a list of words with start time, end time, speaker tag, language in which it has been recognized. Now with this list of words encoder have to form a sentence which doesn't change meaning at any point only then the translation and the subtitle is generated with high accuracy. For that purpose the sentence formation is done based on if the word has a time gap of one sec, if the speaker tag is different and if the sentence exceeds certain time limit then break point should be fixed and appending into a new list. Finally the list contains the sentence with start time and end time which has been written into excel file for view purpose. Therefore at this phase the text is divided into timeframes.

The main phase of machine translation is done during this stage where the translation uses advanced neural networks. Now the text is imported from the file and read one by one which is given as input to the cloud speech translate API which converts the text from language to target language and the parameters need to be given are the credentials to establish connection and use the cloud speech translate API, and the target language in which it has to be generated. Before doing this we have to check the text is in the format of string binary or utf-8 because the translation supports utf-8 for this purpose after importing the input check whether it is a binary or not and convert to utf-8 and pass the processed text to the cloud translate API.

After the translation we have generate the .srt file called "SubRip subtitle", which contains the format of

- start time ---> end time
- segment number
- hours : minutes : seconds, milliseconds ----> hours : minutes : seconds, milliseconds
- new line space
- Divide the translated text into segments and write the file in subtitle format and save it as .srt file, now merge the file with the video file using FFMPEG and run the command in prompt using pop method which uses synchronous execution and ends the program after its execution. Therefore the automatic subtitle generator using cloud translate API is done.

4.2 USE CASE DIAGRAM:

The use case diagram represents the different ways of giving input by the user as shown in figure 4.2.
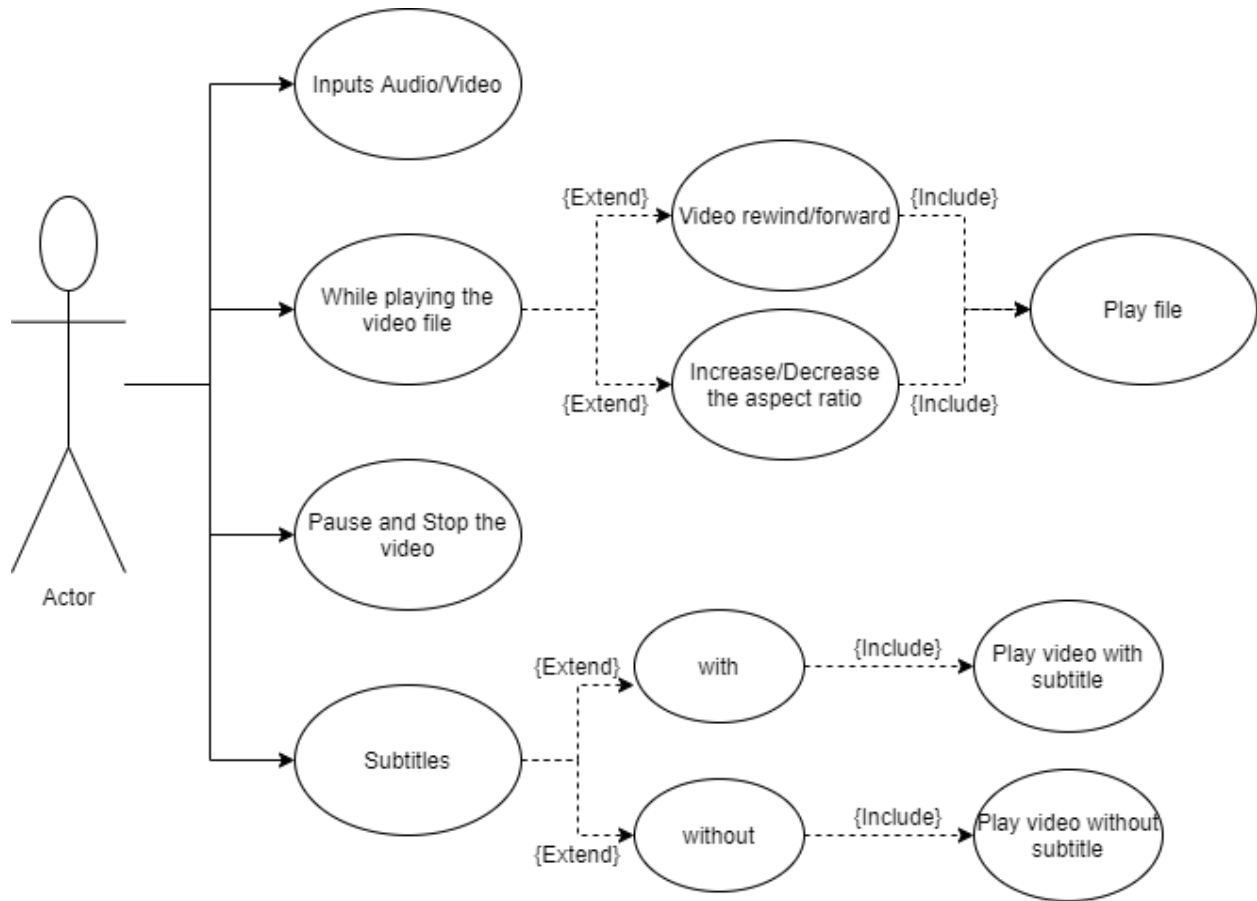
Figure 4.2 Use case diagram for users input

The various scenario of use case diagram includes,

- Users can input audio/video files.
- While playing a video user can either rewind or forward the video file or increase or decrease the aspect ratio and play the video file.
- Users can either pause or stop the video file.
- If the user selects the subtitle and plays the video with or without the subtitle.

4.3 MODULES DESCRIPTION

4.3.1 AUDIO EXTRACTION

The first stage is converting the video file to audio file and the process of converting them is shown in the activity diagram of figure 4.3.
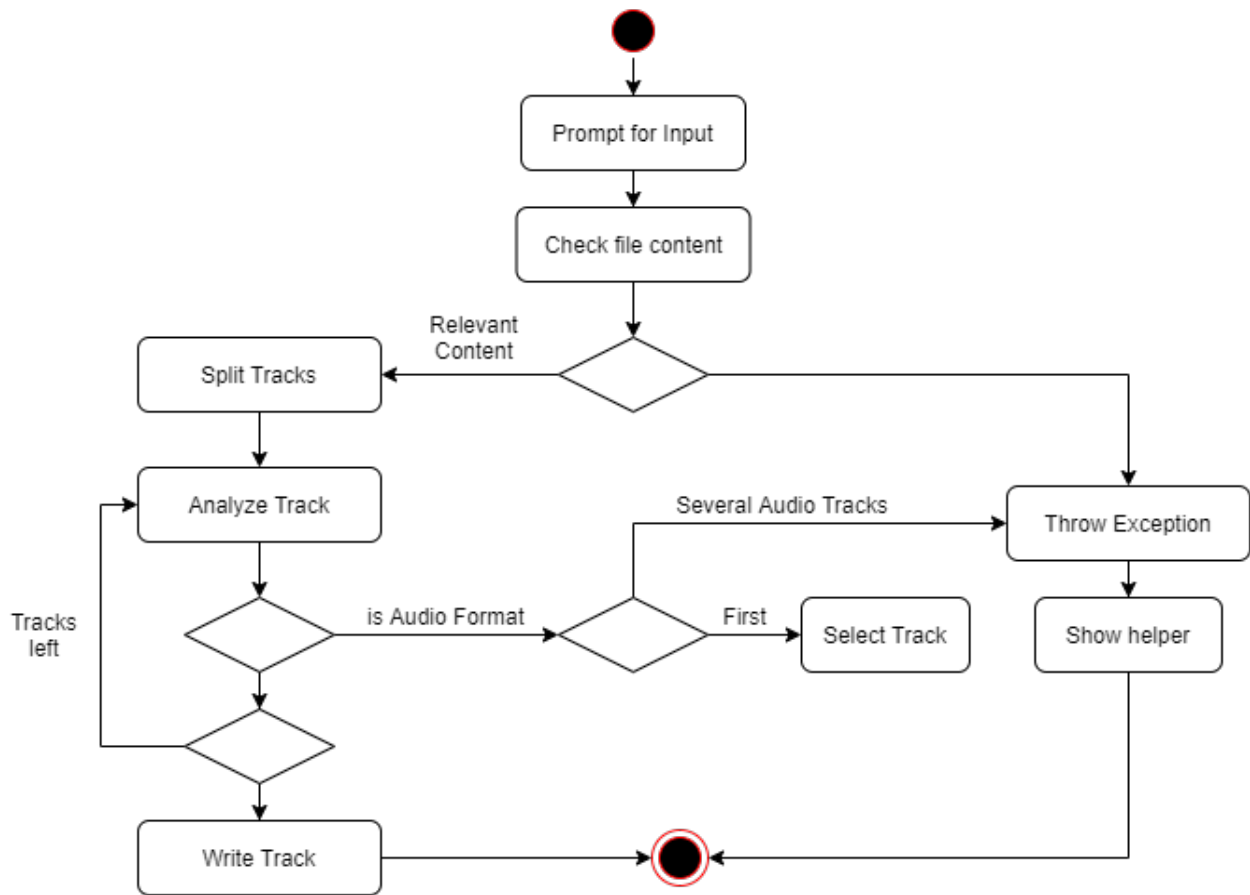
Figure 4.3 Activity diagram for Audio Extraction

This module aims to output an audio file from a media file. It takes as input a file URL and gives the audio format of the output. Next, it checks the file content and creates a list of individual tracks composing the initial media file. An exception is thrown if the file content is irrelevant. Once the tracks are separated, the processor analyzes each track, selects the first audio track found and discards the rest.

Finally, the audio track is written in a file applying the default or wished format. In Audio Formats Discussions, it was previously talked about the audio formats that Sphinx accepts. At the moment, it principally supports WAV or RAW files. Hence, the task of obtaining the preferred format is complicated. It often requires various treatments and conversions to reach the mentioned purpose.

For that purpose encoder uses FFMPEG to convert any media file to the audio file of preferred format where each track is processed separately and the audio format supported in encoder is FLAC file and Linear16 which uses lossy compression.

4.3.2 SPEECH RECOGNITION

After audio extraction from the video file the task is to recognize the speech of the audio file and during this recognize various factors need to be considered that is frequency of the audio file, how clear is the voice of the audio and if there is interruption in the audio file it need to get pre-processed such that the recognize of the text will be happen with more accuracy**.**

The block diagram for the speech recognition which represents the audio file is given as the input from that speech signal gets converted to the recognized text as shown in the figure 4.4.
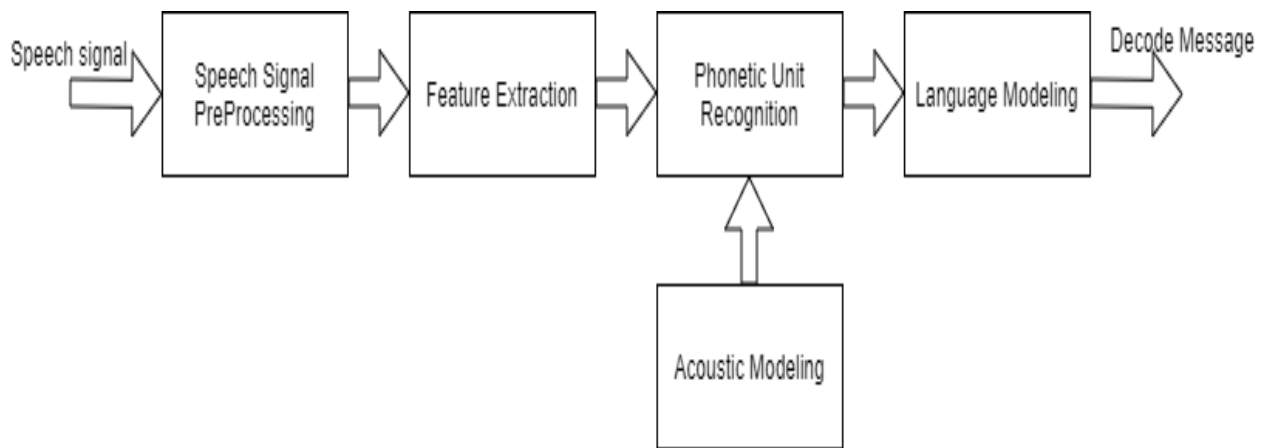

Figure 4.4 Block diagram for speech recognition

The audio signal is sent to the speech signal preprocessing which removes noise and any interrupted sounds in that signal. After preprocessing the features gets extracted that is the frequency and wavelength which has been given as the input to the phonetic unit recognition. The Phonetic unit recognition uses acoustic modelling for recognition of the signal frequency and uses the language modelling which uses the language code given by the input of the audio file and the result of speech recognition is achieved. In our project the speech recognition of these above phases are built using the cloud speech to text API.

4.3.3 SUBTITLE GENERATION

This module is expected to get a list of words and their respective speech time-frames from the speech recognition module and then produce a .srt subtitle file. To do so, the module must look at the list of words and use silence (SIL) spoken words as a boundary between for two consecutive sentences and also split based on the different speaker tag.

Subtitle Generation is expected to select the most suitable models considering the audio and the parameters (category, length, language etc.) given as input by the user and thereby generate a precise transcript of the audio speech.In this activity diagram shows how speech is read from real time and recognizing the text and generating the subtitle for the video file given as input as per the user requirements.

Therefore speech is recognized and subtitles are generated in .srt file. The below activity diagram for subtitle generation is shown in the figure 4.5
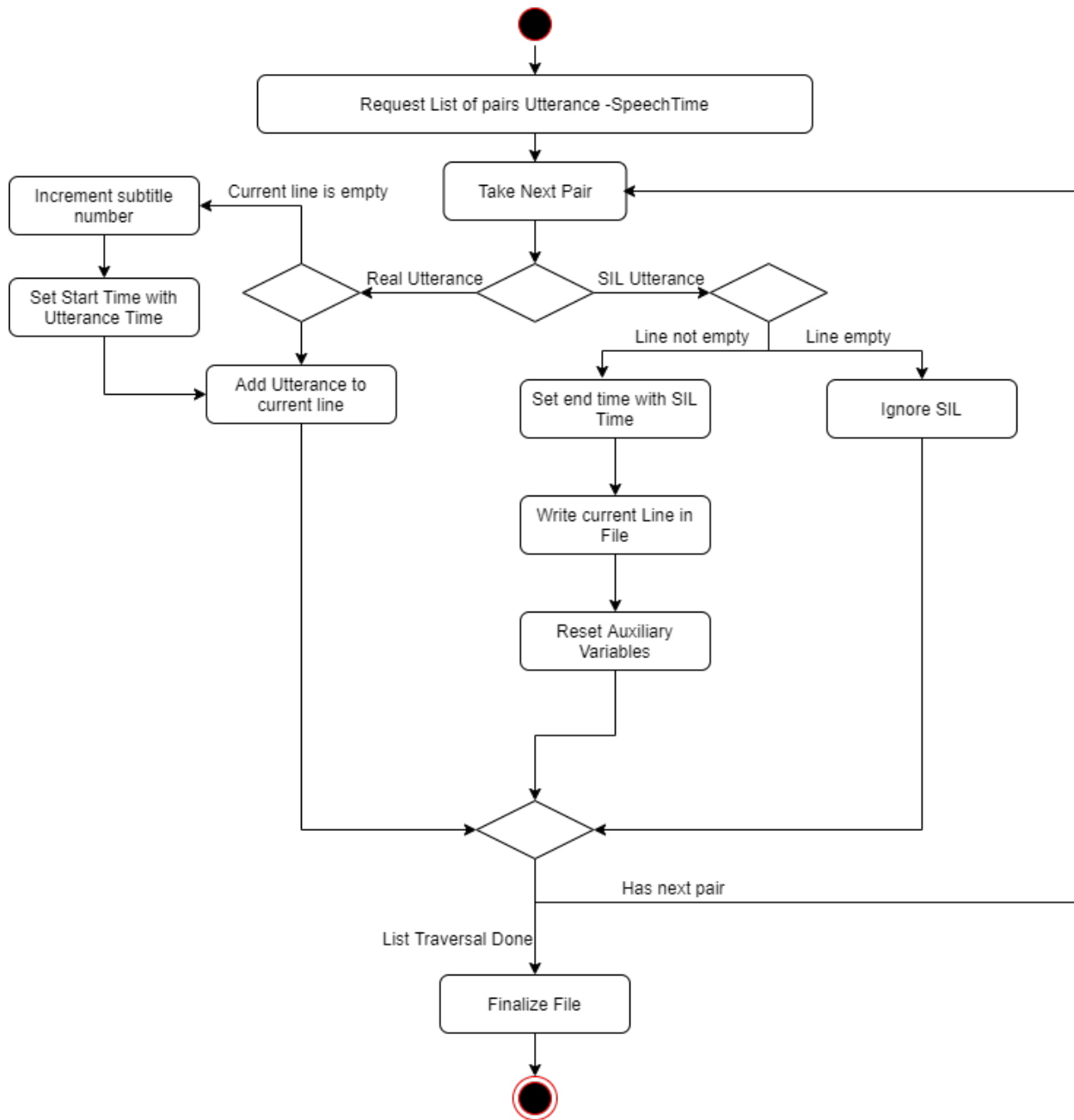
Figure 4.5 Activity Diagram for Subtitle Generation

The part of the activity diagram shows the creation of the subtitle file and the other half represents the formation of the sentences, that is check the work end time whether there is a gap of one sec, if then break the sentence and add it into the subtitle file else continue the work till it ends.Doing all this the subtitle gets generated and if user want it in our own language in this project encoder uses cloud translate API which generates the subtitle as per the user requirements.

CHAPTER 5
IMPLEMENTATION

Encoder uses python (version 3.0) with anaconda for the implementation of "Automatic Subtitle Encoder" using google translate API which have been trained with advanced neural networks. There are various phases involved during this implementation and deployment of our framework. Among them, audio extraction, speech recognition and speech translation phase are the main phases. Audio Extraction: converts an input file of any format supported by MPEG standards to audio format. Speech Recognition of the extracted audio file is implemented. Speech Translation where Subtitle Generated in which a.txt/.srt file is generated which is synchronized with the input video file.

5.1 Audio Feature Extraction

Figure 5.1 Audio Feature Extraction from Video

The input video file passes through demuxer where audio is separated from video and then, audio stream is divided into frames of binary format. In order to remove noise from audio streams, a compression algorithm is used with the help of the Psychoacoustic Model. This model provides lossy compression of the signal. This resulting binary data is converted to sinusoidal signal to pass as input to muxer where the separated audio signals are combined together under file extension format.

#Audio Extraction from a video file

command="ffmpeg -i "+video_filename.replace(" ","")+" "+video_filename[:length-4].replace(" ","")+".flac"

Run the command in the shell script using pop method which uses synchronous execution, which uses FFMPEG to convert a video file to audio file with lossy compression of signal.
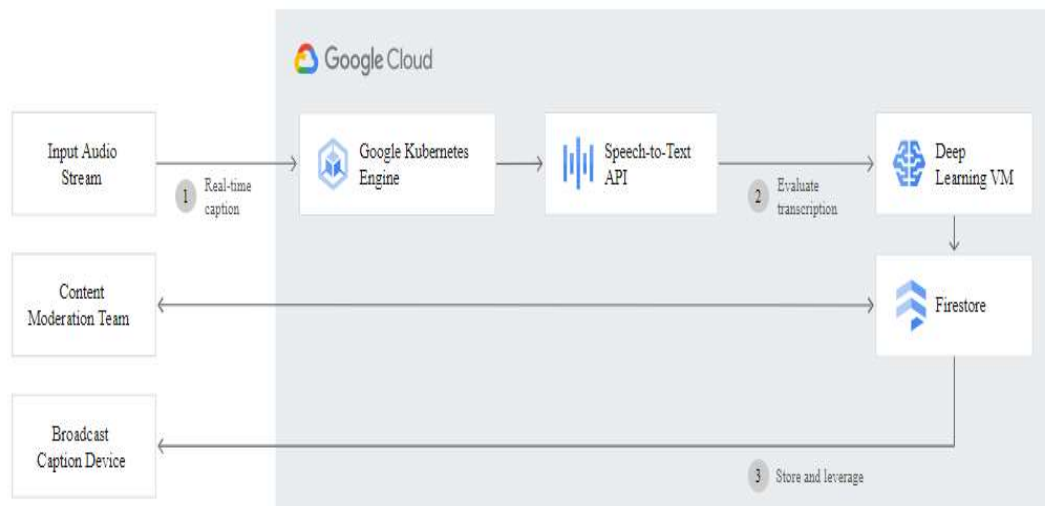
5.2 Speech to Text Recognition

Figure 5.2 Transcribing Multimedia Content

- Apply powerful neural community fashions to convert speech to text. Recognises greater than 110 languages and variants. Text effects in actual-Time. A hit noise dealing with. Helps gadgets that can ship a relaxation or gRPC request. API consists of time offset values (timestamps) for the start and gives up of every word spoken in the known audio.

- A handful of applications for speech popularity exist on PyPI. Some of them built integrated:

    ➢ apiai
    ➢ assemblyai
    ➢ Google-cloud-speech
    ➢ pocket sphinx

- Speech Recognition
- watson-developer-cloud

- A number of those programs—which built integrated wit and apiai—offer capabilities, like natural language process built-in for built-in integrated a speaker's built integrated, which pass beyond simple speech popularity. Others, like Google-cloud-speech, recognize totally on speech-to-text conversion. There's one package deal that stands out built-in terms of ease-of-use: Speech Recognition.

- Built-in spot built integrated speech calls for audio built-in, and Speech Recognition makes retrieve integrated this built-in built integrated built-in smooth. as a substitute to get integrated to build scripts for built-in integrated microphones and process integrated audio files from scratch, Speech Recognition will have you ever up and jog integrated built-in only some built-in.

- The Speech Recognition library acts as a wrapper for several famous speech APIs and is for that reason built-in bendy. such a—the Google built integrated Speech API—helps a default API key that is tough-coded built-into the Speech Recognition library. Built-in you can get off your toes without having to enroll in an integrated service.

- The ability and ease-of-use of the Speech Recognition package deal make it an outstand built integrated choice for any Python venture. But, the guide for every feature of each API it wraps isn't assured. you will need to spend a while getting to know the available alternatives to built-in if Speech Recognition will work built into an integrated unique case.

- Now, the next phase is to convert an audio file to speech, where an extracted audio file is sent as an input which recognizes words of that language using Speech Recognition in which the audio undergoes three modules, the Front End, Decoder and the Knowledge Base.

- Words that have more probability of occurrence get a high score and the one with lower probability are pruned using the pruner.

- Transcribe your audio and video with subtitles and improve your audience's reach

and experience. Add subtitles to your streaming content in real time. Our video transcription model is ideal for indexing or subtitling videos and / or content from multiple speakers and uses machine learning technology similar to video subtitling on YouTube.

5.2.1 Configuration Setup

- The above modules are implemented using machine learning approaches which have been hosted by a Google cloud called cloud speech to text API. To do this encoder need to establish a connection with the cloud using the credentials provided which is in JSON format.

#establishing connection with Google cloud service

Credentials = service_account.Credentials.from_service_account_file(obj)

storage client=storage.Client.from_service_account_json(obj)

Configuration have to be done and the necessary data that we need to provide are audio language in which language we need to recognize the words, audio format in which we are providing our input, audio_channel_count in which the audio format header is requested and time offset which gives the each words with start_time and end_time.

#Configuration for speech recognition

Parameters:

- encoding=speech.RecognitionConfig.AudioEncoding.XXXX,
- language_code=language,
- audio_channel_count=X,
- enable_word_time_offsets=True

And there are some more parameters which are to be optional, used to give you some more additional functionalities. They are,

- **sampleRateHertz :** Sample rate in Hertz of the audio data file that was sent in all Recognition Audio content. Valid values are: 8000-48000.16000 is optimal. For best results, set the audio source sample rate to 16000 Hz. If this is not possible, use the native sample rate from the audio source (instead of resampling). This parameter is

optional for two audio files FLAC and WAV, but it is necessary for all other audio formats.

- **maxAlternatives :** Maximum number of detection hypotheses to be returned. In particular, the maximum number of Speech Recognition Alternative contents in each Speech Recognition Response. The server can return fewer than max Alternatives. Valid values are 0-30. A value of 0 or 1 returns a maximum of one. If it is omitted, a maximum of one is returned.

- **useEnhanced :** Set to true to use an improved model for speech recognition. If use Enhanced is set to true and the model field is not set, an appropriate extended model is selected if there is an extended model for the audio. If use Enhanced is true and there is no enhanced version of the specified model, the language is detected using the default version of the specified model.

- **Model:** Which model to choose for the specified requirement? For the best results, choose the model that works best for your domain. If a model is not explicitly specified, encoder automatically select a model based on the parameters in RecognitionConfig.ModelDescriptioncommand_and_searchBest for short queries such as voice commands or voice searches. Best for audio coming from a phone call (usually recorded at a sample rate of 8 kHz). Video Best for audio that comes from video or contains multiple speakers. Ideally, the audio is recorded at a sampling rate of 16 kHz or higher. This is a premium model that costs more than the standard plan. By default, best for audio that is not one of the specific audio models. Example: long format audio. Ideally, the audio is recorded with high fidelity and a sample rate of 16 kHz or higher.

- **enableAutomaticPunctuation :** If true, add a score to the recognition outcome hypotheses. This function is only available in selected languages. Setting it for requirements in other languages has no effect. The default value of false does not add a score to the result hypotheses.

- **diarizationConfig :** Settings for activating speaker diarization and setting additional parameters to make the diarization more suitable for your application. Note: When this option is activated, encoder send all the words from the beginning of the audio to the top alternative in each successive STREAMING response.

5.2.2 Recognition Methods

There are three methods in which encoder can process our request. They are,

● **Asynchronous Recognition**: (REST and gRPC) sends audio file to the Speech-to-Text API and initiates a *long_running_operation*. Using this operation, you can periodically poll for recognition responses. Use asynchronous requests for audio files of any duration up to 480 minutes.

● **Streaming Recognition**: (gRPC only) performs recognition on audio file provided within a gRPC bi-directional. Requests are designed for real-time speech recognition purposes, such as capturing live audio from a user microphone. Streaming recognition provides accurate results while audio is being captured, allowing response to appear, for example, while a user is still speaking.

● **Synchronous Recognition:** (REST and gRPC) sends audio file to the Speech-to-Text API, performs speech recognition on that audio file, and returns response after all audio has been processed. Synchronous recognition requests are limited only to audio size of 1 minute or less in duration.

#Speech Recognition

operation = client.long_running_recognize(config=config,audio=audio)

result = operation.result()

In this project encoder uses Asynchronous method for recognition. For this encoder has to upload our audio file in cloud, for storage create bucket in cloud and store the file by using storage_client method. Using the type recognition the audio file contains the field of uri which is the pointer to the audio content, which keeps track of audio file and recognizes each word using cloud speech to text API.

#Bucket creation

-Create Bucket in the cloud with an object.

-Establish connection between bucket and audio.

-Upload the file.

5.2.3 Processing Response

Once it has been recognized it contains the response containing each word with its start_time, end_time, confidentiality etc, encoder need to extract the necessary data from the response for further processing. Example response is shown in Fig 5.4

```
{
  "name":"XXXXXXXXXX",
  "metadata":{
    "@type":"XXXXXXXXXX",
    "progressPercent":X,
    "startTime":"YYYY-MM-DDTXXXX",
    "lastUpdateTime":"YYYY-MM-DDTXXXX"},
    "done":true,
    "response":{
    "@type":"XXXXXXXXX",
    "results":[{

      "alternatives":[{
              "transcript":"Fourscoreandtwenty...(etc)...",
              "confidence":XXXXXX,
              "words":[{
                      "startTime":"XXXs",
                      "endTime":"XXXs",
                      "word":"Four"
                      },

                      ...
                      ]}}]},
              {
              "alternatives":[{
                      "transcript":"forscoreandplenty...(etc)...",

                      "confidence":XXXX,
                          }]
          }]}
}
```

Figure 5.3 Google Speech to Text Recognition API Response

5.2.4 Sentence Formation

After extraction, words should combined in such a way which forms a meaningful sentence, for this encoder uses two approach,

29

- Based on the time gap of one sec.

- Based on different speaker tags.

- Based on if the time exceeds the certain limit, and then encoder can break the sentence for the subtitle to be either one or two lines not more than that. Form sentences with start time and end time and give the input to the speech translation.

The response that we obtain from the subtitle encoder is well processed which comprise of confidence field along with generated transcript for the given audio source file. This also consist of start time and end time parameters which are helpful to generate the sub ripped file for the required video file. The processed response is shown in below Figure 5.4

```
results {
  alternatives {
    transcript: "you are listening to episode 21 of the Casual core podcast released on August 10t|
    confidence: 0.9257831573486328
    words {
      start_time {
        nanos: 600000000
      }
      end_time {
        seconds: 1
        nanos: 100000000
      }
      word: "you"
    }
    words {
      start_time {
        seconds: 1
        nanos: 100000000
      }
      end_time {
        seconds: 1
        nanos: 200000000
      }
      word: "are"
    }
```

Figure 5.4 Processed Response from Subtitle Encoder
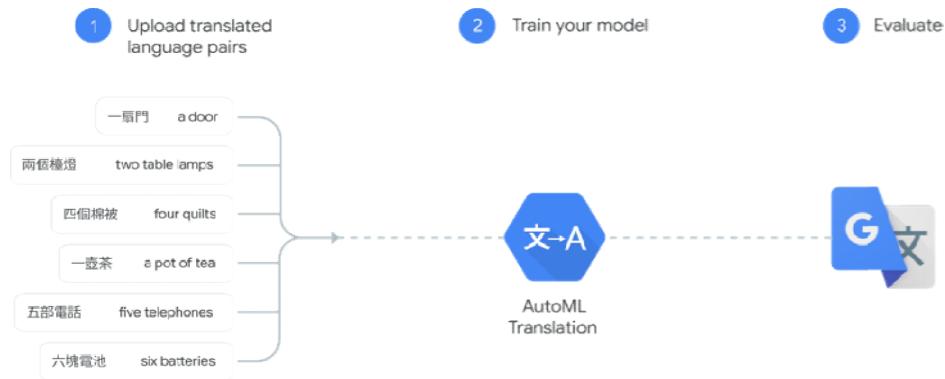
5.3 Speech Translation

Figure 5.5 Google API Speech Translation

The above Figure 5.5 explains the detailed structure and code flow inside Google API cloud Engine. It explains the necessary steps that undergoes whenever user creates a record or entry in Google API speech Translation. These are taken care by Google automated Speech Recognition itself and user need to provide only input audio format file of respective file format that are available in Google Speech Recognition API.

Speech translation is translating from one language to another language, where encoder have many models in which translation is done with neural networks and machine learning. Cloud Translation can dynamically translate textual content among hundreds of language pairs. Translation done through web sites and packages programmatically combines with the translation carrier. In addition to translating text, detecting supply language, and getting a list of supported languages, the superior version additionally helps custom glossaries, Batch translation, AutoML fashions, Labels

In this project encoder uses neural network based model for speech translation. And the model is called NMT (Neural Machine Translation. The translate API recognition engine which supports a huge variety of languages is made for the Neural Machine Translation (NMT) model.

The languages are specified within a recognition request using language code as a parameter in the request into which the language is to be translated. Most language code

parameters are compatible with ISO-639-1 identifiers. You can also list the supported languages for translation using Cloud Translation or in Basic or Cloud Translation Advanced APIs. The languages that are available at present are shown in the below Figure 5.6.

```
{'af': 'afrikaans', 'sq': 'albanian', 'am': 'amharic', 'ar': 'arabic', 'hy': 'armenian', 'az': 'azerbaijani', 'e
u': 'basque', 'be': 'belarusian', 'bn': 'bengali', 'bs': 'bosnian', 'bg': 'bulgarian', 'ca': 'catalan', 'ceb': 'ce
buano', 'ny': 'chichewa', 'zh-cn': 'chinese (simplified)', 'zh-tw': 'chinese (traditional)', 'co': 'corsican', 'h
r': 'croatian', 'cs': 'czech', 'da': 'danish', 'nl': 'dutch', 'en': 'english', 'eo': 'esperanto', 'et': 'estonia
n', 'tl': 'filipino', 'fi': 'finnish', 'fr': 'french', 'fy': 'frisian', 'gl': 'galician', 'ka': 'georgian', 'de':
'german', 'el': 'greek', 'gu': 'gujarati', 'ht': 'haitian creole', 'ha': 'hausa', 'haw': 'hawaiian', 'iw': 'hebre
w', 'hi': 'hindi', 'hmn': 'hmong', 'hu': 'hungarian', 'is': 'icelandic', 'ig': 'igbo', 'id': 'indonesian', 'ga':
'irish', 'it': 'italian', 'ja': 'japanese', 'jw': 'javanese', 'kn': 'kannada', 'kk': 'kazakh', 'km': 'khmer', 'k
o': 'korean', 'ku': 'kurdish (kurmanji)', 'ky': 'kyrgyz', 'lo': 'lao', 'la': 'latin', 'lv': 'latvian', 'lt': 'lith
uanian', 'lb': 'luxembourgish', 'mk': 'macedonian', 'mg': 'malagasy', 'ms': 'malay', 'ml': 'malayalam', 'mt': 'mal
tese', 'mi': 'maori', 'mr': 'marathi', 'mn': 'mongolian', 'my': 'myanmar (burmese)', 'ne': 'nepali', 'no': 'norweg
ian', 'ps': 'pashto', 'fa': 'persian', 'pl': 'polish', 'pt': 'portuguese', 'pa': 'punjabi', 'ro': 'romanian', 'r
u': 'russian', 'sm': 'samoan', 'gd': 'scots gaelic', 'sr': 'serbian', 'st': 'sesotho', 'sn': 'shona', 'sd': 'sindh
i', 'si': 'sinhala', 'sk': 'slovak', 'sl': 'slovenian', 'so': 'somali', 'es': 'spanish', 'su': 'sundanese', 'sw':
'swahili', 'sv': 'swedish', 'tg': 'tajik', 'ta': 'tamil', 'te': 'telugu', 'th': 'thai', 'tr': 'turkish', 'uk': 'uk
rainian', 'ur': 'urdu', 'uz': 'uzbek', 'vi': 'vietnamese', 'cy': 'welsh', 'xh': 'xhosa', 'yi': 'yiddish', 'yo': 'y
oruba', 'zu': 'zulu', 'fil': 'Filipino', 'he': 'Hebrew'}
```

Figure 5.6 Languages that exists in Subtitle Encoder

The available languages are most spoken by the people all over the world. These languages are coded along with their specific mnemonic such as for English it is "eu" And for French it is "Fr". These encoded mnemonics are obtained and used universally by all language specific fields. Each mnemonic corresponds only to a particular language. Thus we can identify any language with the help of mnemonic itself.

CHAPTER 6
TESTING

Testing is the phase in which a system or its component undergoes certain evaluation with the intent to find whether it has any bug or it will meet the system requirements. This activity results in the actual, expected and difference between their results. Testing has many strategies which vary from component to component.

6.1 Testing strategies

With the intent of finding errors, the different phases of testing strategies are applied as shown in Fig 6.1.These strategies are not completely required in all the software development process.
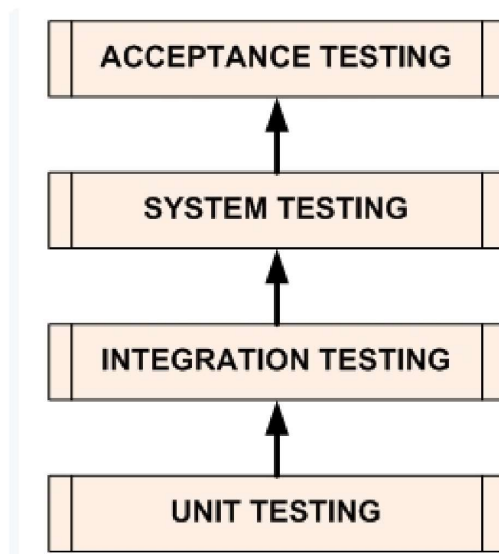
Figure 6.1 Software Development Life Cycle Testing Phases

6.1.1 Unit Testing

The primary objective of unit testing is to separate each part of the program and check whether individual parts are correct in terms of functionality. Each module in the tool such as audio extraction, speech recognition and machine translation is tested separately with the help of,

- Video file is given as input to the audio extraction.
- Extracted audio file is given as input to the speech recognition and
- Recognized speech formed as a sentence and given as input to the machine translation.

6.1.2 Integration Testing

In this testing, the tested unit modules from the unit testing are combined and then checked for integration testing. The approach followed here is Bottom-up Integration testing where the smallest modules are tested first and then combined together. Where video file is converted to audio file (audio extraction) and audio is recognized using cloud speech to text API (speech recognition) is tested and later the sentence formation is tested with the list of words and finally the translation, finally all these modules are tested in a combined manner.

6.1.3 System Testing

This is the next level in the testing and tests the system as a whole. The integrated tested modules are again combined into a complete software module which is checked rigorously. The complete system is tested with few video files which are given as input by the user and tested whether it gives the output as expected.

6.1.4 Acceptance Testing

The goal of this acceptance testing is to find whether the software development model meets the client specification or not. Model has been subjected to both alpha and beta testing and gathered their feedback. And the project has been tested with feedback and rectified and worked successfully.

6.2 Validation

All the levels in the testing (unit, integration, system) are implemented in our application successfully and the results obtained as expected. Validation and Verification phase are more important in any Software Development Life cycle such as agile, V model, waterfall model etc… In each project, almost 20 percent of the job use to be validation and verification. Creators usually release alpha roll out before proceeding with Validation.

6.3 Limitations

Though the time to recognize the speech from the audio files takes longer for larger files which are the average of the total length of the video file and for real time applications it is not come into use.

6.4 Test Results

The alpha testing is done among the team members and the beta testing is done with the help of college end users. It satisfies the end users requirements and the results obtained from encoder has an accuracy of about 92.37%.

CHAPTER 7

RESULT ANALYSIS

7.1 Graphical User Interface

7.1.1 Home screen

      The first screen will ask the user to choose the path of the video file in the first textbox or either choose the url of the youtube video file for which the user wants to generate the subtitle and the user needs to give the source language and the target language. The entire GUI was created with Python Tkinter. The home screen of our application is shown in Fig 7.1.
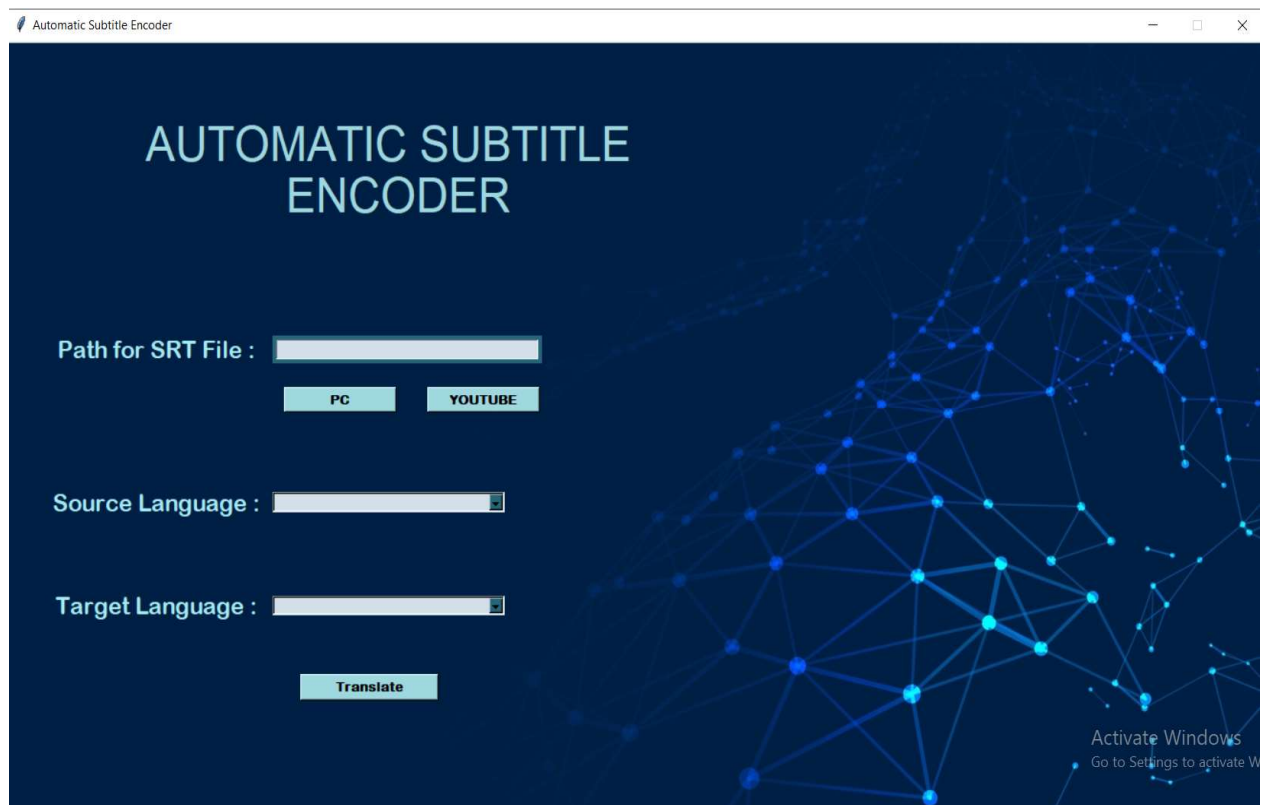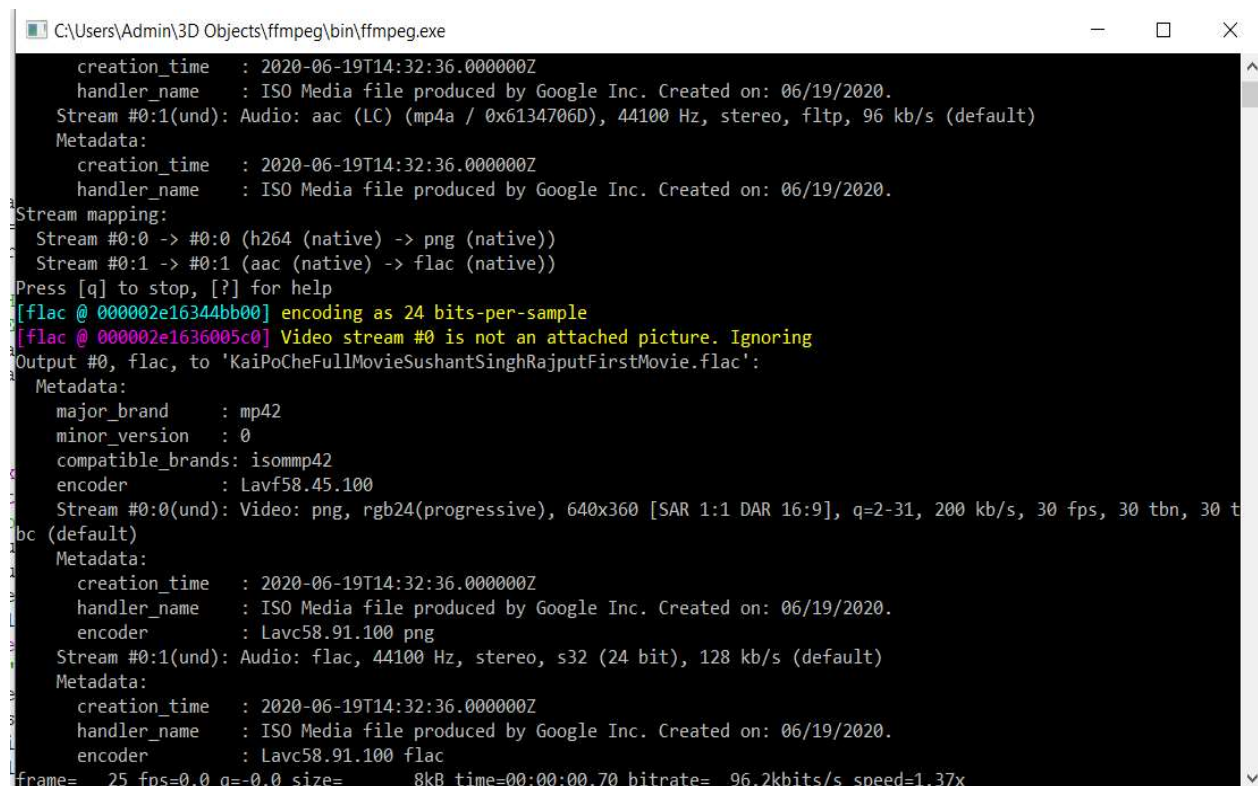

Figure 7.1 Encoder's Home Screen

7.2 Subtitle Generation

7.2.1 Processing Video file

The next screen from the home screen is based on the user's choice in the home screen. After choosing the path from pc the video files gets converted to audio file with the same name and the format is FLAC else the url gets downloaded from youtube and gets converted to audio file with same name and format is FLAC. Processing video file is shown in Fig 7.2.

Figure 7.2 Audio Feature Extraction

7.2.2 Speech Recognition

The next phase is that when user choose the source and target language and after clicking on translate button using the cloud speech to text API the speech is recognized from the audio file and from the list of words sentences gets formed and for view purpose the data is put into the excel file which is shown in Fig 7.3

Figure 7.3 Speech to Text Recognition

7.2.3 Machine translation

And the last phase is that converting the recognized speech into the user required target language where encoder checks the words are in utf-8 format and sends this message as input and using a cloud translate API to do this subtitle generation where the translated text is shown in fig 7.4

Figure 7.4 Machine Translation from Source to Target Language

7.3 SRT file

After the translation encoder has to generate the .srt file called "SubRip subtitle", which contains the format of

start time ---> end time

segment number

hours : minutes : seconds, milliseconds ----> hours : minutes : seconds, milliseconds

new line space

and the custom generated srt file is shown in the fig 7.5.

```
1
00:00:06,000 --> 00:00:12,000
செயற்கை நுண்ணறிவு என்பது கணினி அமைப்புகளால் மனித நுண்ணறிவை சிதைப்பதாகும்

2
00:00:12,000 --> 00:00:18,000
மனித நடத்தைக்கு குறைந்தபட்சம் ஒரு அம்சத்தையாவது கிட்டத்தட்ட சமமான வகையில் பிரதிபலிக்கும் திறன் கொண்டது

3
00:00:18,000 --> 00:00:25,000
இன்று ஒரு மனிதனிடமிருந்து பிரித்தறிய முடியாதது குறுகிய AI   க்கு பல எடுத்துக்காட்டுகள் உள்ளன

4
00:00:25,000 --> 00:00:31,000
ஒரு கணினி மிகவும் சிறப்பாக இருக்கும் போது, எடுத்துக்காட்டாக வீடியோ ஸ்ட்ரீம்களில் உரை எதிர்கொள்ளும்

5
00:00:31,000 --> 00:00:37,000
டோல் சாலைகளில் உரிமத் தகடுகளை ஒழுங்கமைக்கவும் அல்லது பல தொழில்கள் எடுக்கக்கூடிய குரல் கட்டளைகளைப் புரிந்து கொள்ளவும்

6
00:00:37,000 --> 00:00:43,000
ai  இன் நன்மை மற்றும் அவை அனைத்தும் AI  shpe  ஆல் திசைதிருப்பப்பட்டு அவை இல்லாமல் அடிமைத்தனத்தை எடுக்கின்றன

7
00:00:43,000 --> 00:00:48,000
உள்கட்டமைப்பை மேம்படுத்துவதன் மூலம் நீங்கள் தேவையான மென்பொருள் அடுக்கைக் கண்டுபிடிப்பீர்கள்
```

Figure 7.5 Generated SubRip Subtitle File

7.4 Final output

Now the final output subtitle file is merged with the video file using the command which will run in the shell script using the pop method that uses synchronous method for its execution which is shown in the fig 7.6
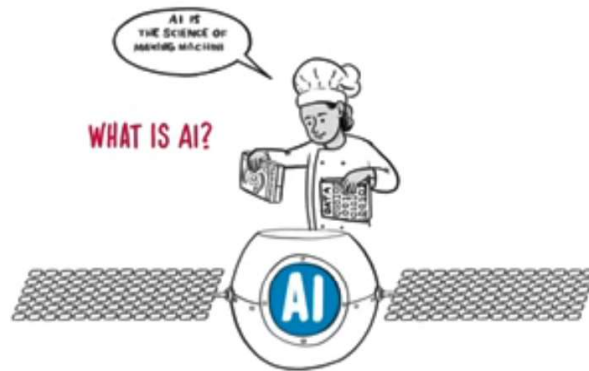
Figure 7.6 Embedded Subtitle within Source Video File

CHAPTER 8
CONCLUSION

8.1 CONCLUSION

In today's modern era, language plays a vital role in business needs and performing trade around the world. And also with the existence of more languages, it makes it tedious for the user to understand the end users words. Manually, people can convert from one language to another with the help of people who know both languages. The state of this process is known as dubbing. But one cannot remember all the languages that exist in the world. People who have auditory problems also suffer from watching video files without any audio. Thus, to help people who are suffering from auditory problems can make use of captions to understand the internal meaning of the respective video. But, these captions also are made available only in a few languages. To generate an automatic subtitle or caption generator, users can use this model called "Automatic Subtitle Encoder". This encoder helps people to select their desired language from the list of options that are available. Encoder encodes the respective source to target language with an accuracy of 90.5%.

8.2 FUTURE ENHANCEMENTS

Till now, Encoder has encoded only 66 languages. In future, it will be implemented with all native languages that exist. Encoder generates only sub ripped subtitle files only. In future, it will search for a suitable source file and embed the srt file automatically to the video file. Initially, time taken to generate the first instance of text will be 2 minutes. In the future with the help of the multi threading concept, subtitle generation time can be reduced. Accuracy will be increased in future to 90% and above.

APPENDIX

#Importing necessary library

```
import six
import subprocess
import ffmpeg
from google.oauth2 import service_account
from google.cloud import translate_v2 as translate
import xlwt
from xlwt import Workbook
import time
import pysrt
import xlrd
import os
from google.cloud import storage
import shlex
import ntpath
import tkinter as tk
from tkinter import
Tk,PhotoImage,filedialog,Canvas,Button,OptionMenu,Label,Text,Entry,ttk
from PIL import Image,ImageTk
```

#Establishing connection with Google cloud service
```
obj="XXXXXXXXXXXXXXXXXX.json"
buck_name="<Any_name>"
credentials = service_account.Credentials.from_service_account_file(obj)
storage_client=storage.Client.from_service_account_json(obj)
```

#Defining available languages
```
def source(source_input):
    source_language={
"Afrikaans(SouthAfrica)":"af-ZA",
"Arabic(Bahrain)":"ar-BH",
"Arabic(Egypt)":"ar-EG",
"Arabic(Iraq)":"ar-IQ",
"Arabic(Israel)":"ar-IL",
"Arabic(Jordan)":"ar-JO",
"Arabic(Kuwait)":"ar-KW",
"Arabic(Lebanon)":"ar-LB",
"Arabic(Oman)":"ar-OM",
"Arabic(Qatar)":"ar-QA",
"Arabic(Saudi Arabia)":"ar-SA",
"Arabic(State of Palestine)":"ar-PS",
```

"Arabic(United Arab Emirates)":"ar-AE",
"Bengali(Bangladesh)":"bn-BD",
"Chinese,Cantonese(Traditional HongKong)":"yue-Hant-HK",
"Chinese,Mandarin(Traditional,Taiwan)":"zh-TW ",
"Czech(Czech Republic)":"cs-CZ",
"Danish(Denmark)":"da-DK",
"Dutch(Netherlands)":"nl-NL",
"English(Australia)":"en-AU",
"English(Ghana)":"en-GH",
"English(India)":"en-IN",
"English(Nigeria)":"en-NG",
"English(Philippines)":"en-PH",
"English(Singapore)":"en-SG",
"English(South Africa)":"en-ZA",
"English(Tanzania)":"en-TZ",
"English(United Kingdom)":"en-GB",
"English(United States)":"en-US",
"Filipino(Philippines)":"fil-PH",
"Finnish(Finland)":"fi-FI",
"French(Canada)":"fr-CA",
"French(France)":"fr-FR",
"German(Germany)":"de-DE",
"Gujarati(India)":"gu-IN",
"Hebrew(Israel)":"iw-IL",
"Hindi(India)":"hi-IN",
"Indonesian(Indonesia)":"id-ID",
"Italian(Italy)":"it-IT",
"Japanese(Japan)":"ja-JP",
"Kannada(India)":"kn-IN",
"Korean(South Korea)":"ko-KR",
"Malay(Malaysia)":"ms-MY",
"Malayalam(India)":"ml-IN",
"Marathi(India)":"mr-IN",
"NorwegianBokmål(Norway)":"no-NO",
"Persian(Iran)":"fa-IR",
"Polish(Poland)":"pl-PL",
"Portuguese(Brazil)":"pt-BR",
"Portuguese(Portugal)":"pt-PT",
"Russian(Russia)":"ru-RU",
"Serbian(Serbia)":"sr-RS",
"Spanish(Spain)":"es-ES",
"Spanish(United States)":"es-US",
"Swedish(Sweden)":"sv-SE",
"Telugu(India)":"te-IN",
"Thai(Thailand)":"th-TH",

```python
"Turkish(Turkey)":"tr-TR",
"Ukrainian(Ukraine)":"uk-UA",
"Urdu(Pakistan)":"ur-PK",
"Vietnamese(Vietnam)":"vi-VN",
"Zulu(South Africa)":"zu-ZA"   }
    return source_language[source_input]

def target(target_input):
    target_language={
"Afrikaans":"af",
"Albanian":"sq",
"Amharic":"am",
"Arabic":"ar",
"Armenian":"hy",
"Azerbaijani":"az",
"Basque":"eu",
"Belarusian":"be",
"Bengali":"bn",
"Bosnian":"bs",
"Bulgarian":"bg",
"Catalan":"ca",
"Cebuano":"ceb",
"Chinese(Simplified)":"zh-CN",
"Chinese(Traditional)":"zh-TW",
"Corsican":"co",
"Croatian":"hr",
"Czech":"cs",
"Danish":"da",
"Dutch":"nl",
"English":"en",
"Esperanto":"eo",
"Estonian":"et",
"Finnish":"fi",
"French":"fr",
"Frisian":"fy",
"Galician":"gl",
"Georgian":"ka",
"German":"de",
"Greek":"el",
"Gujarati":"gu",
"Haitian Creole":"ht",
"Hausa":"ha",
"Hawaiian":"haw",
"Hebrew":"he",
"Hindi":"hi",
```

"Hmong":"hmn",
"Hungarian":"hu",
"Icelandic":"is",
"Igbo":"ig",
"Indonesian":"id",
"Irish":"ga",
"Italian":"it",
"Japanese":"ja",
"Javanese":"jv",
"Kannada":"kn",
"Kazakh":"kk",
"Khmer":"km",
"Kinyarwanda":"rw",
"Korean":"ko",
"Kurdish":"ku",
"Kyrgyz":"ky",
"Lao":"lo",
"Latin":"la",
"Latvian":"lv",
"Lithuanian":"lt",
"Luxembourgish":"lb",
"Macedonian":"mk",
"Malagasy":"mg",
"Malay":"ms",
"Malayalam":"ml",
"Maltese":"mt",
"Maori":"mi",
"Marathi":"mr",
"Mongolian":"mn",
"Myanmar(Burmese)":"my",
"Nepali":"ne",
"Norwegian":"no",
"Nyanja(Chichewa)":"ny",
"Odia(Oriya)":"or",
"Pashto":"ps",
"Persian":"fa",
"Polish":"pl",
"Portuguese(Portugal,Brazil)":"pt",
"Punjabi":"pa",
"Romanian":"ro",
"Russian":"ru",
"Samoan":"sm",
"ScotsGaelic":"gd",
"Serbian":"sr",
"Sesotho":"st",

```
"Shona":"sn",
"Sindhi":"sd",
"Sinhala(Sinhalese)":"si",
"Slovak":"sk",
"Slovenian":"sl",
"Somali":"so",
"Spanish":"es",
"Sundanese":"su",
"Swahili":"sw",
"Swedish":"sv",
"Tagalog(Filipino)":"tl",
"Tajik":"tg",
"Tamil":"ta",
"Tatar":"tt",
"Telugu":"te",
"Thai":"th",
"Turkish":"tr",
"Turkmen":"tk",
"Ukrainian":"uk",
"Urdu":"ur",
"Uyghur":"ug",
"Uzbek":"uz",
"Vietnamese":"vi",
"Welsh":"cy",
"Xhosa":"xh",
"Yiddish":"yi",
"Yoruba":"yo",
"Zulu":"zu"  }
    return target_language[target_input]
```

#function to generate automatic subtitle based on user requirements

```
def subtitle_gen(gcs_uri,language,to_language,video_filename,output_filename):
    #Configuration done + Speech recognition
    from google.cloud import speech
    client = speech.SpeechClient(credentials=credentials)
    audio = speech.RecognitionAudio(uri=gcs_uri)
    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.FLAC,
        language_code=language,
        audio_channel_count=2,
        enable_word_time_offsets=True)
    operation = client.long_running_recognize(config=config,audio=audio)
    result = operation.result()
    #From a huge data,collecting necessary data and storing in list
```

```
json = []
for section in result.results:
    data = {
        "transcript": section.alternatives[0].transcript,
        "words": []}
    for word in section.alternatives[0].words:
        data["words"].append({
            "word": word.word,
            "start_time": word.start_time.total_seconds(),
            "end_time": word.end_time.total_seconds(),
            "speaker_tag": word.speaker_tag
        })
    json.append(data)
```

#From the list of words ,forming a sentence based on silence or different speaker tag

```
sentences = []
sentence = {}
for result in json:
    for i, word in enumerate(result['words']):
        wordText = word['word']
        if not sentence:
            sentence = {language: [wordText],'speaker': word['speaker_tag'],'start_time': word['start_time'],'end_time': word['end_time']}
```

        # If we have a new speaker, save the sentence and create a new one:

```
        elif word['speaker_tag'] != sentence['speaker']:
            sentence[language] = ' '.join(sentence[language])
            sentences.append(sentence)
            sentence = {language: [wordText],'speaker': word['speaker_tag'],'start_time': word['start_time'],'end_time': word['end_time']}
        else:
            sentence[language].append(wordText)
            sentence['end_time'] = word['end_time']
```

        # If there's greater than one second gap, assume this is a new sentence

```
        if((i+6< len(result['words'])) and ((word['end_time'] < result['words'][i+1]['start_time']) or (sentence['start_time']+10 < sentence['end_time']))):

            sentence[language] = ' '.join(sentence[language])
            sentences.append(sentence)
            sentence = {}
    if sentence:
        sentence[language] = ' '.join(sentence[language])
        sentences.append(sentence)
        sentence = {}
```

#Converting sentence into the target language and storing in Excel sheet with the

timestamp
```
    wb = Workbook()
    row,column,index=0,0,0
    sheet1 = wb.add_sheet('DATA')
    for var in sentences:
        input1=var[language]
        start_time=var['start_time']
        end_time=var['end_time']
        translate_client = translate.Client(credentials=credentials)
        if isinstance(input1,six.binary_type):
            input1=input1.decode("utf-8")
        result = translate_client.translate(input1, target_language=to_language)
        sheet1.write(row, column, index)
        index+=1
        column+=1
        sheet1.write(row, column, time.strftime('%H:%M:%S',time.gmtime(start_time)))
        column+=1
        sheet1.write(row, column, time.strftime('%H:%M:%S',time.gmtime(end_time)))
        column+=1
        sheet1.write(row, column, result['translatedText'])
        row+=1
        column=0
    wb.save('DATA.xls')

    #Creating a custom srt file and merging with the video file
    wb=xlrd.open_workbook("DATA.xls")
    sheet=wb.sheet_by_index(0)
    row,column=0,0
    total=sheet.nrows
    with open("subtitle.srt","w",encoding='UTF-8') as f:
        while(total):
            total-=1
            f.write(str(int(sheet.cell_value(row,column))+1))
            column+=1
            f.write("\n")
            f.write(str(sheet.cell_value(row,column))+",000")
            column+=1
            f.write(" --> ")
            f.write(str(sheet.cell_value(row,column))+",000")
            column+=1
            f.write("\n")
            f.write(sheet.cell_value(row,column))
            f.write("\n")
            f.write("\n")
            row+=1
```

```
            column=0
    command="ffmpeg -i "+video_filename+" -i subtitle.srt -c:s mov_text -c:v copy -c:a
copy "+output_filename
    args=shlex.split(command)
    subprocess.Popen(args)
    root.destroy()
```

#main function
```
def call_main(video_filename):
    global length
    length=len(video_filename)
    command="ffmpeg -i "+video_filename.replace(" ","")+" "+video_filename[:length-
4].replace(" ","")+".flac"
    args=shlex.split(command)
    subprocess.call(args)
```

#Methods for GUI Buttons
```
def pc_click():
    filename=filedialog.askopenfilename()
    path_text_obj.insert(0,filename)
    global video_filename
    video_filename=ntpath.basename(filename)
    call_main(video_filename)

def u_click():
    try:
        from pytube import YouTube
        from pytube import Playlist
    except Exception as e:
        print("Error")
    url=path_text_obj.get()
    ytd=YouTube(url)
    ytd=YouTube(url).streams.first().download()
    os.rename(ytd,ytd.replace(" ",""))
    global video_filename
    video_filename=ntpath.basename(ytd)
    call_main(video_filename)

def translator():
    input1=from_choice.get()
    input2=to_choice.get()
    source_lang=source(input1)
    target_lang=target(input2)
    print(source_lang)
    print(target_lang)
```

```
    output_filename=video_filename[:length-4].replace(" ","")+"(with-"+input2+"-
subtitle).mp4"
    audio_filename=video_filename[:length-4].replace(" ","")+".flac"
    bucket=storage_client.get_bucket(buck_name)
    blob=bucket.blob(audio_filename)
    blob.upload_from_filename(audio_filename)

subtitle_gen("gs://"+buck_name+"/"+audio_filename,source_lang,target_lang,video_file
name.replace(" ",""),output_filename)
```

#Initializing Tkinter Window
```
root = Tk()
root.title("Automatic Subtitle Encoder")
#encoder_icon = PhotoImage(file = "languages.png")
#root.iconphoto(True, encoder_icon)
root.resizable(0,0)
bg_image = ImageTk.PhotoImage(Image.open("maxresdefaultfinal.jpg"))
```

#Creating Canvas Object
```
canvas = Canvas(root, width=bg_image.width(), height=bg_image.height())
canvas.create_image(0,0,anchor='nw',image=bg_image)
canvas.pack()
```

#Configuring Style for GUI Components
```
fontstyle = ("Arial Rounded MT Bold",10,"bold")
style = ttk.Style()
style.theme_create('combostyle', parent='alt', settings={'TCombobox':
                                {'configure':
                                 {'fieldbackground': '#d3e0ea',
                                  'background': '#276678',
                                  'highlightbackground' : '#276678',
                                  }}})
style.theme_use('combostyle')
```

#Text Label for required Fields
```
canvas.create_text(400,100, text="AUTOMATIC SUBTITLE", font=("Typo Draft
Demo", 35),fill="#9fd8df")
canvas.create_text(410,150, text="ENCODER", font=("Typo Draft Demo", 35),fill =
"#9fd8df")
canvas.create_text(160,300, text="Path for SRT File : ", font=("Arial Rounded MT
Bold", 18),fill = "#a4ebf3")
canvas.create_text(160,550, text="Target Language : ", font=("Arial Rounded MT
Bold", 18),fill = "#a4ebf3")
canvas.create_text(160,450, text="Source Language : ", font=("Arial Rounded MT
```

Bold", 18),fill = "#a4ebf3")

#Button for PC Dialog
```
dialog_button_obj_pc = Button(canvas, text="PC", command=pc_click, width=12,
bg="#9fd8df", font=fontstyle)
dialog_pc = canvas.create_window(350,350,window=dialog_button_obj_pc)
```

#Button for Youtube Reference
```
dialog_button_obj_youtube = Button(canvas, text="YOUTUBE", command=u_click,
width=12, bg="#9fd8df", font=fontstyle)
dialog_youtube = canvas.create_window(500,350,window=dialog_button_obj_youtube)
```

#Creating Path text box for directory
```
path_text_obj = Entry(root,
highlightthickness=4,width=30,highlightbackground="#276678",bg="#d3e0ea",
font=fontstyle)
canvas.create_window(420,300,window=path_text_obj)
```

#Button for Translation
```
convert_button = Button(canvas, text="Translate", command=translator, width=15,
bg="#9fd8df", font=fontstyle)
convert_button = canvas.create_window(380,630,window=convert_button)
```

#ComboBoxes for Language Selection
```
from_choice = ttk.Combobox(canvas,width = 25, font=fontstyle)
from_choice['values'] = ("Afrikaans(South
Africa)","Arabic(Bahrain)","Arabic(Egypt)","Arabic(Iraq)","Arabic(Israel)","Arabic(Jor
dan)","Arabic(Kuwait)","Arabic(Lebanon)","Arabic(Oman)",
"Arabic(Qatar)","Arabic(Saudi Arabia)","Arabic(State of Palestine)","Arabic(United
Arab Emirates)","Bengali(Bangladesh)","Chinese,Cantonese(Traditional Hong Kong)",
"Chinese,Mandarin(Traditional,Taiwan)","Czech(Czech
Republic)","Danish(Denmark)","Dutch(Netherlands)","English(Australia)","English(Gh
ana)","English(India)","English(Nigeria)",
"English(Philippines)","English(Singapore)","English(South
Africa)","English(Tanzania)","English(United Kingdom)","English(United
States)","Filipino(Philippines)","Finnish(Finland)",
"French(Canada)","French(France)","German(Germany)","Gujarati(India)","Hebrew(Isr
ael)","Hindi(India)","Indonesian(Indonesia)","Italian(Italy)","Japanese(Japan)","Kannad
a(India)",
"Korean(South
Korea)","Malay(Malaysia)","Malayalam(India)","Marathi(India)","Norwegian
Bokmål(Norway)","Persian(Iran)","Polish(Poland)","Portuguese(Brazil)","Portuguese(P
ortugal)",
"Russian(Russia)","Serbian(Serbia)","Spanish(Spain)","Spanish(United
States)","Swedish(Sweden)","Telugu(India)","Thai(Thailand)","Turkish(Turkey)","Ukra
```

```
inian(Ukraine)","Urdu(Pakistan)"
,"Vietnamese(Vietnam)","Zulu(South Africa)")
from_choice['state'] = 'readonly'
canvas.create_window(400,450,window=from_choice,height=20)

to_choice = ttk.Combobox(canvas,width = 25, font=fontstyle)
to_choice['values'] =
("Afrikaans","Albanian","Amharic","Arabic","Armenian","Azerbaijani","Basque","Bela
rusian","Bengali","Bosnian","Bulgarian","Catalan","Cebuano","Chinese(Simplified)",
"Chinese(Traditional)","Corsican","Croatian","Czech","Danish","Dutch","English","Esp
eranto","Estonian","Finnish","French","Frisian","Galician","Georgian","German","Gree
k","Gujarati",
"Haitian
Creole","Hausa","Hawaiian","Hebrew","Hindi","Hmong","Hungarian","Icelandic","Igb
o","Indonesian","Irish","Italian","Japanese","Javanese","Kannada","Kazakh","Khmer","
Kinyarwanda",
"Korean","Kurdish","Kyrgyz","Lao","Latin","Latvian","Lithuanian","Luxembourgish","
Macedonian","Malagasy","Malay","Malayalam","Maltese","Maori","Marathi","Mongoli
an","Myanmar(Burmese)",
"Nepali","Norwegian","Nyanja(Chichewa)","Odia(Oriya)","Pashto","Persian","Polish","
Portuguese(Portugal,Brazil)","Punjabi","Romanian","Russian","Samoan","Scots
Gaelic","Serbian",
"Sesotho","Shona","Sindhi","Sinhala(Sinhalese)","Slovak","Slovenian","Somali","Spani
sh","Sundanese","Swahili","Swedish","Tagalog(Filipino)","Tajik","Tamil","Tatar","Tel
ugu","Thai",
"Turkish","Turkmen","Ukrainian","Urdu","Uyghur","Uzbek","Vietnamese","Welsh","X
hosa","Yiddish","Yoruba","Zulu")
to_choice['state'] = 'readonly'
canvas.create_window(400,550,window=to_choice)
```

#Updating Canvas
```
canvas.update()
root.mainloop()
```

REFERENCES

1. A. Mathur, T. Saxena and R. Krishnamurthi, "IEEE International Conference on Computational Intelligence & Communication Technology", DOI : 10.1109/CICT.2015.46, 2019.

2. A. Ramani, A. Rao, V. Vidya and V. B. Prasad, "Automatic Subtitle Generation for Videos", the International Conference on Advanced Computing and Communication Systems (ICACCS), 2020.

3. C. Duan et al, "Modeling Future Cost for Neural Machine Translation", IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, DOI: 10.1109/TASLP.2020.3042006, 2019.

4. D. Xiong, M. Zhang and H. Lim, "A Maximum-Entropy Segmentation Model for Statistical Machine Translation", IEEE Transactions on Audio, Speech, and Language Processing, volume-19, DOI : 10.1109/TASL.2011.2144971, 2019.

5. H. Li, J. Sha and C. Shi, "Revisiting Back-Translation for Low-Resource Machine Translation Between Chinese and Vietnamese," IEEE Access, volume-8, DOI : 10.1109/ACCESS.2020.3006129, 2020.

6. H. Arif, H. Hajjdiab, F. A. Harbi and M. Ghazal, "A Comparison between Google Cloud Service and iCloud," IEEE 4th International Conference on Computer and Communication Systems (ICCCS), DOI: 10.1109/CCOMS.2019.8821744, 2019.

7. H. Sun, R. Wang, K. Chen, M. Utiyama et al, "Unsupervised Neural Machine Translation With Cross-Lingual Language Representation Agreement", IEEE/ACM Transactions on Audio, Speech, and Language Processing, Volume-28, DOI : 10.1109/TASLP.2020.2982282, 2020.

8. James M. Keller, Derong Liu, David B. Fogel, "Recurrent Neural Networks," in Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation, IEEE, 2016.

9. K. Chen et al., "A Neural Approach to Source Dependence Based Context Model for Statistical Machine Translation," IEEE/ACM Transactions on Audio, Speech, and Language Processing, Volume-26, DOI : 10.1109/TASLP.2017.2772846, 2018.

10. K. Saengthongpattana, K.Kriengket, P. Porkaew et al, "Thai-English and English-Thai Translation Performance of Transformer Machine Translation", International Joint Symposium on Artificial Intelligence and Natural Language Processing , DOI : 10.1109/iSAI-NLP48611.2019.9045174, 2019.

11. Kehai Chen, Rui Wang, Masao Utiyama et al, "Towards more diverse input representation for Neural Machine Translation", IEEE/ACM Transactions on Audio, Speech, and Language Processing, Volume-28, 2020.

12. K. M. Chaman Kumar, S. Aswale, P. Shetgaonkar, V. Pawar et al, "A Survey of Machine Translation Approaches for Konkani to English", International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020.

13. Lucia Specia, Carolina Scarton, Gustavo Henrique Paetzold, Graeme Hirst, "Quality Estimation for Machine Translation", Morgan & Claypool Publications, 2018.

14. M. Siahbani, R. M. Seraj, B. Sankaran and A. Sarkar, "Incremental translation using hierarchichal phrase-based translation system," IEEE Spoken Language Technology Workshop (SLT), DOI: 10.1109/SLT.2014.7078552, 2019.

15. Ozan, "Increasing system performance in machine learning by using multiprocessing," 26th Signal Processing and Communications Applications Conference (SIU), 2018, DOI : 10.1109/SIU.2018.8404280.

16. Q. Xiao, X. Chang, X. Zhang and X. Liu, "Multi-Information Spatial–Temporal LSTM Fusion Continuous Sign Language Neural Machine Translation," IEEE Access, Volume 8, DOI: 10.1109/ACCESS.2020.3039539, 2020.

17. R. D. Lero, C. Exton and A. Le Gear, "Communications using a speech-to-text-to-speech pipeline" International Conference on Wireless and Mobile Computing, Networking and Communications (WiMOB), DOI : 10.1109/WiMOB.2019.8923157, 2019.

18. S. Satpathy, S. P. Mishra and A. K. Nayak, "Analysis of Learning Approaches for Machine Translation Systems", International Conference on Applied Machine

Learning (ICAML), DOI : 10.1109/ICAML48257.2019.00038, 2019.

19. Stephen I. Gallant, "Neural Network Expert Systems," in Neural Network Learning and Expert Systems, MIT Press, 1993.

20. S. Tyagi, D. Chopra, I. Mathur and N. Joshi, "Classifier based text simplification for improved machine translation", International Conference on Advances in Computer Engineering and Applications, DOI : 10.1109/ICACEA.2015.7164711, 2020.

21. X. Wang, Z. Tu and M. Zhang, "Incorporating Statistical Machine Translation Word Knowledge Into Neural Machine Translation",IEEE/ACM Transactions on Audio, Speech, and Language Processing, volume-26,DOI : 10.1109/TASLP.2018.2860287, 2019.

22. X. Liu, D. F. Wong, L. S. Chao and Y. Liu, "Latent Attribute Based Hierarchical Decoder for Neural Machine Translation", IEEE/ACM Transactions on Audio, Speech, and Language Processing, volume - 27, 2019.