**1.INTRODUCTION**

Speech-to-speech translation is a technology that automatically translates one language into another in order to enable communication between two persons whose mother tongues are different. In other words, speech translation is one of the process in which spoken words of sentences are immediately translated and spoken aloud in second language. From speech-to-speech technology it is possible to build the system that translates continuously spoken words from one language to another.

Speech translation technology is selected as one of the ten technologies that will change the world. Recent technological advances and breakthroughs leading to the constant increase in computational power with an unprecedented tendency of building smaller and smaller devices that yield higher and higher performance, has inevitably lead to a strong demand for information retrieving and communication enabling technologies that are multilingual aware.

Speech-to-speech (S2S) translation is a complex process designed to assist communication between individuals that speak different languages. Speech systems find wide range applications in education, medicine, military, marketing and cross-border operations etc. Language of speech is an important factor to be dealt within order to complete an effective communication link. Over the past few decades, the need to overcome this language barrier between people belonging to different linguistic backgrounds has been an area of interest in research. Language translation systems have served as a major breakthrough for this issue.

In police interrogations in order to investigating other language people for case or basic information getting we need to use another person to translate everything it is hard make the documentation the translated text we need manual typing. Our goal is to realize an automatic speech-to-speech translator for police enquires and easy reporting.

**2. LITERATURE SURVEY**

Quoc Truong Do, Sakriani Sakti, et al., in 2018 proposed a work on emphasis speech translation. Emphasis is used to distinguish between focused and unfocused part of an utterance and it is useful in misheard situations in which speakers must repeat the most important words or phrases.The proposed approach to handle continuous emphasis levels which is based on sequence models and also they have combined the machine and emphasis translation into the single model. They have proposed hard attention emphasis speech translation and joint model. In the hard attention emphasis translation that can translate continuous emphasis weights without quantization. And in the joint model which simplifies the translation and jointly translates words and emphasis with one-word delay. They have used LSTM-based encoder-decoder to solve the problems in single model of hard attention sequence-to-sequence model.

Hariz Zakka Muhammad, Muhammad Nasrun, et al., in 2018 proposed a work on speech recognition by using hidden markov model. They consider the translation language from English to Indonesian. The classification method used is the

Mel frequency Cepstral coefficients (MFCC) and Hidden markov model (HMM). The proposed system converts speech-to-text and uses the existing Google translation or

Microsoft translation for translation. The speech signal will be processed by using MFCC. The algorithm used here is kmeans algorithm.

J Poornakala, A Maheshwari.,in 2016 proposed a work speech-speech translation from English to Tamil language. Speech recognition system recognizing English speech via a voice recognition device. English speech translate the text into English using the speech synthesis, after converting it to text, compare with the words stored in the database if it matches the Tamil text stored in the database. The English text is converted to Tamil text is made by the machine translation system, and then the text will be displayed on the screen. It is designed to translate up 60 words in the Android application. The algorithm used here is HMM algorithm.

Sangmi Shin, Eric T.Matson, Jinok Park, et al., in 2015 proposed a work on speech-to-speech translation humanoid robot. The proposed system is helpful for English speaking patients they can explain their problems to Korean doctors. It

can solve the problems of many patients who don’t know the Korean language although they can describe their problems to doctor which replace the role of human workers. It uses CMU Sphinx-4 tool for speech recognition. English-Korean translation is based on rulebased translation.

Seung Yun, Young-Jik Lee, et al., in 2014 proposed a work on ultilingual Speech-to-Speech Translation System for Mobile Consumer Devices. In this paper they established a massive language speech database closest to the environment

where the speech-to-speech translation device is actually used after mobilize many people based on the user survey requests. It was possible to secure excellent basic

performance in the environment similar to speech-to-speech translation environment rather than just under the experimental environment. Moreover, with the speech-to speech translation interface, a user-friendly interface has been designed and at the same time the errors were reduced during the translation process so many steps to improve the user satisfaction were employed.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

In general police investigations are done physically If police interrogations is to investigating other language person for case or for information gaining we need to use another person to translate everything and it is hard make the documentation of the translated text we need to type manually. It is easy to translate local neighbour languages because of local neighbour language translating persons are available but for nonlocal languages it is very tough to translate.

**Disadvantages:**

* Difficult to translate some other country languages.
* Time taking process to make documents of investigation.

**3.2 Proposed System**

We are designing a model which converts the speech-to-speech translation. It means the system automatically recognise the speech and converts to required language produces the speech through artificial intelligence. Through this system any language will convert into any language and easy to make documents with translated speech. It helps to police for enquires and information gaining.it is easy to handle.

**Advantages:**

* Easy to translate one language to another any language.
* Easy to prepare documents of investigation with multiple languages.
* Easy to record audio’s and storing.

**3.3 System Requirements**

**3.3.1 Software Requirements**

Operating System : Windows

Programming Languages : Python

**3.3.2 Hardware Requirements**

Processor : i3 or above

RAM : 4GB (minimum)

Hard Disk : 500GB (minimum)

Monitor : 14’ color monitor

Keyboard and mouse.

Microphone

**4. TECHNOLOGY OVERVIEW**

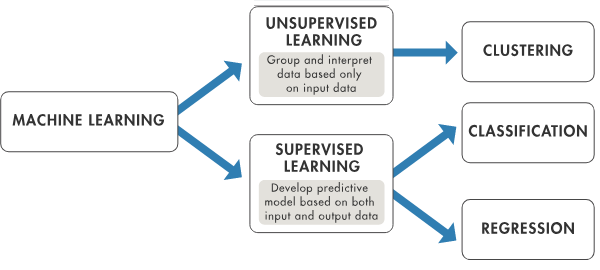
**4.1 MACHINE LEARNING**

**4.1.1 Introduction**

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

In this project, we’ll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning. We’ll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we’ll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

 **Fig. 4.1 Machine Learning Methods**

**4.1.2 Machine Learning Methods**

Two of the most widely adopted machine learning methods is **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let’s explore these methods in more detail.

1. **Supervised Learning**

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to “learn” by comparing its actual output with the “taught” outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

1. **Unsupervised Learning**

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together

**4.2 NMT (Neural Machine Translation)**

Machine Translation (MT) is a subfield of computational linguistics that is focused on translating text from one language to another. With the power of deep learning, Neural Machine Translation (NMT) has arisen as the most powerful algorithm to perform this task. While Google Translate is the leading industry example of NMT, tech companies all over the globe are going all in one of NMT. This state-of-the-art algorithm is an application of deep learning in which massive datasets of translated sentences are used to train a model capable of translating between any two languages. With the vast amount of research in recent years, there are several variations of NMT currently being investigated and deployed in the industry. One of the older and more established versions of NMT is the Encoder Decoder structure. This architecture is composed of two recurrent neural networks (RNNs) used together in tandem to create a translation model. And when coupled with the power attention mechanisms, this architecture can achieve impressive results.

# 4.2.1Brief Explanation of NMT and the Encoder Decoder Structure

The ultimate goal of any NMT model is to take a sentence in one language as input and return that sentence translated into a different language as output. The figure below is a naive representation of a translation algorithm (such as Google Translate) tasked with translating from English to Spanish.

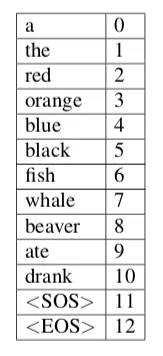


**Fig. 4.2.1 Translation from English to Spanish**

Before diving into the Encoder Decoder structure that is oftentimes used as the algorithm in the above figure, we first must understand how we overcome a large hurdle in any machine translation task. Namely, we need a way to transform sentences into a data format that can be inputted into a machine learning model. In essence, we must somehow convert our textual data into a numeric form.

To do this in machine translation, each word is transformed into a One Hot Encoding vector which can then be inputted into the model. A One Hot Encoding vector is simply a vector with a 0 at every index except for a 1 at a single index corresponding to that particular word. In this way, each word has a distinct One Hot Encoding vector and thus we can represent every word in our dataset with a numerical representation.

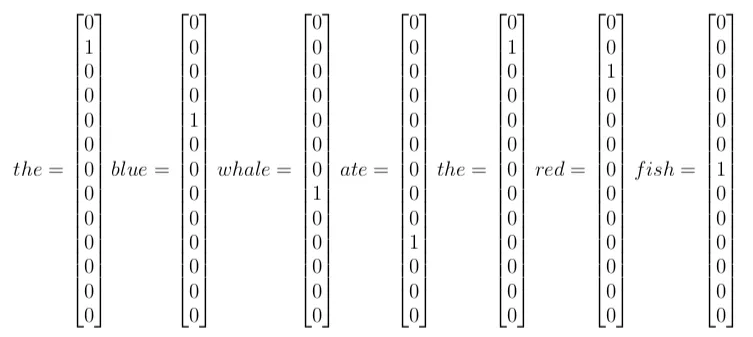
The first step towards creating these vectors is to assign an index to each unique word in the input language, and then repeat this process for the output language. In assigning a unique index to each unique word, we will be creating what is referred to as a Vocabulary for each language. Ideally, the Vocabulary for each language would simply contain every unique word in that language. However, given that any single language can have hundreds of thousands of words, the vocabulary is often trimmed to the N most common words in the dataset we are working with (where N is chosen arbitrarily, but often ranges from 1,000–100,000 depending on the dataset size).To understand how we can then use a Vocabulary to create One Hot Encoding vectors for every word in our dataset, consider a mini-Vocabulary containing just the words in Table below.

****

**Fig. 4.2.2 Mini-vocabulary for the English language**

Given this table, we have assigned a unique index 0–12 to every word in our mini-Vocabulary. The <SOS> and <EOS> tokens in the table are added to every Vocabulary and stand for START OF SENTENCE and END OF SENTENCE respectively. They are used by the NMT model to help identify these crucial points in sentences.

Now, let’s say we want to convert the words in the sentence “the blue whale ate the red fish” to their one hot encoding vectors. Using Table 1, we would do this as shown in Figure below

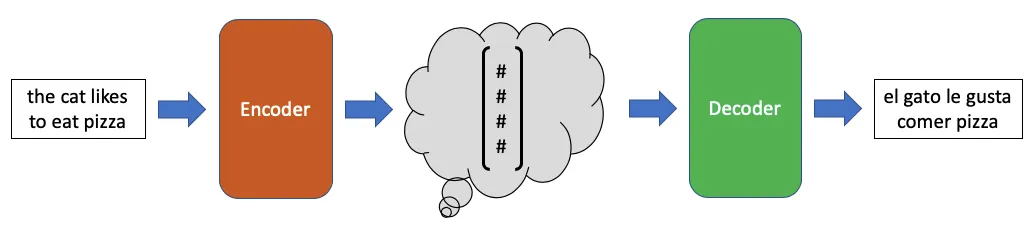


**Fig. 4.2.3 One Hot Encoding vectors for the sentence**

As you can see above, each word becomes a vector of length 13 (which is the size of our vocabulary) and consists entirely of 0s except for a 1 at the index that was assigned to that word in Table 1.

By creating a vocabulary for both the input and output languages, we can perform this technique on every sentence in each language to completely transform any corpus of translated sentences into a format suitable for the task of machine translation.

Now, with an understanding of how we can represent textual data in a numeric way, let’s look at the magic behind this Encoder Decoder algorithm. At the most basic level, the Encoder portion of the model takes a sentence in the input language and creates a thought vector from this sentence. This thought vector stores the meaning of the sentence and is subsequently passed to a Decoder which outputs the translation of the sentence in the output language. This process is shown in the figure below.



**Fig. 4.2.4 Encoder Decoder structure translating the English sentence**

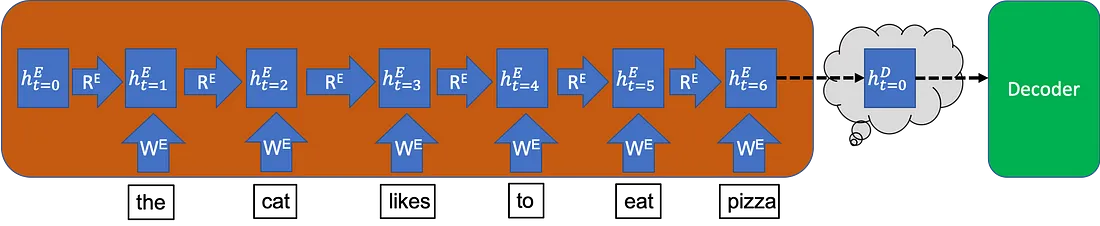
In the above architecture, the Encoder and the Decoder are both recurrent neural networks (RNN). In this particular tutorial, we will be using Long Short-Term Memory (LSTM) models, which are a type of RNN. However other RNN architectures, such as a GRU, are often used. At a basic level, RNNs are neural networks designed specifically to deal with temporal/textual data. This article will give a high-level overview of how RNNs work in the context of NMT, however, I would strongly recommend looking further into these concepts if you are not already familiar with them.

In the case of the Encoder, each word in the input sentence is fed separately into the model in a number of consecutive time-steps. At each time-step, t, the model updates a hidden vector, h, using information from the word inputted to the model at that time-step. This hidden vector works to store information about the inputted sentence. In this way, since no words have yet been inputted to the Encoder at time-step t=0, the hidden state in the Encoder starts out as an empty vector at this time-step. We represent this hidden state with the blue box in Figure 4, where the subscript t=0 indicates the time-step and the superscript E corresponds to the fact that it’s a hidden state of the Encoder (rather than a D for the Decoder).

****

**Fig. 4.2.5 Encoder hidden vector at t=0**

At each time-step, this hidden vector takes in information from the inputted word at that time-step, while preserving the information it has already stored from previous time-steps. Thus, at the final time-step, the meaning of the whole input sentence is stored in the hidden vector. This hidden vector at the final time-step is the thought vector referred to above, which is then inputted into the Decoder. The process of encoding the English sentence “the cat likes to eat pizza” is represented in Figure below

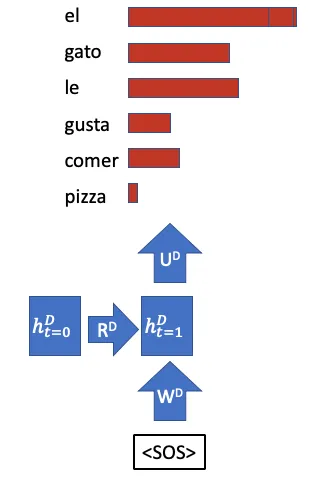
.

**Fig. 4.2.6 Encoding of the sentence**

In the above figure, the blue arrows correspond to weight matrices, which we will work to enhance through training to achieve more accurate translations.

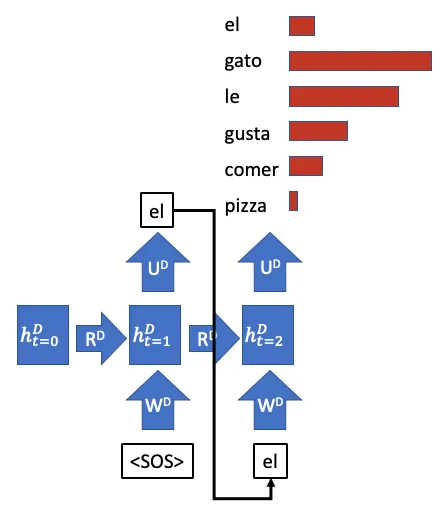
Also, notice how the final hidden state of the Encoder becomes the thought vector and is relabeled with superscript D at t=0. This is because this final hidden vector of the Encoder becomes the initial hidden vector of the Decoder. In this way, we are passing the encoded meaning of the sentence to the Decoder to be translated to a sentence in the output language. However, unlike the Encoder, we need the Decoder to output a translated sentence of variable length. Thus, we are going to have our Decoder output a prediction word at each time-step until we have outputted a complete sentence.

In order to start this translation, we are going to input a <SOS> tag as the input at the first time-step in the Decoder. Just as in the Encoder, the Decoder will use the <SOS> input at time-step t=1 to update its hidden state. However, rather than just proceeding to the next time-step, the Decoder will use an additional weight matrix to create a probability over all of the words in the output vocabulary. In this way, the word with the highest probability in the output vocabulary will become the first word in the predicted output sentence. This first step of the Decoder, translating from “the cat likes to eat pizza” to “el gato le gusta comer pizza” is shown in Figure 6. For the sake of simplicity, the output vocabulary is restricted to the words in the output sentence (but in practice would consist of the thousands of words in the entire output vocabulary).

****

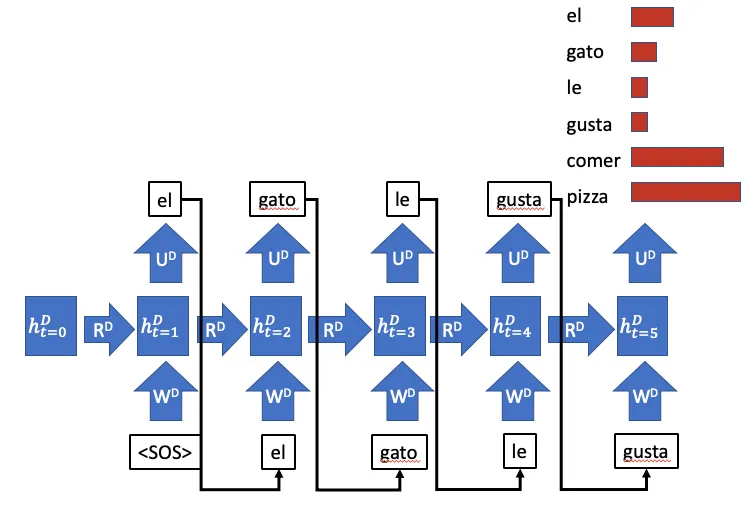
**Fig 4.2.7 First step of the Decoder**

Now, given that the word “el” was given the highest probability, this word becomes the first word in our outputted prediction sentence. And we proceed by using “el” as the input in the next time-step as in Figure below



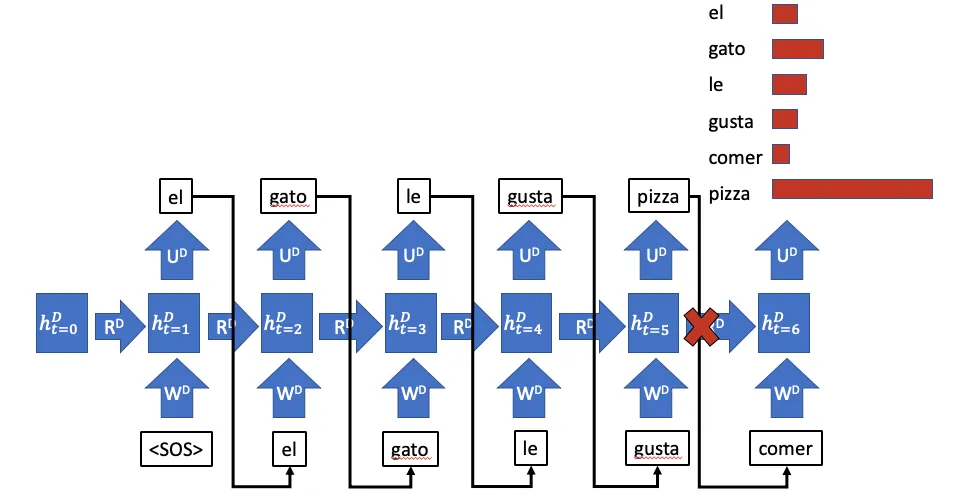
**Fig. 4.2.8 Second step of the Decoder**

We proceed in this way through the duration of the sentence — that is until we run into an error such as that depicted below in Figure below.

****

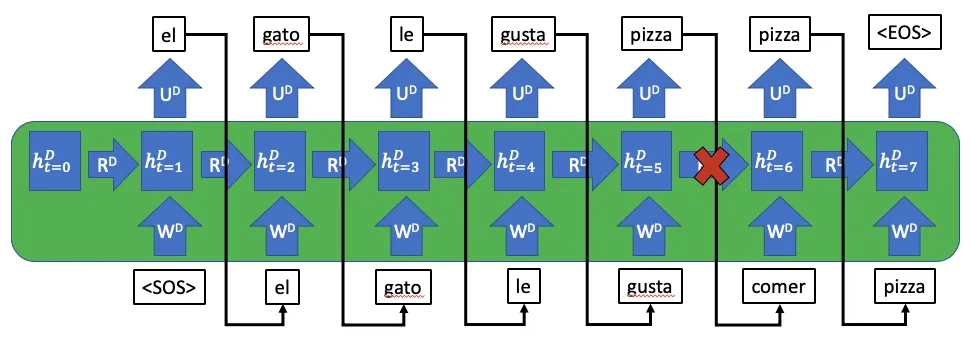
**Fig. 4.2.9 Translation error in Decoder**

As you can see, the Decoder has predicted “pizza” to be the next word in the translated sentence, when it should actually be “comer”. When testing the model on the test set, we would do nothing to correct this error and would allow the Decoder to use this improper prediction as the input at the next time-step. However, during the training process, we are going to keep “pizza” as the predicted word at that point in the sentence, but force our Decoder to input the correct word “comer” as the input for the next time-step. This is a strategy referred to as teacher forcing and helps speed up the training process. It is shown in the below figure.

****

**Fig. 4.2.10 Teacher-Forcing**

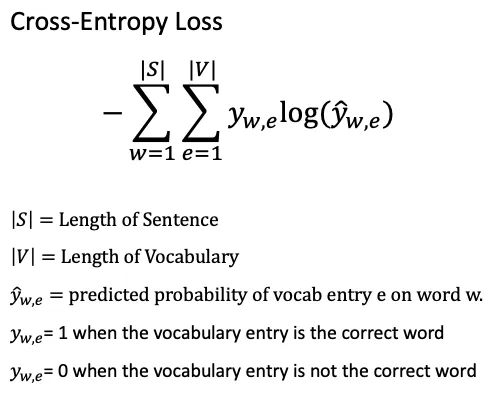
Now, since the Decoder has to output prediction sentences of variable lengths, the Decoder will continue predicting words in this fashion until it predicts the next word in is complete and we are left with a complete predicted translation of the input sentence. The entire process of decoding the thought vector for the input sentence “the cat likes to eat pizza” is shown in Figure



**Fig.4.2.11 Decoding of the sentence**

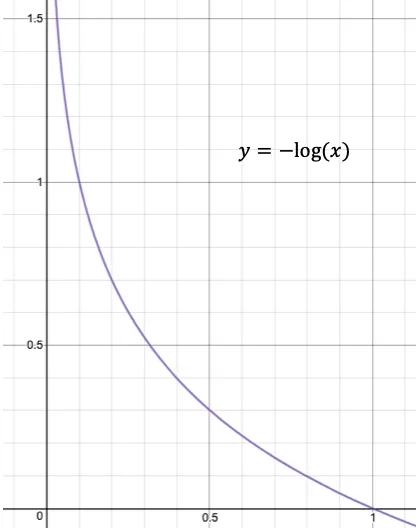
We can then compare the accuracy of this predicted translation to the actual translation of the input sentence to compute a loss. While there are several varieties of loss functions, a very common one to utilize is the Cross-Entropy Loss. The equation of this loss function is detailed in Figure below

We can then compare the accuracy of this predicted translation to the actual translation of the input sentence to compute a loss. While there are several varieties of loss functions, a very common one to utilize is the Cross-Entropy Loss. The equation of this loss function is detailed in Figure

****

**Fig. 4.2.12 Cross-Entropy Loss function**

In essence, what this loss function does is sum over the negative log likelihoods that the model gives to the correct word at each position in the output sentence. Given that the negative log function has a value of 0 when the input is 1 and increases exponentially as the input approaches 0 (as shown in Figure below), the closer the probability that the model gives to the correct word at each point in the sentence is to 100%, the lower the loss.

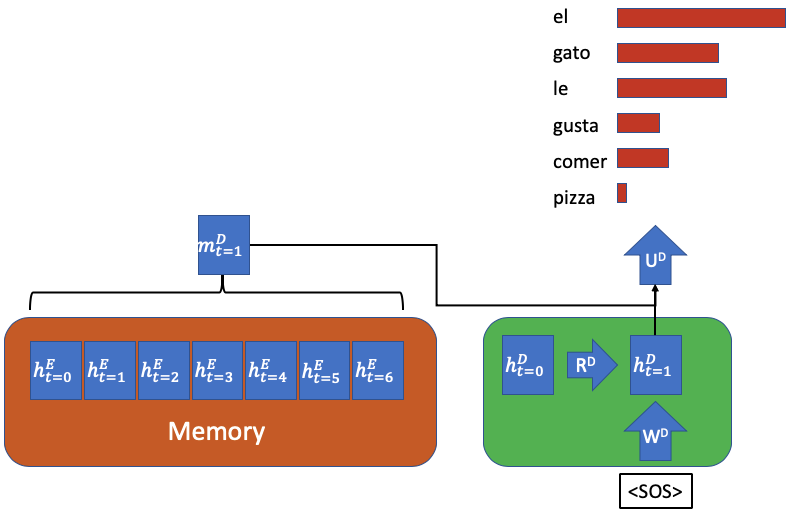


**Fig. 4.2.13 Graph of the function y = -log(x)**

For example, given that the correct first word in the output sentence above is “el”, and our model gave a fairly high probability to the word “el” at that position, the loss for this position would be fairly low. Conversely, since the correct word at time-step t=5 is “comer”, but our model gave a rather low probability to the word “comer”, the loss at that step would be relatively high.

By summing over the loss for each word in the output sentence a total loss for the sentence is obtained. This loss corresponds to the accuracy of the translation, with lower loss values corresponding to better translations. When training, the loss values of several sentences in a batch would be summed together, resulting in a total batch loss. This batch loss would then be used to perform mini-batch gradient descent to update all of the weight matrices in both the Decoder and the Encoder. These updates modify the weight matrices to slightly enhance the accuracy of the model’s translations. Thus, by performing this process iteratively, we eventually construct weight matrices that are capable of creating quality translations.

As mentioned in the introduction, an attention mechanism is an incredible tool that greatly enhances an NMT model’s ability to create accurate translations. In this method of attention, at each time-step, the Decoder “looks back” at all of the hidden vectors of the Encoder to create a memory vector. It then uses this memory vector, along with the hidden vector in the Decoder at that time-step, to predict the next word in the translated sentence. In doing this, the Decoder utilizes valuable information from the Encoder that would otherwise go to waste. A visual representation of this process is shown in Figure below. I’d recommend reading the linked article in this paragraph to learn more about the various ways this memory vector can be calculated to gain a better understanding of this important concept.



**Fig. 4.2.14 Attention mechanism for time-step t=1 in Decoder**

**4.3 Recurrent Neural Network (RNN)**

A recurrent neural network (**RNN**) is a kind of artificial neural network mainly used in **speech recognition** and **natural language processing** (NLP). RNN is used in deep learning and in the development of models that imitate the activity of neurons in the human **brain**.

Recurrent Networks are designed to **recognize patterns** in sequences of data, such as **text, genomes, handwriting, the spoken word,** and **numerical** time series data emanating from sensors, stock markets, and government agencies.

A recurrent neural network looks similar to a traditional neural network except that a memory-state is added to the neurons. The computation is to include a simple memory.

The recurrent neural network is a type of deep learning-oriented algorithm, which follows a sequential approach. In neural networks, we always assume that each input and output is dependent on all other layers. These types of neural networks are called recurrent because they sequentially perform mathematical computations.

**4.3.1. Long short-term memory (LSTM) RNN**

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. It was proposed in 1997 by **Sepp Hochreiter** and **Jurgen schmidhuber**. Unlike standard feed-forward neural networks, LSTM has feedback connections. It can process not only single data points (such as images) but also entire sequences of data (such as speech or video).

**For example,** LSTM is an application to tasks such as unsegmented, **connected handwriting recognition,** or **speech recognition**.

A general **LSTM** unit is composed of a cell, an input gate, an output gate, and a forget gate. The cell remembers values over arbitrary time intervals, and three gates regulate the flow of information into and out of the cell. LSTM is well-suited to classify, process, and predict the time series given of unknown duration.

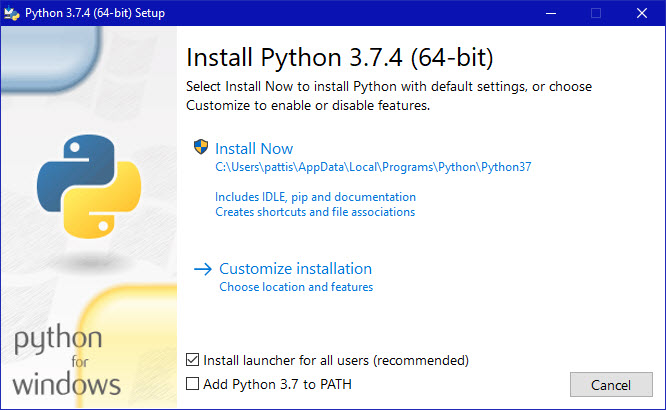
**4.4 PYTHON**

**4.4.1 PYTHON DESCRIPTION**

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, <https://www.python.org/>, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

Python is simple to use, but it is a real programming language, offering much more structure and support for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much more error checking than C, and, being a *very-high-level language*, it has high-level data types built in, such as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much larger problem domain than Awk or 

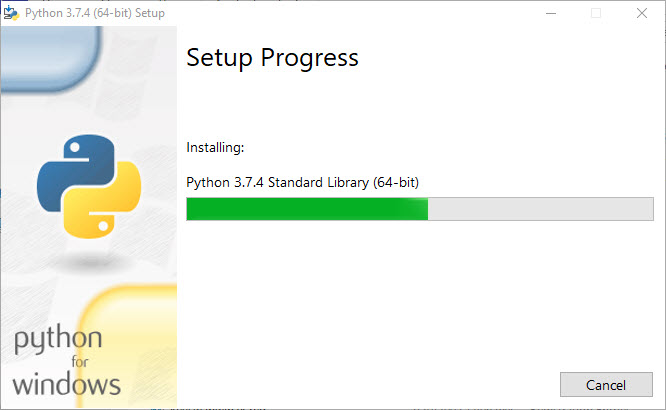
**Fig. 4.4.1(a) Python Installation**

Ensure that the **Install launcher for all users (recommended)** and the **Add Python 3.7 to PATH** checkboxes at the bottom are checked.

If the Python Installer finds an earlier version of Python installed on your computer, the **Install Now** message may instead appear as **Upgrade Now** (and the checkboxes will not appear).

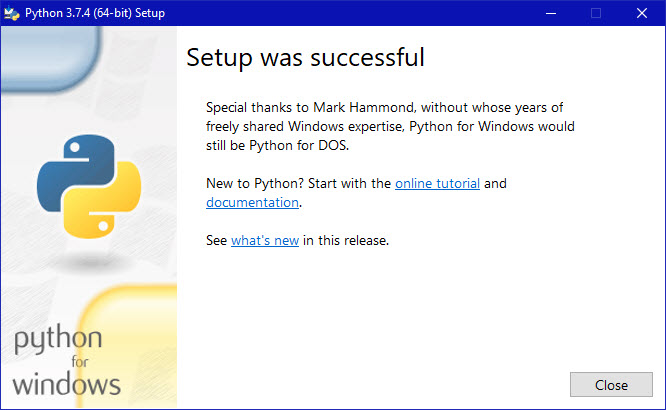
1. Highlight the **Install Now** (or **Upgrade Now**) message, and then click it. When run, a **User Account Control** pop-up window may appear on your screen. I could not capture its image, but it asks, **Do you want to allow this app to make changes to your device**.
2. Click the **Yes** button.

A new **Python 3.7.4 (64-bit) Setup** pop-up window will appear with a **Setup Progress** message and a progress bar.



**Fig. 4.4.1(b) Python Setup process**

During installation, it will show the various components it is installing and move the progress bar towards completion. Soon, a new **Python 3.7.4 (64-bit) Setup** pop-up window will appear with a **Setup was successfully** message.



**Fig. 4.4.1(c) Python Setup Complete**

1. Click the **Close** button.

Python should now be installed.

Python allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throw-away programs, or to test functions during bottom-up program development. It is also a handy desk calculator.

Python enables programs to be written compactly and readably. Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:

* the high-level data types allow you to express complex operations in a single statement;
* statement grouping is done by indentation instead of beginning and ending brackets;
* no variable or argument declarations are necessary.

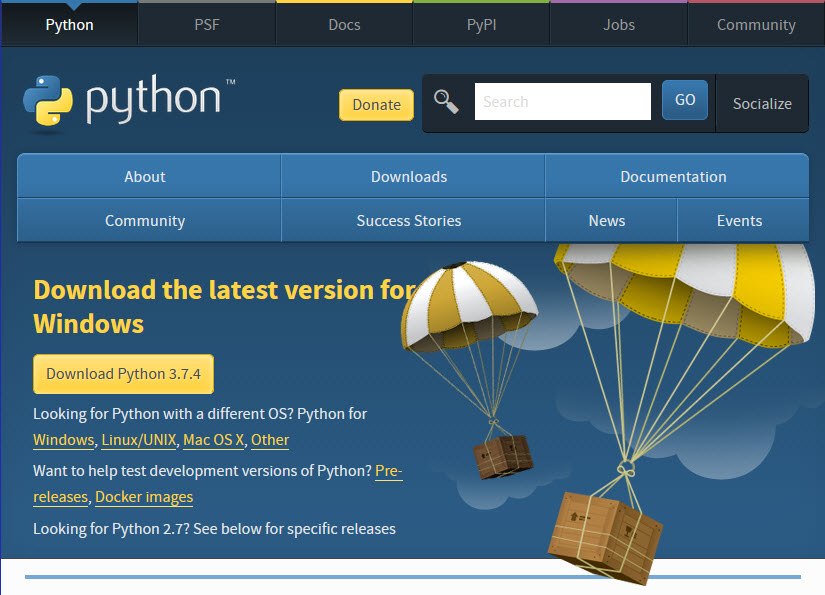
Python is extensible: if you know how to program in C it is easy to add a new built-in function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you are really hooked, you can link the Python interpreter into an application written in C and use it as an extension or command language for that application.

**4.4.2 INSTALL PYTHON ON WINDOWS**

## Python: Version 3.7.4

The Python download requires about 25 Mb of disk space; keep it on your machine, in case you need to re-install Python. When installed, Python requires about an additional 90 Mb of disk space.

1. Click <https://www.python.org/downloads/>
2. The following page will appear in your browser.



**Fig. 4.4.2(a) Python Software Website Home Page**

1. Click the **Windows** link (two lines below the **Download Python 3.7.4** button). The following page will appear in your browser.



**Fig. 4.4.2(b) Python Software Website Download Page**

1. Click on the **Download Windows x86-64 executable installer** link under the top-left **Stable Releases**. Click the **Save File** button.

The file named **python-3.7.4-amd64.exe** should start downloading into your standard download folder. This file is about 30 Mb so it might take a while to download fully if you are on a slow internet connection (it took me about 10 seconds over a cable modem).

The file should appear as https://www.ics.uci.edu/~pattis/common/handouts/pythoneclipsejava/images/python/exefile.jpg

1. Move this file to a more permanent location, so that you can install Python (and reinstall it easily later, if necessary).
2. Feel free to explore this webpage further; if you want to just continue the installation, you can terminate the tab browsing this webpage.
3. Start the **Installing** instructions directly below.

**Installing**

1. Double-click the icon labeling the file **python-3.7.4-amd64.exe**. A **Python 3.7.4 (64-bit) Setup** pop-up window will appear.

## 4.4.3 Speech Recognition

The SpeechRecognition library acts as a wrapper for several popular speech APIs and is thus extremely flexible. One of these—the Google Web Speech API—supports a default API key that is hard-coded into the SpeechRecognition library. That means you can get off your feet without having to sign up for a service.

The flexibility and ease-of-use of the SpeechRecognition package make it an excellent choice for any Python project. However, support for every feature of each API it wraps is not guaranteed. You will need to spend some time researching the available options to find out if SpeechRecognition will work in your particular case.

## So, now that you’re convinced you should try out SpeechRecognition, the next step is getting it installed in your environment

## 

## Installing Speech Recognition

Speech Recognition is compatible with Python 2.6, 2.7 and 3.3+, but requires some [additional installation steps for Python 2](https://github.com/Uberi/speech_recognition#requirements). For this tutorial, I’ll assume you are using Python 3.3+.

You can install SpeechRecognition from a terminal with pip:

$ pip install SpeechRecognition

Once installed, you should verify the installation by opening an interpreter session and typing:

>>>

>>> import speech\_recognition as sr

>>> sr.\_\_version\_\_

'3.8.1'

## The Recognizer Class

All of the magic in SpeechRecognition happens with the Recognizer class.

The primary purpose of a Recognizer instance is, of course, to recognize speech. Each instance comes with a variety of settings and functionality for recognizing speech from an audio source.

Creating a Recognizer instance is easy. In your current interpreter session, just type:

>>> r = sr.Recognizer()

We are using below API for Recognizing speech from an audio source

recognize\_google(): [Google Web Speech API](https://w3c.github.io/speech-api/speechapi.html)

### The Microphone Class

Open up another interpreter session and create an instance of the recognizer class.

>>> import speech\_recognition as sr

>>> r = sr.Recognizer()

Now, instead of using an audio file as the source, you will use the default system microphone. You can access this by creating an instance of the Microphone class.

>>>

>>> mic = sr.Microphone()

### Using listen() to Capture Microphone Input

Now that you’ve got a Microphone instance ready to go, it’s time to capture some input.

just like the AudioFile class, Microphone is a context manager. You can capture input from the microphone using the listen() method of the Recognizer class inside of the with block. To handle ambient noise, you’ll need to use the adjust\_for\_ambient\_noise() method of the Recognizer class, just like you did when trying to make sense of the noisy audio file.

>>> with mic as source:

... r.adjust\_for\_ambient\_noise(source)

... audio = r.listen(source)

...

**4.4.4 google-trans-new**

Machine translation is the automatic translation of a text into a different language using computer software. One of the modules offered by python for this is google-trans-new.

## ****How to install?****

pip install google\_trans\_new

After importing the libraries, we can use as follows:

**from** google\_trans\_new **import** google\_translatortranslator=google\_translator()translated\_text=translator.translate("merhaba dünya", lang\_tgt='en')

Output:



To translate a text from one language to another, you have to import the google\_translator class from google\_trans\_new module. Then you have to create an object of the google\_translator class and finally pass the text as a parameter to the **translate** method and specify the target language by using **lang\_tgt** parameter e.g lang\_tgt=”en”.

**from google\_trans\_new import google\_translator**

**translator = google\_translator()**

**sentence = "Tanzania ni nchi inayoongoza kwa utalii barani afrika"**

**translate\_text = translator.translate(sentence,lang\_tgt='en')**

**print(translate\_text)**

**4.4.5 pyttsx3**

pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application invokes the pyttsx3.init() factory function to get a reference to a pyttsx3. Engine instance. it is a very easy to use tool which converts the entered text into speech. The pyttsx3 module supports two voices first is female and the second is male which is provided by “sapi5” for windows. It supports three TTS engines :

• sapi5 – SAPI5 on Windows

• nsss – NSSpeechSynthesizer on Mac OS X

• espeak – eSpeak on every other platform

Installation pyttsx3 is a text-to-speech conversion library in Python. Unlike To install the pyttsx3 module, first of all, you have to open the terminal and write

**pip install pyttsx3**

# Import the required module for text

# to speech conversion

import pyttsx3

# init function to get an engine instance for the speech synthesis

engine = pyttsx3.init()

# say method on the engine that passing input text to be spoken

engine.say('Hello sir, how may I help you, sir.')

# run and wait method, it processes the voice commands.

engine.runAndWait()

**output:**

**'Hello sir, how may I help you, sir.'**

**gtts:**

gTTS (Google Text-to-Speech), a Python library and CLI tool to interface with Google Translate's text-to-speech API. Write spoken mp3 data to a file, a file-like object (bytestring) for further audio manipulation, or stdout.

Customizable speech-specific sentence tokenizer that allows for unlimited lengths of text to be read, all while keeping proper intonation, abbreviations, decimals and more;

Customizable text pre-processors which can, for example, provide pronunciation corrections;

Installation

$ pip install gTTS

Example:

>>> from gtts import gTTS

>>> tts = gTTS('hello')

>>> tts.save('hello.mp3')

**5. FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

1. Operational Feasibility
2. Technical Feasibility
3. Economic Feasibility

**5.1 Operational Feasibility**

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

In traditional approach, the time required to process the find the parking space is taking very much time. So by using this project, we can reduce parking free slots checking time. It will take less time to get parking free slots.

**5.2 Technical Feasibility**

Technical feasibility is one of the first studies that must be conducted after the project has been identified. In large engineering projects consulting agencies that have large staffs of engineers and technicians conduct technical studies dealing with the projects.

In traditional approach the technician find out parking space by his/him own. This may take long time to find parking space. Here the parking space will be displayed on digital screen. So, it is technically feasible.

**5.3 Economical Feasibility**

According to the analysis of this project, it’s the simple project because it implements in linear sequential model. The budget for this project is less hardware and software requirements.

**6. SYSTEM DESIGN**

**6.1 PROPOSED ARCHITECTURE**

The total design consists of these phases:

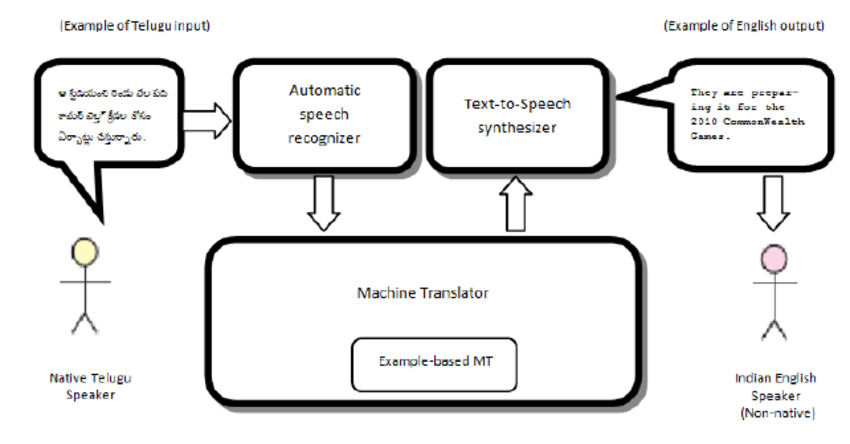
1) Collection of data which is in speech format.

2) Analyse the voice and convert it to text.

3) storing the data and processing it.

4) Speech generation from the text output that is processed.

5) Making the document which is translated directly



**Fig. 6.1 Proposed Architecture**

The data that is collected in the speech form is stored and used as input for next phase of the process. In next phase, the input which is given in the form of voice is processed continuously and is converted into text by using STT. In third phase, the text which is converted, is analysed by Python Script which processes it and identifies the action to be taken for the command. In the last phase, after the action to be taken is identified, output will be obtained from text to speech conversion using TTS.

Speech translation requires the integration of three software technologies.

**1) Automatic Speech Recognition:**It will help in converting the spoken words & phrases into the text in the same language.

**2) Machine Translation:**It will help in converting the text into a second language. It will replace each word in the text with the appropriate word in the second language.

**3) Speech Synthesis:**It will estimate the pronunciation of the text generated by machine translation and generate the speech in the second and desired language.

**6.2 UML Diagrams**

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software system, as well as for business modelling and other non-software systems.

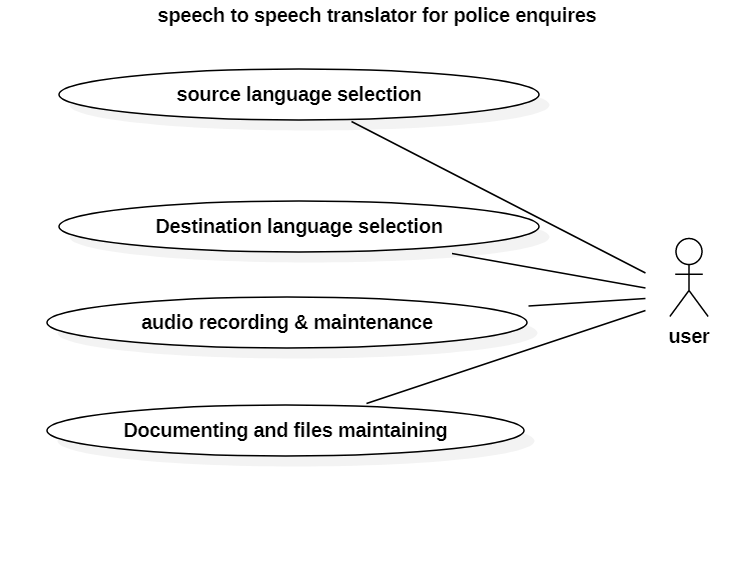
The UML diagrams are made for business users, developers, ordinary people, or anyone who is looking forward to understand the system, such that the system can be software or non-software.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**6.2.1 USECASE DIAGRAM:**

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors**.**

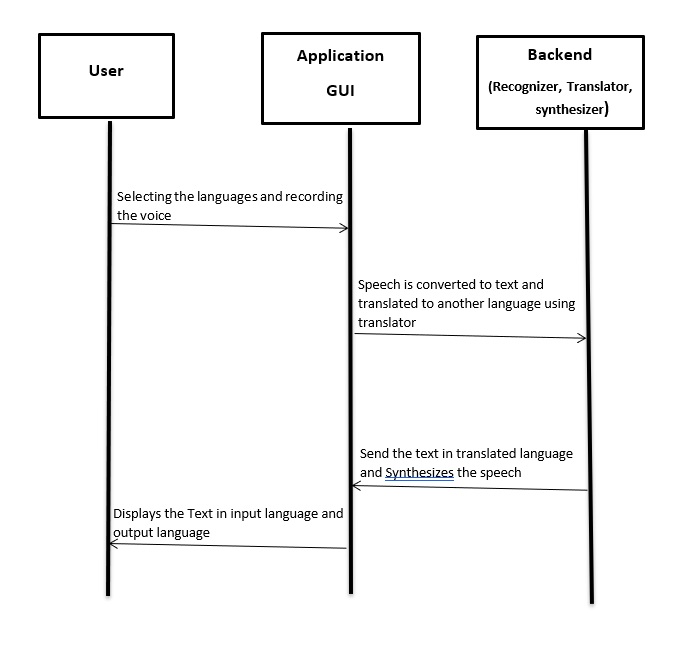


**Fig 6.2 (a) Usecase Diagram**

In the above usecase diagram user is connected to all use cases. Use can select the source language and destination language. He can record and translate into any language provided and can store the audios automatically. He can copy the text and easy make the documents.

**6.2.2 SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



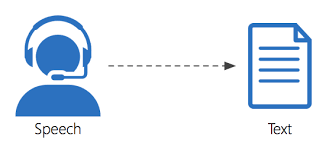
**Fig 6.2 (b) Sequence Diagram**

In the above sequence diagram the source and destination languages are selected is recorded and sends to the backend it will covert speech to text and then translator translates the text into destination language text, then it produces the speech in destination language. Text in Source language and destination language is displays in the text box.

**7. IMPLEMENTATION**

This “SPEECH TO SPEECH TRANSLATOR FOR POLICE ENQUIRIES” requires four modules

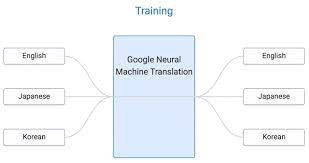
**1)Automatic Speech Recognition:** This module automatically collects the speech from the user through microphone while the translator system started running. It will help in converting the spoken words & phrases into the text in the same language. The output of this module is transfers to the second module



**Fig. 7.1 Speech Recognition**

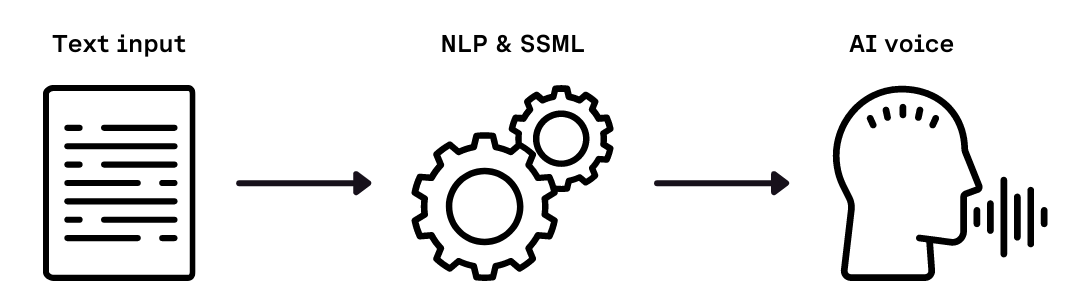
**2) Machine Translation:** The output from the first module is the input of this module.This module will help in converting the text into a second language. It will replace each word in the text with the appropriate word in the second language. The

Text generated in this module is transfers to the next module.



**Fig. 7.2 Machine Translation**

**3) Speech Synthesis:**The translated text which is generated in the previous module is the input of this module. It will estimate the pronunciation of the text generated by machine translation and generate the speech in the second and desired language.

****

**Fig. 7.3 Speech Synthesising**

**4) Text Output to Document:** The translated text generated in the module 2 is the input of this module. If we need to report document of the interrogation through this module documentation becomes easy in this module output text is converted to .txt file Through this .txt we can make any type of Document

OUTPUT



**Fig. 7.4. Document Generating**

**8. SAMPLE CODE**

**Importing library Files:**

from tkinter import \*

from tkinter.ttk import \*

import speech\_recognition as sr

from google\_trans\_new import google\_translator

import pyttsx3

from gtts import gTTS

import os

from datetime import datetime

languages={"English":"en","Telugu":"te",

           "Hindi":"hi","Tamil":"ta",

           "Kannada":"kn","Malayalam":"ml"

            }

master = Tk()

title = Label(master, text = "Speech Project",font= ("Helvetica 20"), justify= CENTER)

l1 = Label(master, text = "Source Language:",font= ("Helvetica 16"))

l2 = Label(master, text = "Destination Language:",font= ("Helvetica 16"))

l3 = Label(master, text = "Click on Speak Now Button",font= ("Helvetica 16"), justify= CENTER)

src = Combobox(master)

dest = Combobox(master)

t1 = Text(master, height = 20, width = 52)

lang=list(languages.keys())

src['values']=lang

src.set(lang[0])

dest['values']=lang

dest.set(lang[2])

style = Style()

def trans(msg,in\_lang,out\_lang):

#translating the language basing on text

    from googletrans import Translator

    translator = Translator()

    translate\_text\_1 = translator.translate(msg, src=in\_lang, dest=in\_lang)

    print(translate\_text\_1.text)

    translate\_text = translator.translate(msg, src=in\_lang, dest=out\_lang)

    print(translate\_text.text)

    final\_msg = "{} ({})\n".format(translate\_text\_1.text,translate\_text.text)

# printing the sentences in both source and destination language

    t1.insert(END,final\_msg)

    print("END")

    date\_string = datetime.now().strftime("%d%m%Y%H%M%S")

    fname="voices/Voice\_"+date\_string+".mp3"

    myobj = gTTS(text=translate\_text.text, lang=out\_lang, slow=False)

    myobj.save(fname)

    from playsound import playsound #speech synthesising

    playsound(fname)

        print(translate\_text)

def recog():

#speech recognizing through microphone ,recording and converting into text

    recognizer=sr.Recognizer()

    engine = pyttsx3.init()

    global l3

    with sr.Microphone() as source:

        print('Clearing background noise...')

        recognizer.adjust\_for\_ambient\_noise(source,duration=0)

        print('Waiting for message..') #waiting for input message

audio = recognizer.listen(source,timeout=8)

        l3["text"] = 'Recording Completed..' #recording

        print('Done recording..')

    try:

        print('Recognizing..')

        result = recognizer.recognize\_google(audio)

        print("Message:",result) #converting speech to text

        print("Destination Language:",dest.get())

        trans(result,languages[src.get()],languages[dest.get()])

    except Exception as ex:

        print(ex)

def swap():

#swapping the source and destination languages

    a=src.get()

    b=dest.get()

    src.set(b)

    dest.set(a)

style.configure('big.TButton', font=("Helvetica", 20), foreground="blue4")

b2 = Button(master,text=u"\u21c5",command=swap, width=2, style="big.TButton")

b1 = Button(master,text="Speak Now",command=recog, style="big.TButton")

title.grid(row = 0, column = 0, pady = 2,columnspan = 2)

l1.grid(row = 1, column = 0, sticky = W, pady = 2)

l2.grid(row = 2, column = 0, sticky = W, pady = 2)

src.grid(row = 1, column = 1, pady = 2)

dest.grid(row = 2, column = 1, pady = 2)

b2.grid(row = 1, column = 2, pady = 2,rowspan=2)

b1.grid(row = 3, column = 1, pady = 2)

l3.grid(row = 4, column = 0, pady = 2,columnspan=2)

t1.grid(row = 5, column = 0, pady = 2,columnspan = 2)

master.geometry('500x550')

mainloop()

**9. TESTING**

**9.1 Unit Testing**

Unit testing is a level of software testing where individual units/components of software are tested. The purpose is to validate that each unit of software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

In object-oriented programming, the smallest unit is a method, which may belong to a base/super class, abstract class or derived class/child class. It is performed by using the **white box testing.** Unit testing is the first level of software testing and is performed prior to the integration testing.

Unit testing increases the confidence in changing/maintaining the code of our project to detect malaria disease. If good unit tests are written and if they are run every time, if any code is changed, we will be able to catch any defects introduced due to the change. Also, if codes are already made less independent to make unit testing possible, the unintended impact of changes to any code is less. Codes are more reusable in order to make unit testing possible, codes need to be modular. This means that codes are easier to reuse.

The cost of fixing a defect detected in our project code during unit testing is lesser in comparison to that of defects detected at higher levels.

**9.2 Integration testing**

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in integrating testing.

Any of black box testing, white box testing and gray box testing methods can be used. Normally the method depends upon your definition of unit.

It is the second level of testing performed after unit testing and before system testing.

**Functional testing**

Functional testing is a type of software testing where by the system is tested against the functional requirements/specifications.

Functions are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but doesn’t make any system structure assumptions.

During functional testing, black box testing is used in which the internal logic of the system being tested is not known to the tester. Functional testing is normally performed during the levels of system testing and acceptance testing.

**9.3 System testing**

System testing is a level of software testing where complete and integrated software is tested. The purpose of this test is to evaluate system’s compliance with the specified requirements.

It is the process of testing an integrated system to verify that it meets specified requirements. Usually black box testing method is used.

System testing is the third level of software testing performed after integrated testing and before acceptance testing.

**9.4 Black Box testing**

Block box testing is also known as behavioral testing is a software testing method in which the internal structure of our project to detect malaria being tested is not known to the tester. These tests are usually functional or non-functional.

Tests are done from a user’s point of view and will help in exposing discrepancies in the specifications. Here, tester need not know programming languages or how the software has been implemented.

Test cases can be designed as soon as the specifications are complete.

**9.5 White Box Testing**

White box testing is also known as clear box testing, open box testing, glass box testing, transparent box testing, code based testing or structural testing is a software testing method in which the structure of an item being tested is known to the tester. The tester chooses input to exercise paths through the code and determines the appropriate outputs.

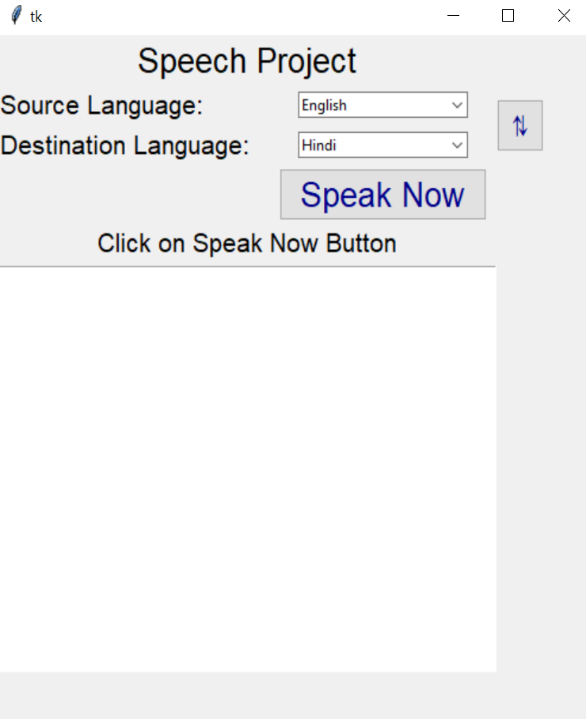
Testing based on an analysis of the internal structure of the component or system. It is a procedure to derive test cases of our project to detect malaria parasites based on the analysis of the internal structure of a component or system.

**Unit testing:**

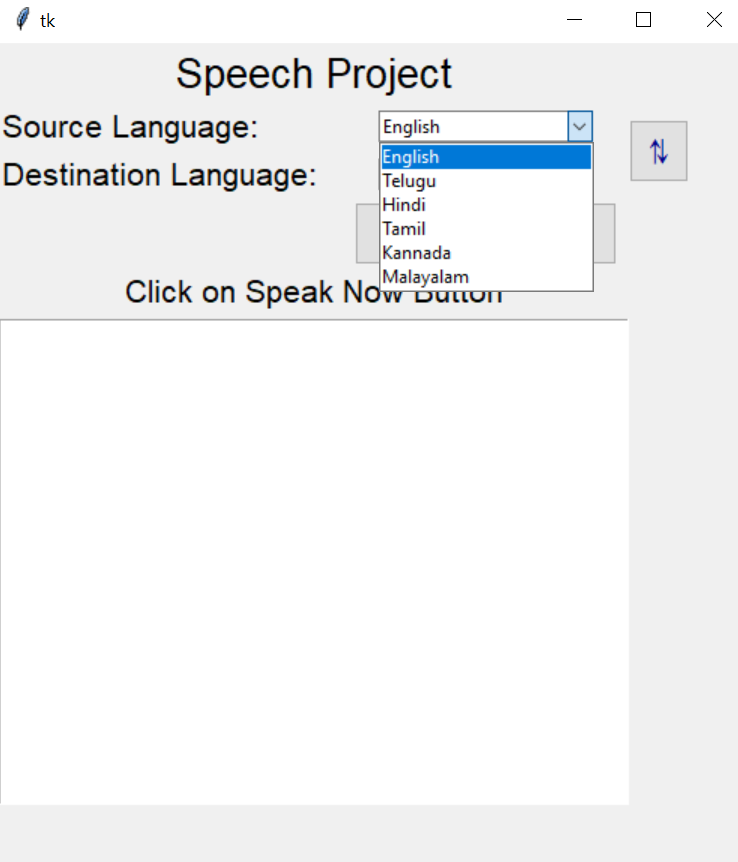
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Testcase Id** | **Test Description** | **Expected Results** | **Actual Results** | **Status** |
| TC001 | Speak the sentence in one language | speech should be recorded | Speech recorded | pass |
| TC002 | Translate the text  sentence into another language text | Translate the sentence | Translated | pass |
| TC003 | Changing of destination language to source and vice versa | Swapping of source language and destination language | Swapped Successfully | pass |
| TC004 | Generate the speech of translated text | Generating speech  With pronounciation | Speech generated | pass |
| TC005 | Displaying the source language text and destination language text in the text box | Both languages text should be display | Both languages text should be displayed | pass |

**TABLE 9.1 Testcases Table**

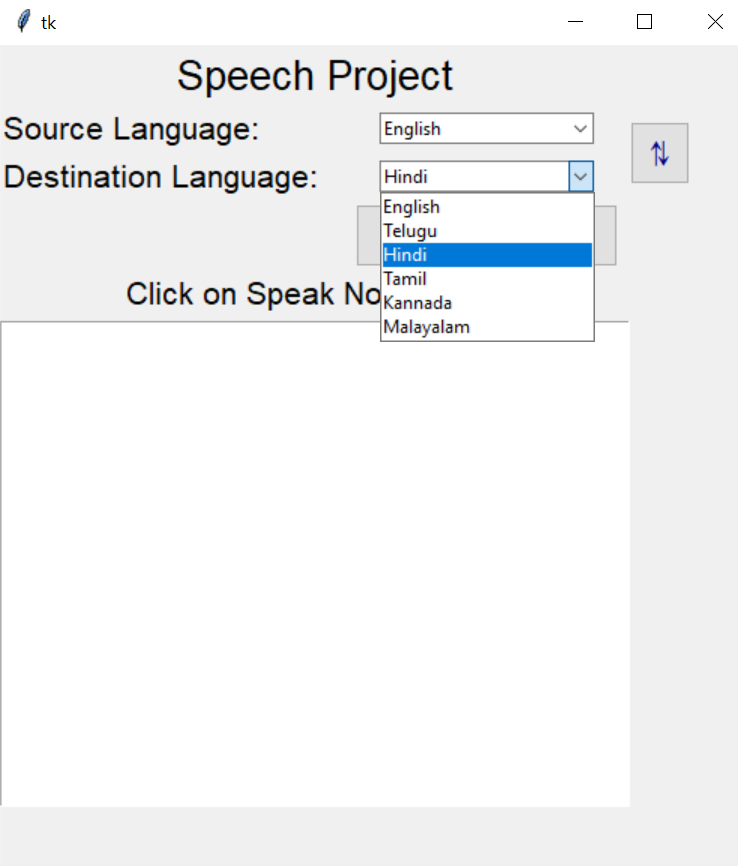
**10. SCREENSHOTS**



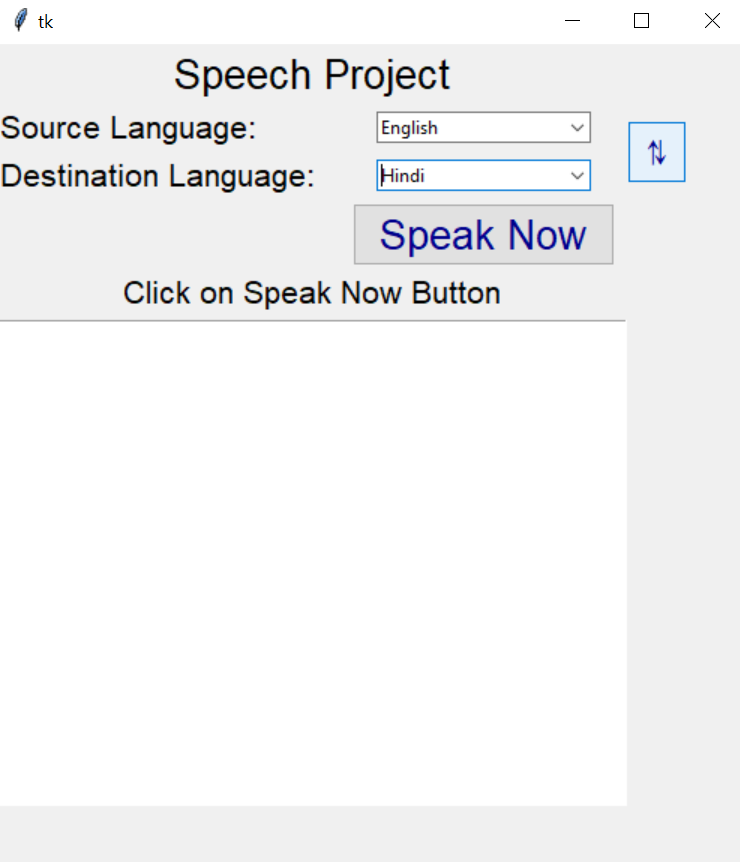
**Screenshot 10.1 Home page**



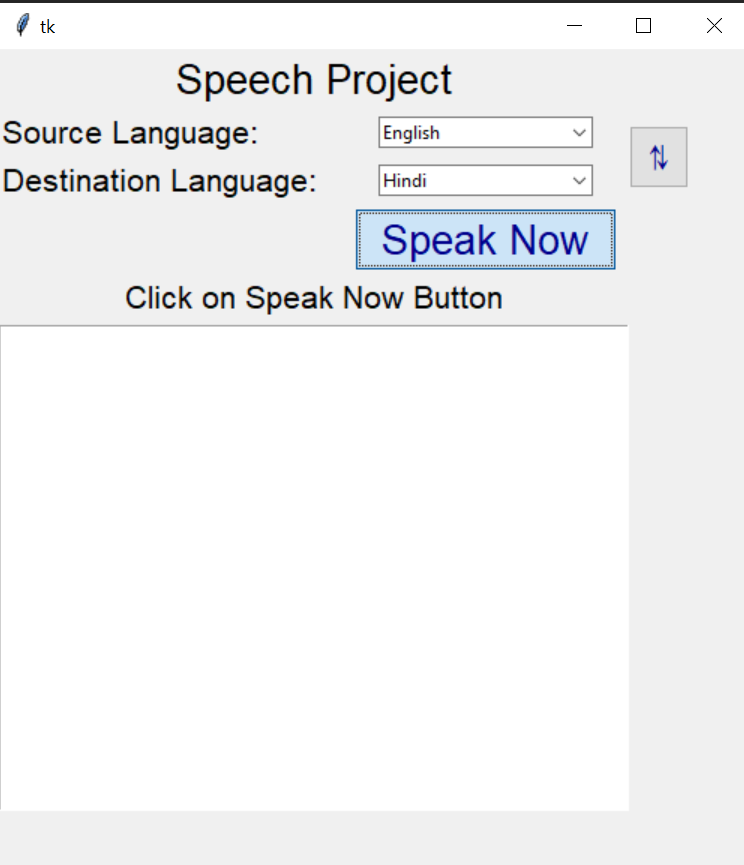
**Screenshot 10.2 Selecting Source Language**



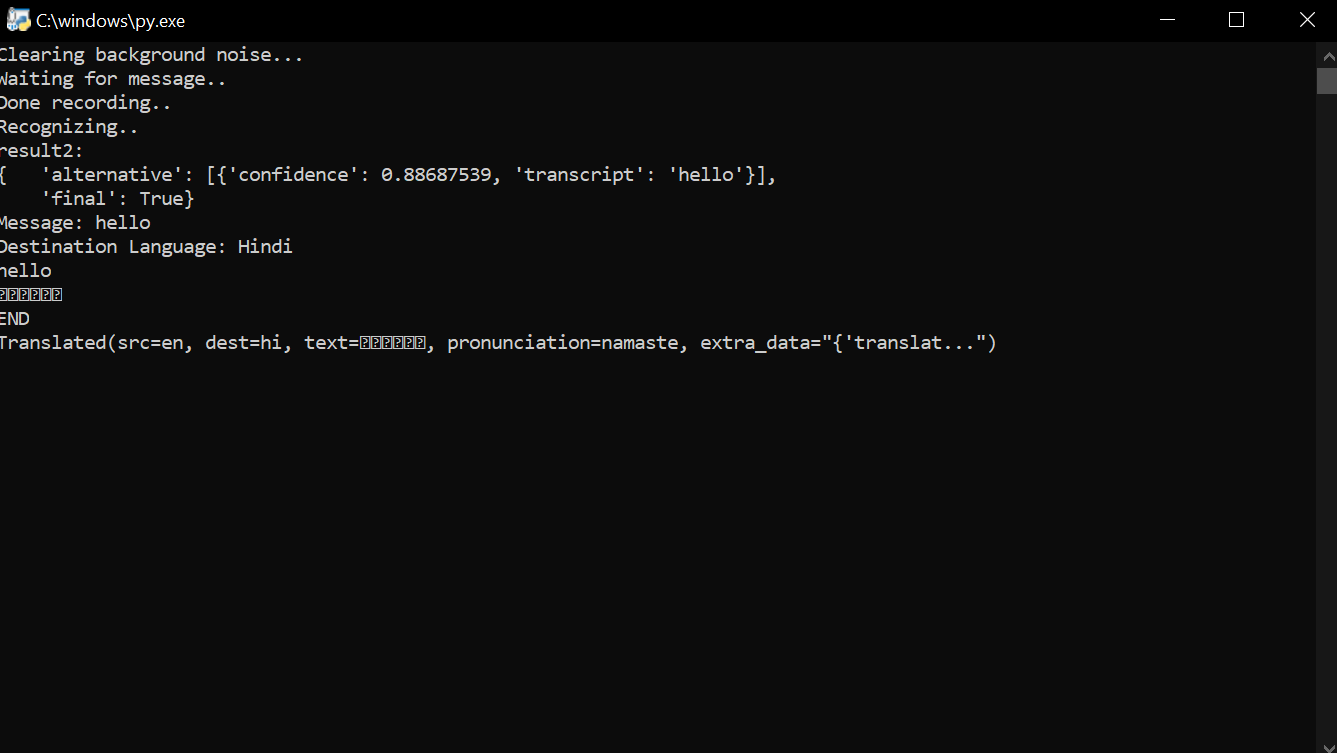
**Screenshot 10.3 Selecting Destination Language**



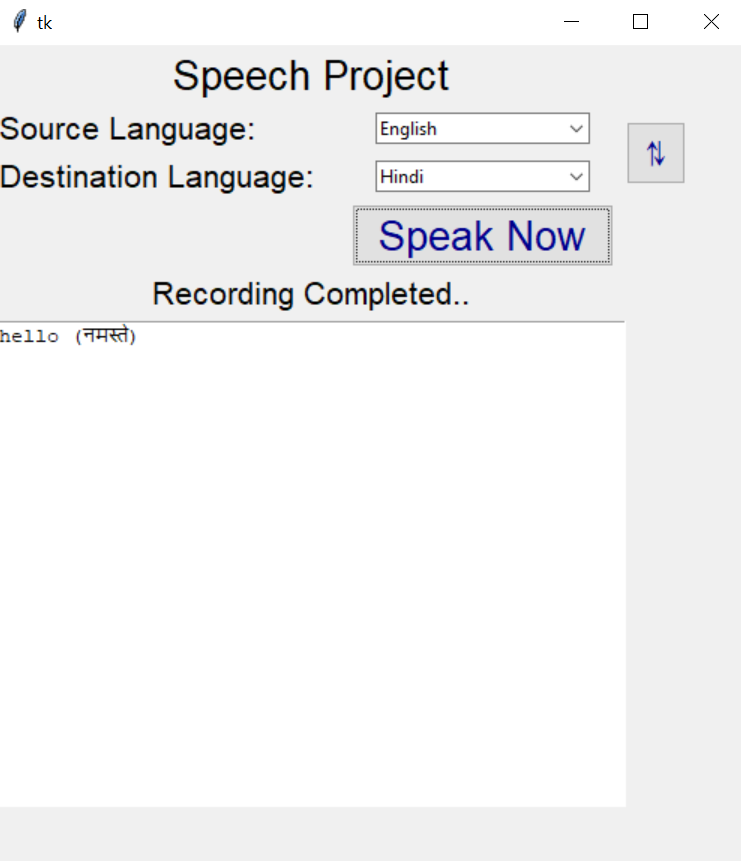
**Screenshot 10.4 Swap Button**



**Screenshot 10.5 Recording the speech**



**Screenshot 10.6 Translating and Accuracy**



**Screenshot 10.7 Printing the Source and Destination Language text**

**CONCLUSION**

This section summarizes and concludes the contributions made by our project. A system is developed for the real time speech to speech translation based on the input speech given by user in any of the six languages they are Tamil, Malayalam, Kannada, English, Hindi and Telugu. After the input speech is given, language translation is done via the Google API. Language barrier reduction is done via the speech-to-speech translation system. The system can defeat the constant challenges of unskilled individuals and improve their way of interrogating.

The outcomes show sensibly great achievement in perceiving Continuous speech from different speakers, for an enormous vocabulary. The various modules were examined in their separate areas and were effectively checked for various speech input.

**REFERENCES**

1. Kei Hashimoto1, Junichi Yamagishi2, William Byrne3 Simon King2, Keiichi Tokuda, “An analysis of machine translation and speech synthesis in speech-to-speech translation system” proceedings of 5108978-1-4577-0539-7/11/$26.00 ©2011 IEEE.
2. R. Rosenfeld, D. Olsen, and A. Rudnicky, "Universal speech interfaces," in Proceedings of the International Conference on Spoken Language Processing, Beijing, China: 2000.
3. <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>
4. Multilingual speech-to-speech translation systemfor mobile consumer devices.Seung Yun, Young-Jik Lee, Sang-Hun Kim.2014 IEEE Transaction onConsumer Electronics.
5. Sanjib Das, “Speech Recognition Technique: A Review”,International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 Vol. 2, Issue 3, May-Jun 2012.
6. DOUGLAS O’SHAUGHNESSY, SENIOR MEMBER, IEEE, “Interacting With Computers by Voice: Automatic Speech Recognition and Synthesis” proceedings of THE IEEE, VOL. 91, NO. 9, SEPTEMBER 2003
7. Hirschberg, Julia, and Christopher D. Manning. “Advances in Natural Language Processing.” Science 349(6245): 261–266. doi:10.1126/science.aaa8685.
8. Moore, Clayton. “The Most Useful Skills for Google Home.” Digital Trendsm(May3,2017).https://www.digitaltrends.com/home/google-home-most-useful-skills/.
9. https://towardsdatascience.com/neural-machine-translation-15ecf6b0b
10. P. Aguero, J. Adell, and A. Bonafonte, “Prosody generation for speech-to-speech translation,” in Proc.ICASSP, 2006.