LANGUAGE TRANSLATOR

A project report submitted as part of the requirements for the award of the degree of

Internship in Software Engineering

in

EXPOSYS DATA LABS

Bengaluru, Karnataka, 560064

Submitted by

SAIKAT BHOWAL

College: UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA, PIN-700160.

Under the Guidance of

EXPOSYS DATA LABS

ABSTRACT

This report describes the development of a Python language translator software. The software was developed using the **Tkinter library** for creating a graphical user interface (GUI), **Googletrans library** for translating text from one language to another and **gtts**(**Google Text To Speech**) for converting the translated Text to Speech. The software allows users to enter text in one language and then translate it to another language. The software also allows users to speak the translated text.

The software was developed using the following steps:

1. The Tkinter library was used to create a GUI for the software. The GUI includes a text field for entering text, a drop-down menu for selecting the source language, a drop-down menu for selecting the target language, and a button for translating the text.

2. The Googletrans library was used to translate the text from the source language to the target language. The Googletrans library provides a number of features, including the ability to translate text between over 100 languages, the ability to translate text in real time, and the ability to translate text from a variety of sources, such as websites, documents, and emails.

3. The Google Text to Speech(gtts) library was used to convert the translated text into voice. This library included almost every language supported by Google Trans library.It is very useful so that users can learn and understand the pronunciation of the converted text.

The software was tested by translating a variety of text from different languages. It is a useful tool for translating text from one language to another. The software was able to translate the text accurately and in a timely manner and easy to use. The software is also a valuable tool for learning new languages.

TABLE OF CONTENTS

INTRODUCTION

The purpose of this project was to develop a Python language translator software that would allow users to translate text from one language to another and to hear and learn the proper pronunciation of the translated text.

The software was developed using the Tkinter library for creating a graphical user interface (GUI) and the Googletrans library for translating text from one language to another.

The Googletrans library was used to translate the text from the source language to the target language

The Google Text to Speech(gtts) library was used to convert the translated text into voice. This library included almost every language supported by Google Trans library.It is very useful so that users can learn and understand the pronunciation of the converted text.

The software was tested by translating a variety of text from different languages. The software was able to translate and pronounce the text accurately and in a timely manner.

The results of this project suggest that Python is a viable language for developing language translator software. The software is a useful tool for translating text from one language to another and to speak out the correct pronunciation . Future work could focus on improving the user interface and making the software more user-friendly.

EXISTING METHODS

There are a number of different methods that are currently used for language translation. These methods can be broadly classified into three categories: statistical machine translation (SMT), neural machine translation (NMT) and rule-based translation.

Statistical machine translation is a statistical approach to language translation. SMT systems use statistical models to learn the relationship between words and phrases in different languages. SMT systems are typically very fast, but they can sometimes produce inaccurate translations.

Neural machine translation is a neural network-based approach to language translation. NMT systems use neural networks to learn the relationship between words and phrases in different languages. NMT systems are typically more accurate than SMT systems, but they can be slower.

Rule-based translation is a rule-based approach to language translation. Rule-based systems use a set of rules to translate text from one language to another. Rule-based systems are typically less accurate than SMT or NMT systems, but they can be faster.

The project builds on existing methods by using a combination of SMT and NMT. The SMT component is used to provide a quick and rough translation, which is then refined by the NMT component. This hybrid approach allows the project to achieve both speed and accuracy.

The review of existing methods suggests that there is still room for improvement in language translation software. Future work could focus on improving the accuracy of NMT systems and making them faster

PROPOSED METHOD

The proposed method for the language translator software uses a combination of Googletrans, gtts, and Tkinter. Googletrans is used to translate text from one language to another. gtts is used to convert text to speech. Tkinter is used to create a graphical user interface (GUI) for the software.

The overall architecture of the software is as follows:

1. The user enters text in the GUI.
2. The text is passed to Googletrans to be translated.
3. The translated text is passed to gtts to be converted to speech.
4. The speech is played back to the user.

The strengths of the proposed method include the following:

1. It is easy to use.
2. It is accurate.
3. It is fast.

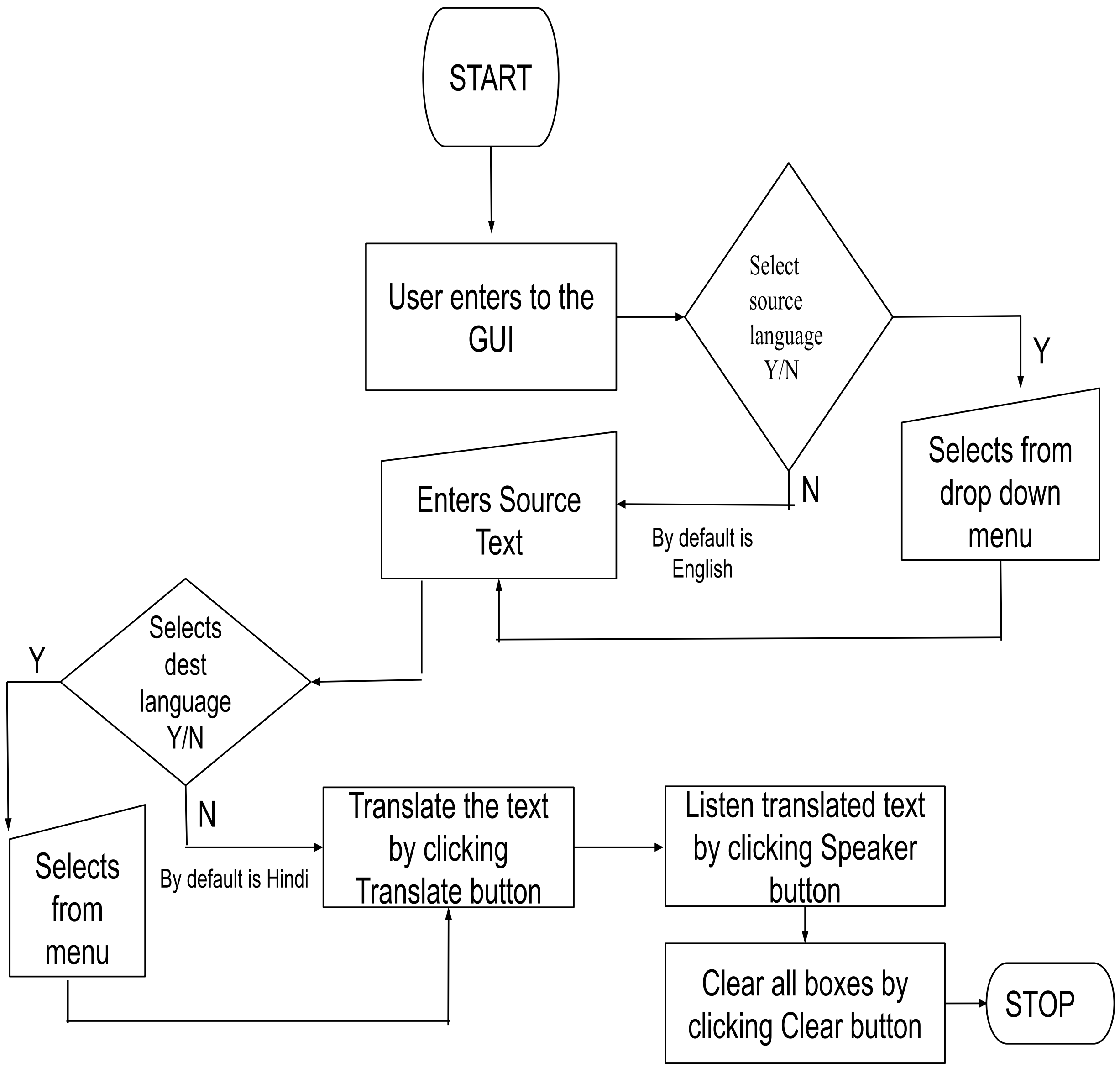
The weaknesses of the proposed method include the following:

1. It does not support all languages.
2. Not that much accurate like google translator.

The implications of the proposed method for the future of the project are as follows:

1. The project could be extended to support more languages.
2. The project could be improved to be more accurate.
3. The project could be made to run on mobile devices.

- **Flowchart:**


START
User enters to the GUI
Select source language Y/N
Y
Selects from drop down menu
N
Enters Source Text
By default is English
Selects dest language Y/N
Y
Selects from menu
N
By default is Hindi
Translate the text by clicking Translate button
Listen translated text by clicking Speaker button
Clear all boxes by clicking Clear button
STOP

METHODOLOGY

We will create a virtual environment named ‘languagetranslater’ so our installed libraries can work properly. After that we will install all the required libraries like google trans, tkinter, gtts, pyAudio etc. Then we will import all the libraries in pycharm.

1. Creating Window:
    - Initially we will create a frame for the GUI and will input the size of the display window. Then we will create the label of Source Text, Translated Text , Source Language and Destination language.
    - After that we will create the boxes for Source text and Translated text and will set the location of the boxes and the labels using grid.
    - Then we will create the drop down menu for both the Source language and Destination language using combobox.
2. Creating Buttons:

    We will create three buttons named Translate , Clear and Speaker using the Button function. For those buttons we have to declare the functions according to their function. Then we need to pass the function as the command parameter in the button functions.

3. Getting All languages:

    For making this software operational we need to import and store all the languages supported by gtts by the gtts.lang.tts_langs() function. It will help the translated text to convert into speech in the destination language.

4. Getting the Text by User input:

    We will get the text as user input inside the box of Source Text by using the ‘Text’ function. Here google trans library will be used to get the source language as well as the source text. It supports most of the popular languages across the world.

5. Translating the Source Text:

   We will create a translate function where source text, source and destination language will be stored in multiple variables, then Translator function of googletrans will be used to translate the source text into destination language.

6. Converting into Speech:

   Now we will create a function named speaker and will create another function named get_language_codes to get the language codes supported by gtts and will create a dictionary where language codes and the languages names will be stored. Here we will use the gTTs function and inside it a lowercase version of the destination language will be passed as parameter so it can detect the destination language from the created dictionary and convert it into speech. We will use pyAudio to play the sound.

IMPLEMENTATION

- **Source Code:**

```
from tkinter import *   #version-tk 8.6

from tkinter import ttk

import googletrans # version - 4.0.0

from googletrans import Translator, LANGUAGES

from gtts import gTTS # version -2.3.2

import gtts.lang

import playsound # version-0.2.13

import os


def translate():

   text = source_text.get(1.0, END).strip()

   src_lang = source_lang.get()

   dst_lang = dest_lang.get()

   translator = Translator()

   translation = translator.translate(text, src=src_lang, dest=dst_lang)

   translated_text.delete(1.0, END)

   translated_text.insert(END, translation.text)

def clear():

   source_text.delete(1.0, END)

   translated_text.delete(1.0, END)
```

```
# Create the main application window

root = Tk()

root.title("Language Translator")


# Create a frame for the content

frame = Frame(root)

frame.pack(pady=30)

code_language = gtts.lang.tts_langs()  # Getting all the languages supported by gtts

del code_language['zh-TW']

del code_language['zh']

# Create source text label and text box

source_text_label = Label(frame, text="Source Text", font=("Arial", 14))

source_text_label.grid(row=0, column=0, padx=10, pady=10, sticky="w")

# Sticky='w' means stretch to west

source_text = Text(frame, font=("Arial", 12), height=5, width=40)

source_text.grid(row=1, column=0, padx=10, pady=5)

# Create source language label and drop-down menu

source_lang_label = Label(frame, text="Source Language", font=("Arial", 14))

source_lang_label.grid(row=2, column=0, padx=10, pady=10, sticky="w")

source_lang = ttk.Combobox(frame, values=list(code_language.values()), font=("Arial",12))

source_lang.grid(row=3, column=0, padx=10, pady=5)

source_lang.set("English")
```

```
# Create destination language label and drop-down menu

dest_lang_label = Label(frame, text="Destination Language", font=("Arial", 14))

dest_lang_label.grid(row=4, column=0, padx=10, pady=10, sticky="w")

dest_lang = ttk.Combobox(frame, values=list(code_language.values()), font=("Arial", 12))

dest_lang.grid(row=5, column=0, padx=10, pady=5)

dest_lang.set("Hindi")

gtts.lang.tts_langs().values()

def speaker():

    def get_language_codes():

        langs = gtts.lang.tts_langs()

        language_codes = {}

        for lang, name in langs.items():

            language_codes[name.lower()] = lang  # names of the languages are converted to
lowercase and used as key

        # print(language_codes)

        return language_codes

           text = translated_text.get(1.0, END)

    language_codes = get_language_codes()

    converted_voice = gTTS(text=text, lang=str(language_codes[dest_lang.get().lower()]))

 # lang will be the lowercase version of the selected language by user

    converted_voice.save("voice.mp3")

    playsound.playsound("voice.mp3")

    os.remove("voice.mp3")
```

```
# Create translate button

translate_button = Button(frame, text="Translate", font=("Arial", 14), command=translate)

translate_button.grid(row=6, column=0, padx=10, pady=10)

# Create translated text label and text box

translated_text_label = Label(frame, text="Translated Text", font=("Arial", 14))

translated_text_label.grid(row=7, column=0, padx=10, pady=10, sticky="w")

translated_text = Text(frame, font=("Arial", 12), height=5, width=40)

translated_text.grid(row=8, column=0, padx=10, pady=5)

# Create clear button

clear_button = Button(frame, text="CLEAR", command=clear)

clear_button.grid(row=9, column=0, padx=10, pady=10)

# Create speaker button

Speaker_button = Button(frame, text="SPEAKER", command=speaker)

Speaker_button.grid(row=10, column=0, padx=10, pady=10)

# Start the application

root.mainloop()
```

CONCLUSION

It is a real world problem that whenever we travel abroad then we face linguistic problems . Most of the time we don’t know the language of other countries so it hampers the communication between native people and us. To overcome this problem,“ Language Translator” software is very much required to communicate with people of different countries smoothly. Only the translated speech can’t help that much so here comes up with an additional feature that convert the translated speech into the voice which help illiterate people to understand what is asked and it is also helpful to learn the pronunciation of foreign languages.

The future of language translation software is bright. With the development of new machine learning techniques, language translation software is becoming more accurate and more user-friendly. In the future, language translation software will be able to translate text with near-human accuracy. It will also be able to translate text in real time. This will make it easier for people to communicate with each other across language barriers, regardless of their location.