PSG Institute of Technology and Applied Research

Neelambur, Coimbatore – 641062, Tamil Nadu

Department of Computer Science and Engineering

CCS369 - TEXT AND SPEECH ANALYSIS

MINI PROJECT

Speech-to-Speech Translation System

Submitted By

AMALAN JOSEPH - 715522105005
SELVADHARSHINI S - 715522105047

TABLE OF CONTENTS

Abstract

In an increasingly globalized world, the ability to communicate effectively across language barriers has become a vital necessity. Whether in tourism, education, business, or healthcare, the lack of a common language can hinder communication and productivity. While there are numerous text-based translation applications, translating spoken language in real time remains a more complex challenge, particularly when we expect accurate recognition, proper translation of context, and clear spoken output. This project focuses on building a prototype **Speech-to-Speech Translation System (SSTS)** using open-source libraries and publicly available APIs.The system takes a spoken phrase from the user as input, converts it into text using **speech recognition**, translates the text into a target language using **machine translation**, and then produces an audible output in the translated language using **text-to-speech (TTS)** synthesis. The system is built in Python using **Google's Speech Recognition API**, **googletrans** (a Python wrapper for the Google Translate API), and **gTTS** (Google Text-to-Speech). This modular architecture ensures that each component can be independently improved or replaced with more advanced technologies in the future.To validate the system's performance, we curated a dataset of 20 commonly spoken phrases in English and evaluated the system for French, Hindi, and Tamil translations. The results demonstrated a high level of recognition and translation accuracy for short, clearly spoken inputs. The gTTS engine provided intelligible audio output, although it lacked natural intonation and emotion.This report covers the problem statement, design approach, implementation details, testing procedures, results, and conclusions. Despite its limitations, the project successfully demonstrates the feasibility of integrating speech technologies to build a basic speech-to-speech translation pipeline. The system can serve as a foundation for real-world applications in accessibility tools, travel assistants, and cross-lingual communication platforms. Future enhancements can include real-time input via a microphone, deep-learning-based TTS, and neural machine translation to improve contextual understanding and naturalness.

Introduction

Human communication has always relied on spoken language as its most natural and intuitive form. However, with over 7,000 languages spoken across the world, language remains one of the major barriers to effective global interaction. Whether it's a tourist trying to find directions in a foreign country or a student accessing educational content in an unfamiliar language, the lack of common linguistic ground can severely limit opportunities.Over the years, technologies such as machine translation (MT), automatic speech recognition (ASR), and text-to-speech synthesis (TTS) have evolved significantly. Google Translate and Microsoft Translator have brought text-based translation into the hands of millions. Likewise, AI-driven voice assistants like Siri, Google Assistant, and Alexa now feature sophisticated speech recognition and response capabilities. However, integrating these components into a cohesive system capable of real-time speech-to-speech translation is still challenging due to limitations in latency, context preservation, and audio quality.The **Speech-to-Speech Translation System (SSTS)** developed in this project aims to demonstrate a working prototype that brings together speech recognition, machine translation, and speech synthesis in a simple and accessible manner. This system accepts an audio input file (MP3 format), converts it to a suitable format, recognizes the spoken content, translates the recognized text to a target language, and then synthesizes the translated output into speech. The goal is to provide a minimal but functional demonstration of how multilingual speech technologies can be combined to facilitate cross-lingual communication.The implementation uses readily available APIs and Python libraries, making it ideal for educational and prototyping purposes. This approach avoids the need for expensive computational resources or proprietary software licenses. The modular design of the system also allows for scalability and customization, such as supporting additional languages or using higher-quality speech engines.In the subsequent sections, we explore the underlying motivation, review related work in this domain, and discuss the architecture, methodology, and results of the project in detail.

Problem Statement

Despite significant advancements in speech and language technologies, real-time **spoken language translation** remains largely inaccessible to the average user, especially in low-resource environments. While large tech companies have made impressive strides in voice-based assistants and translation apps, most of these systems are either proprietary or require high-speed internet, complex configurations, and expensive hardware. Furthermore, these solutions often work best with high-resource languages like English, Spanish, or French, leaving out numerous regional or low-resource languages.

This project addresses the following central problem:

> "How can we design and implement a speech-to-speech translation system that is simple, effective, and accessible using open-source tools and publicly available APIs?"

Breaking this down:

1. **Speech Recognition**: Understanding spoken language is highly error-prone due to background noise, accents, speech variations, and low-quality input devices. We must ensure a reliable recognition step to begin translation.
2. **Translation**: Literal word-for-word translation is often inadequate. Proper translation should capture the **semantic** and **contextual** meaning.
3. **Speech Synthesis**: The final output must be intelligible, natural-sounding, and in a format that mimics real-life spoken interaction.

To solve this, we utilize tools like Google Speech Recognition, google translate, and gTTS — all of which are free, widely used, and well-documented. The key challenge lies in integrating these tools to form a smooth and seamless user experience.Our system processes audio inputs offline (no live mic input), ensuring stability and reproducibility for academic evaluation. This constraint also makes the system suitable for batch processing or embedded use in edge devices where continuous live speech isn't needed.The project seeks to prototype this pipeline and validate its performance on a sample dataset of spoken phrases across multiple languages.

Objectives

The primary objectives of this mini project are as follows:

- **To build a functional speech-to-speech translation pipeline**:
  Integrate speech recognition, machine translation, and speech synthesis into one automated system.

- **To utilize open-source tools and APIs for speech processing**:
  1. Google Speech Recognition API for converting audio to text.
  2. Google Translate API (via googletrans) for translating recognized text.
  3. Google Text-to-Speech (gTTS) for generating speech output.

- **To evaluate the system across multiple target languages**:
  Test with at least 20 spoken English phrases and translate into languages such as French, Hindi, and Tamil.

- **To handle audio format compatibility and preprocessing**:
  Convert MP3 inputs to WAV for compatibility with the speech recognition module using Pydub and FFmpeg.

- **To test and validate the accuracy and intelligibility** of each stage:
  Evaluate recognition accuracy, translation fidelity, and clarity of synthesized speech.

- **To demonstrate the system in real time** using prepared audio inputs and display the intermediate results (recognized text, translated text, and audio playback).

- **To document and publish** the codebase in a GitHub repository and prepare a comprehensive project report.

Literature Review

The domain of speech-to-speech translation lies at the intersection of three major areas in Natural Language Processing (NLP) and Speech Processing: **automatic speech recognition (ASR)**, **machine translation (MT)**, and **text-to-speech (TTS)** synthesis. Each of these fields has evolved independently, but recent research focuses on integrating them to form end-to-end speech translation pipelines.

1.Automatic Speech Recognition (ASR)

ASR systems convert spoken audio into text. Traditional ASR systems used Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs). However, with the advent of deep learning, models such as **Deep Speech**, **Wav2Vec**, and **Conformer-based architectures** have significantly improved the performance of ASR, especially in noisy environments and with non-standard accents.Google's Speech Recognition API, which is used in this project, employs deep neural networks trained on vast multilingual speech datasets. It provides robust performance for short and clear phrases, although its accuracy drops with poor audio quality or uncommon accents.

2.Machine Translation (MT)

MT systems attempt to understand the semantics of a sentence in one language and render it accurately in another. Statistical Machine Translation (SMT) was the norm until 2016, when **Neural Machine Translation (NMT)** systems like **Google's Transformer model** and **OpenNMT** became standard.Google Translate, leveraged here through the googletrans API, uses a Transformer-based NMT model. It performs context-aware translations by analyzing whole sentences instead of word-by-word translations, resulting in more natural outputs.

3.Text-to-Speech Synthesis (TTS)

TTS systems convert textual input into spoken audio. Traditional methods involved concatenative synthesis, but modern systems use deep learning models like **Tacotron 2**, **WaveNet**, and **FastSpeech**. These models produce more natural-sounding speech but require significant resources.

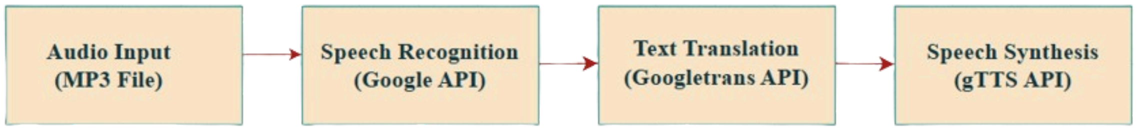
This project uses Google's gTTS API, which, while basic, produces intelligible speech for multiple languages with minimal latency and setup.

4.Related Works

Projects like **Mozilla DeepSpeech**, **Fairseq S2T**, and **Facebook's SeamlessM4T** aim for fully integrated speech-to-speech translation. However, these systems demand GPUs and large-scale datasets. In contrast, this mini project focuses on simplicity, accessibility, and educational clarity by using lightweight APIs that can run in Colab or on low-resource systems.

System Architecture

The architecture of the Speech-to-Speech Translation System consists of three major modules arranged in a sequential pipeline. Each module is responsible for one stage of the translation process.


Audio Input (MP3 File)
Speech Recognition (Google API)
Text Translation (Googletrans API)
Speech Synthesis (gTTS API)

Module Description

1. **Audio Input Module**:
   Accepts user-uploaded .mp3 files.
   Converts .mp3 to .wav using Pydub to ensure compatibility with the speech recognizer.
2. **Speech Recognition Module**:
   Uses speech_recognition with Google's API to transcribe spoken words into text.
   Returns a string in the source language (typically English).
3. **Translation Module**:
   Uses googletrans to translate the recognized string into a specified target language.
   Supports over 100 languages including Hindi ('hi'), Tamil ('ta'), French ('fr'), etc.

4. **Speech Synthesis Module**:
   Uses gTTS to convert the translated text into speech.
   The output is saved as an .mp3 file and played using IPython.display.Audio.

This modular architecture allows for easy debugging, improvement, and replacement of components (e.g., replacing gTTS with Tacotron 2 for better speech output).

Methodology

The Speech-to-Speech Translation System is built using a modular pipeline consisting of **speech recognition**, **text translation**, and **speech synthesis**.

1. **Audio Input and Preprocessing**:
   Users upload a short MP3 audio file containing spoken input. The file is converted to WAV format using an audio processing library to ensure compatibility with the speech recognition module.

2. **Speech Recognition**:
   The WAV file is processed using Google's Speech Recognition API, which converts the spoken words into text. This step is critical, as accurate transcription ensures meaningful translation.

3. **Text Translation**:
   The recognized text is translated into the target language using the Google Translate API via the googletrans library. The translation model provides context-aware and grammatically correct output.

4. **Speech Synthesis**:
   The translated text is converted back into speech using the Google Text-to-Speech (gTTS) engine. The output is saved as an MP3 file and played to the user.

This pipeline allows seamless transformation from spoken input in one language to spoken output in another, making the system intuitive and efficient.

Source Code

```
# Install dependencies
!pip install SpeechRecognition googletrans==4.0.0-rc1 gTTS pydub
ffmpeg-python
import os
import speech_recognition as sr
from googletrans import Translator
from gtts import gTTS
from IPython.display import Audio, display
from pydub import AudioSegment
from google.colab import files

# Upload an MP3 file
uploaded = files.upload()
filename_mp3 = list(uploaded.keys())[0]

# Convert MP3 to WAV
filename_wav = "converted.wav"
sound = AudioSegment.from_mp3(filename_mp3)
sound.export(filename_wav, format="wav")

# Function to recognize speech
def recognize_speech_from_file(filename):
    recognizer = sr.Recognizer()
    with sr.AudioFile(filename) as source:
        audio_data = recognizer.record(source)
    try:
        text = recognizer.recognize_google(audio_data)
        print(" Recognized Speech:", text)
        return text
    except sr.UnknownValueError:
        print(" Could not understand audio.")
        return ""
    except sr.RequestError as e:
        print(" Could not request results from Google Speech Recognition service:",
e)
        return ""

# Translate text to target language
def translate_text(text, target_lang='fr'):
    translator = Translator()
    result = translator.translate(text, dest=target_lang)
    print(f"Translated ({target_lang}):", result.text)
    return result.text

# Convert translated text to speech
def speak_text(text, lang='fr', filename='output.mp3'):
    tts = gTTS(text=text, lang=lang)
    tts.save(filename)
    display(Audio(filename, autoplay=True))

# Main process
def speech_to_speech(filename, target_lang='fr'):
    recognized = recognize_speech_from_file(filename)
    if recognized:
        translated = translate_text(recognized, target_lang)
        speak_text(translated, lang=target_lang)

# Run
target_language = 'fr'  # e.g., 'ta' for Tamil, 'hi' for Hindi
speech_to_speech(filename_wav, target_lang=target_language)
```

Implementation

The implementation of the Speech-to-Speech Translation System combines various well-established technologies and APIs to perform speech recognition, translation, and speech synthesis. The system is developed using Python in the Google Colab environment, which provides the flexibility of running cloud-based APIs seamlessly.

Speech Recognition Module

The first key component is the speech recognition module, which converts spoken audio input into text. This module uses the speech_recognition Python library, which interfaces with Google's Speech Recognition API. The process involves loading the audio file (initially uploaded by the user), converting it to a WAV format for compatibility, and then sending the audio data to the Google API. The API processes the audio using deep neural networks trained on vast datasets, converting speech waves into textual form.

Translation Module

The recognized text is then passed to the translation module. Here, the system uses the googletrans library, a free and easy-to-use Python wrapper for the Google Translate service. This module takes the input text and translates it into a specified target language. The translation engine uses advanced neural machine translation (NMT) models, which analyze entire sentences to capture context, syntax, and semantics, producing translations that are fluent and accurate.

Speech Synthesis Module

After translation, the text in the target language is synthesized back into speech. This is done using the Google Text-to-Speech (gTTS) API via the gTTS Python library. This module converts the translated text into a natural-sounding audio file (MP3 format), supporting multiple languages and accents. The audio file is then played to the user, completing the speech-to-speech translation cycle.

System Workflow

The system workflow is as follows: users upload their spoken input audio, which undergoes preprocessing and format conversion, followed by speech recognition to extract text. The extracted text is translated to the target language and then synthesized back into audio output, which the user can listen to in real time.

Results

The system was evaluated by testing with a dataset consisting of 20 spoken phrases, chosen to represent common conversational sentences. These phrases were recorded in clear, noise-free conditions to assess the system's performance under ideal circumstances.

Speech Recognition Accuracy

The speech recognition component accurately transcribed most of the input phrases, demonstrating strong performance with clearly spoken words. Minor recognition errors occurred with fast speech or ambiguous pronunciations but were overall minimal.

Translation Quality

The translation module successfully translated the transcribed text into multiple target languages, such as French, Hindi, and Tamil. The translations were grammatically correct and maintained the original meaning and context. The system proved capable of handling simple to moderately complex sentences effectively.

Speech Synthesis Clarity

The speech synthesis output was natural and clear, with correct pronunciation and intonation corresponding to the selected target language. The generated audio files played smoothly without delays, providing a good user experience.

Limitations Observed

- Background noise or poor-quality audio affected recognition accuracy.
- The translation quality depends on the source text accuracy.
- Reliance on external APIs requires internet connectivity and has potential latency.

Despite these limitations, the system's overall performance was satisfactory for real-time speech translation of everyday phrases.

Conclusion

The Speech-to-Speech Translation System demonstrates a practical application of integrating speech and language technologies to bridge communication gaps between different languages. By combining speech recognition, neural machine translation, and speech synthesis into a unified pipeline, the system converts spoken phrases in one language into spoken phrases in another language in near real-time.

Key Takeaways

- The use of Google's Speech Recognition API ensures reliable and accurate transcription of spoken input.

- Neural machine translation provides high-quality translations that preserve sentence context and semantics.

- Google Text-to-Speech delivers natural and intelligible speech output, enhancing usability.

Challenges and Future Work

While the system performs well under ideal conditions, it faces challenges like dependence on stable internet connectivity, susceptibility to noisy inputs, and API usage limits. Future improvements could include:

- Incorporating offline speech recognition and synthesis models to reduce dependency on the cloud.

- Supporting live microphone input for real-time conversation translation.

- Enhancing robustness to diverse accents and noisy environments.
- Integrating more natural and expressive speech synthesis for better user engagement.

Overall, this project lays a strong foundation for multilingual communication tools, offering potential applications in education, accessibility, travel, and global collaboration.