Enhancing Multilingual Communication through NLP and Speech Recognition Integration

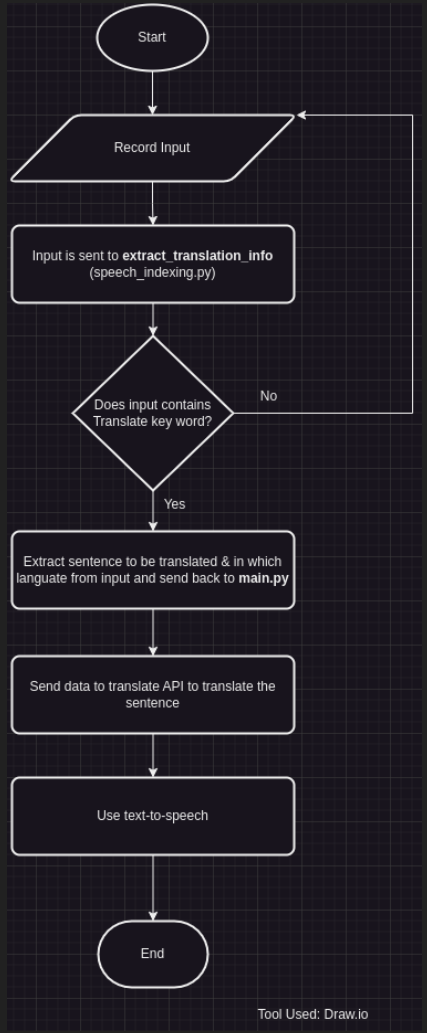
Name: Pranav Dalvi
Roll No.: 04
Organisation: Kirti M. Doongursee College
Date: 13/06/2024

Introduction

- Challenges of communication across languages
  - Language barriers, Cultural differences, Vocabulary limitations, Vocabulary limitations.
- Advancements in NLP and Speech Recognition.
  - Powerful language models like spaCy, BERT, and GPT can understand and process human language with greater accuracy. NLP can analyze text sentiment, identify keywords, and even generate human-quality text.
  - Technologies like Google Speech Recognition allow computers to transcribe spoken language into text with high accuracy. This has revolutionized voice assistants, voice search, and dictation software.

System Overview

- The project integrates the speech_indexing.py module for language detection and cleaning the input with the main.py module for speech recognition and translation.
- The project uses combination of spaCy for NLP tasks and Google APIs for speech recognition and translation.



Methodology

- Language Detection and Translation Command Extraction:
  - Utilizes spaCy for tokenizing and analyzing the structure of input sentences.
  - Detects language keywords using tokenized lists of input and pre-defined lists.
  - Cleans the text to remove unnecessary command phrases for accurate translation using regular expressions.
- Speech Recognition and Translation:
  - Uses the SpeechRecognition library to capture and transcribe spoken input.
  - Integrates Google Translate API for translating transcribed text.
  - Converts translated text to speech using Google Text-to-Speech (gTTS).

Implementation: speech_indexing.py

```python
def clean_text(text, language):
    text = re.sub(rf"\b(?:translate|convert|translation)\b.*\b(into|to)\s+{language}\b", "", text, flags=re.IGNORECASE)
    text = re.sub(rf"\b(into|to)\s+{language}\b", "", text, flags=re.IGNORECASE)
    text = re.sub(r"\b(this sentence|of following sentence)\b", "", text, flags=re.IGNORECASE)
    return text
```

```python
def extract_translation_info(sentence):
    doc = nlp(sentence)
    translate_index = None
    for token in doc:
        if token.lemma_ == "translate" or token.lemma_ == "convert" or token.lemma_ == "translation":
            translate_index = token.i
            break
    language = detect_language(sentence)
    if translate_index is not None and translate_index + 1 < len(doc):
        text_to_translate = ""
        for token in doc[translate_index + 1:]:
            text_to_translate += token.text + " "
    else:
        text_to_translate = ""
    text_to_translate = clean_text(text_to_translate, language)
    return {
        "translate_index" : translate_index,
        "language": language,
        "text_to_translate": text_to_translate.strip()
    }
```

```python
def detect_language(text):
    supported_languages = ["marathi", "hindi"]
    for language in supported_languages:
        if language in text.lower():
            return language.lower()
    return None
```

Implementation: main.py

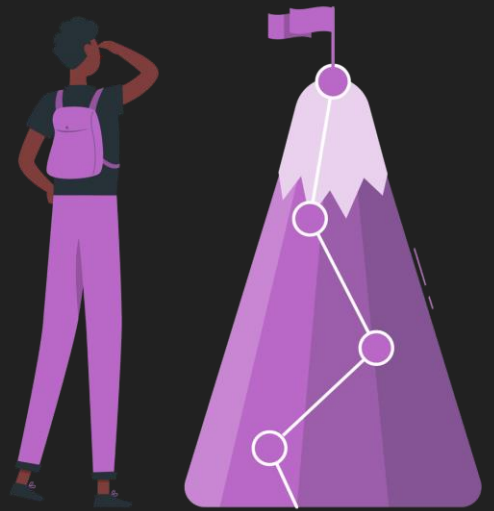
```python
def recordInput(src_lang):
    r = sr.Recognizer()
    with sr.Microphone() as source:
        r.pause_threshold = 1
        r.adjust_for_ambient_noise(source)
        audio = r.listen(source)
    try:
        text = r.recognize_google(audio, language = src_lang)
        print(f"Input: {text}")
        nlp_op = extract_translation_info(text)
        return nlp_op
    except Exception as e:
        print(f"Error: {e}")
        return "None"
```

```python
def translateText(text, tgt_lang):
    tgt_out = translator.translate(text, dest=tgt_lang)
    print(f"Output: {tgt_out.text}")
    return tgt_out.text
```

```python
def speak(text, tgt_lang):
    tts = gTTS(text=text, lang=tgt_lang)
    tts.save('./speech.mp3')
    playsound('./speech.mp3')
```

Aims and Objectives

- To develop a robust system that can accurately recognize and transcribe spoken language.
- To implement a language detection mechanism that identifies the target language for translation.
- To evaluate the system's performance across different languages and use cases.



Future Work

- To develop an Android app for the user interface.
- Increase accuracy by applying noise reduction techniques.
- Develop APIs for seamless application integrations.
- Testing with offline models for translation and speech recognition.

Thank You!

Got any questions?

References:

- The ATR Multilingual Speech-to-Speech Translation System: https://www.researchgate.net/publication/3457553_The_ATR_Multilingual_Speech-to-Speech_Translation_System
- SpaCy: https://spacy.io/usage
- Google Text-to-Speech API: https://gtts.readthedocs.io/en/latest/
- Google Translation API: https://py-googletrans.readthedocs.io/en/latest/