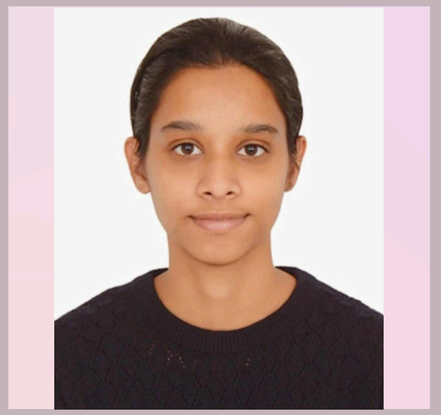
VOICE LINGUA

An Innovative Interface for Natural Language Processing Tasks

GEN-SPARK

HARSHDEEP SINGH



KIRANDEEP KAUR

LIZA KUMARI

INTRODUCTION

VOICE LINGUA is an innovative interface developed using Python and advanced AI technologies to provide a comprehensive suite of Natural Language Processing (NLP) tasks. It is designed to enhance the user experience with high accuracy and versatility in language processing.

COMPREHENSIVE NLP TASKS

Speech Recognition

Translation

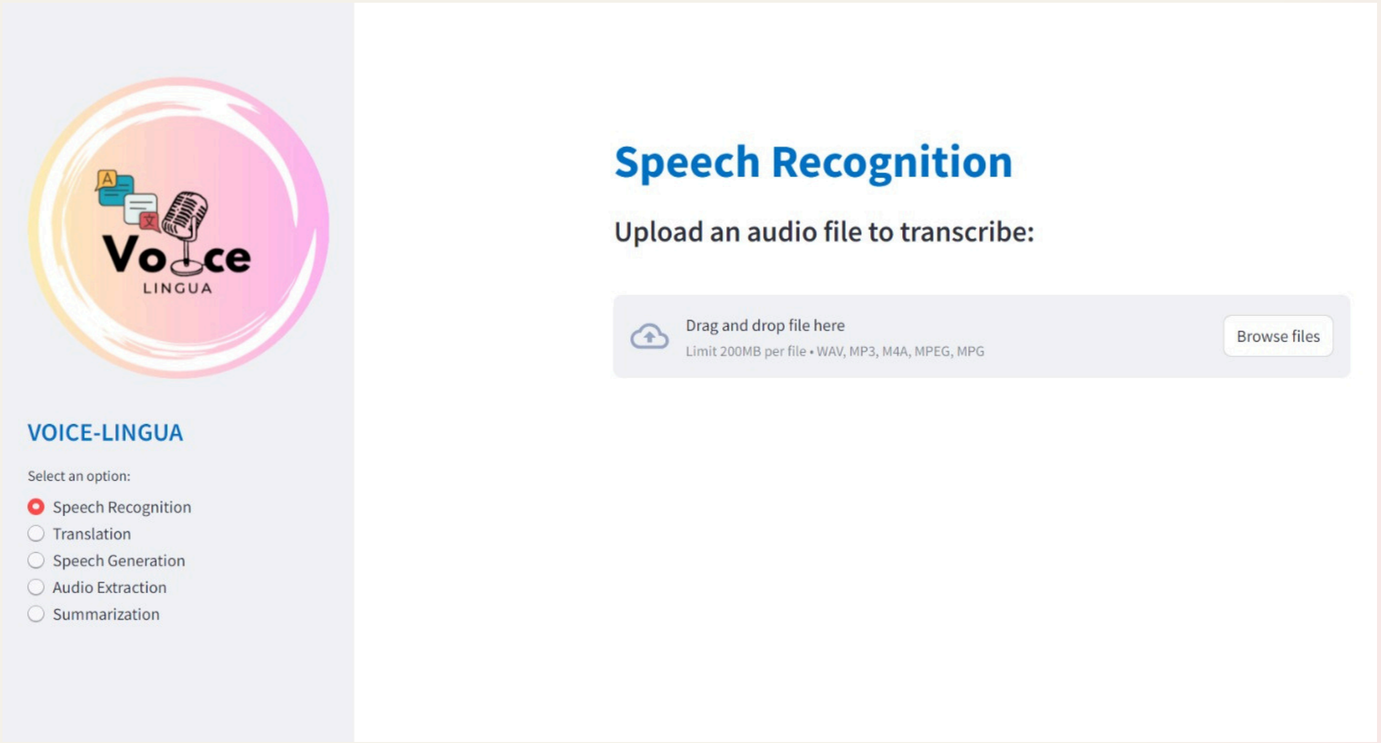
Speech Generation

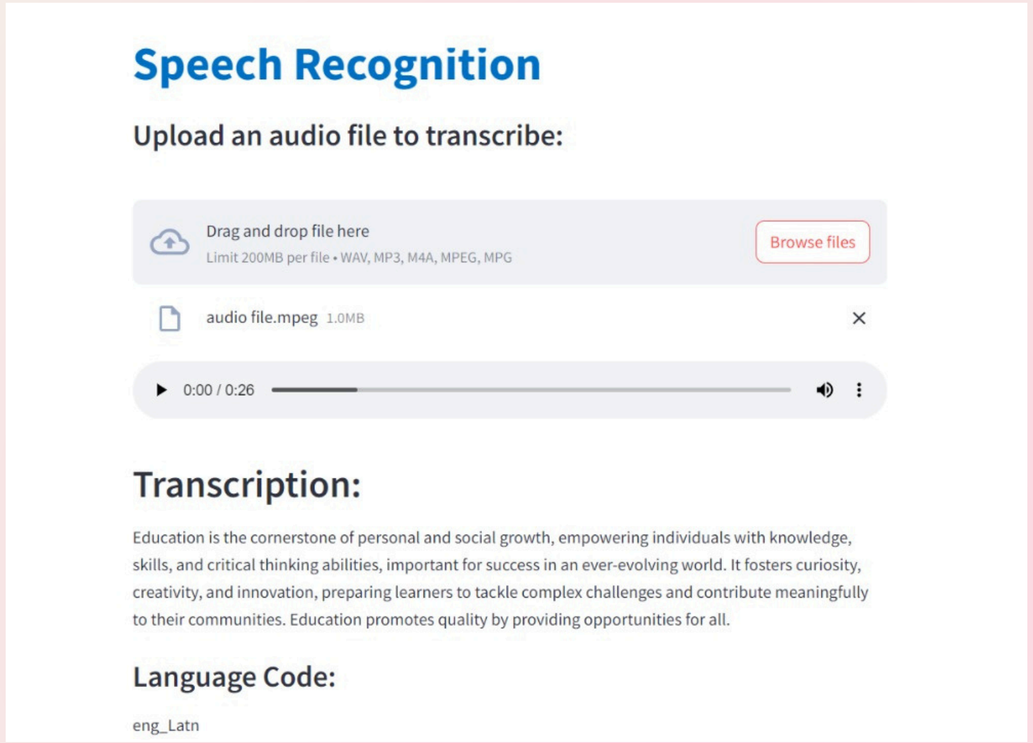
Audio Extraction

Summarization

SPEECH RECOGNITION

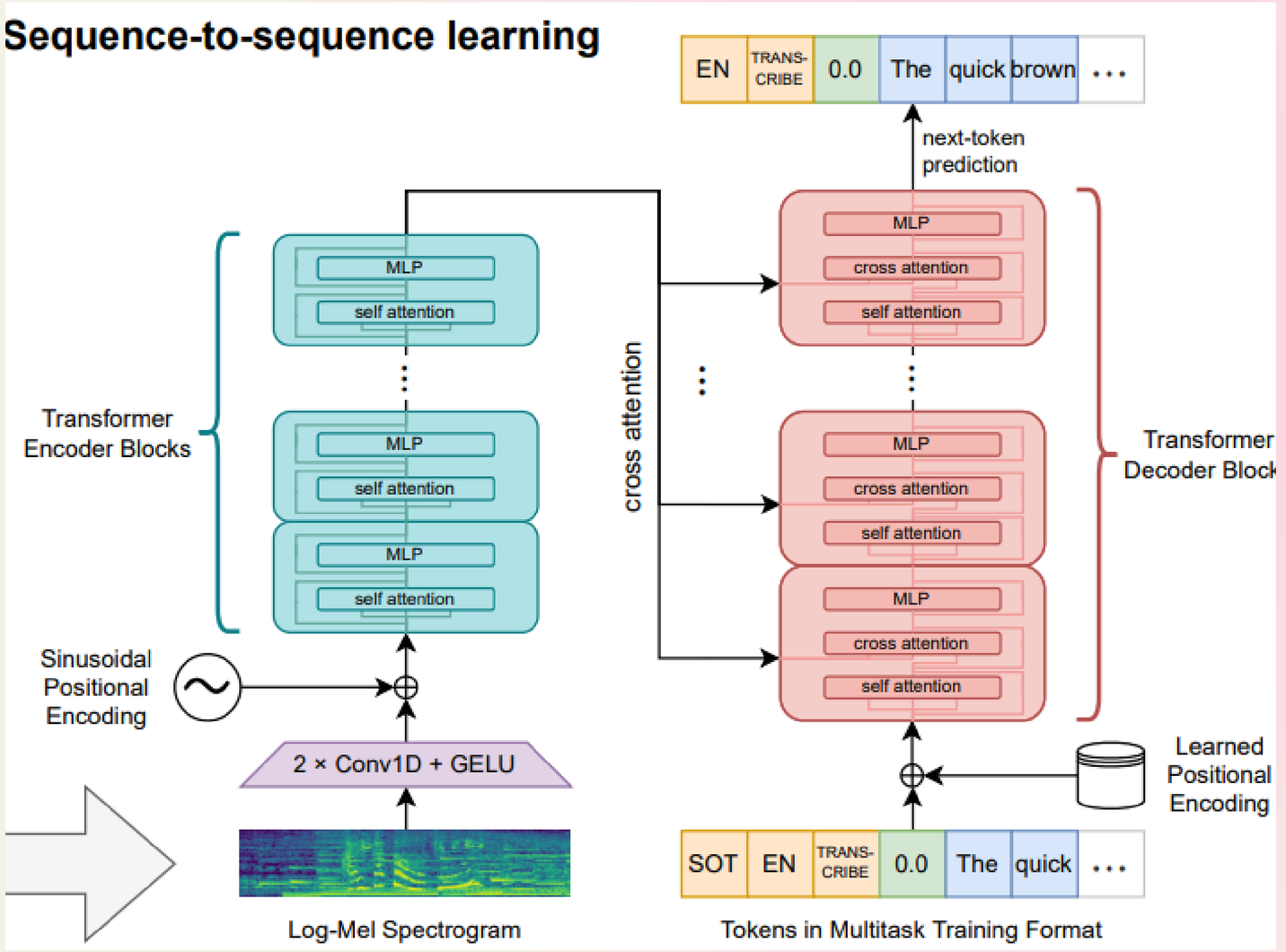
- Coverts speech into text in multiple languages.
- Support 50+ languages.
- Given by openai/whisper-large-v3 model





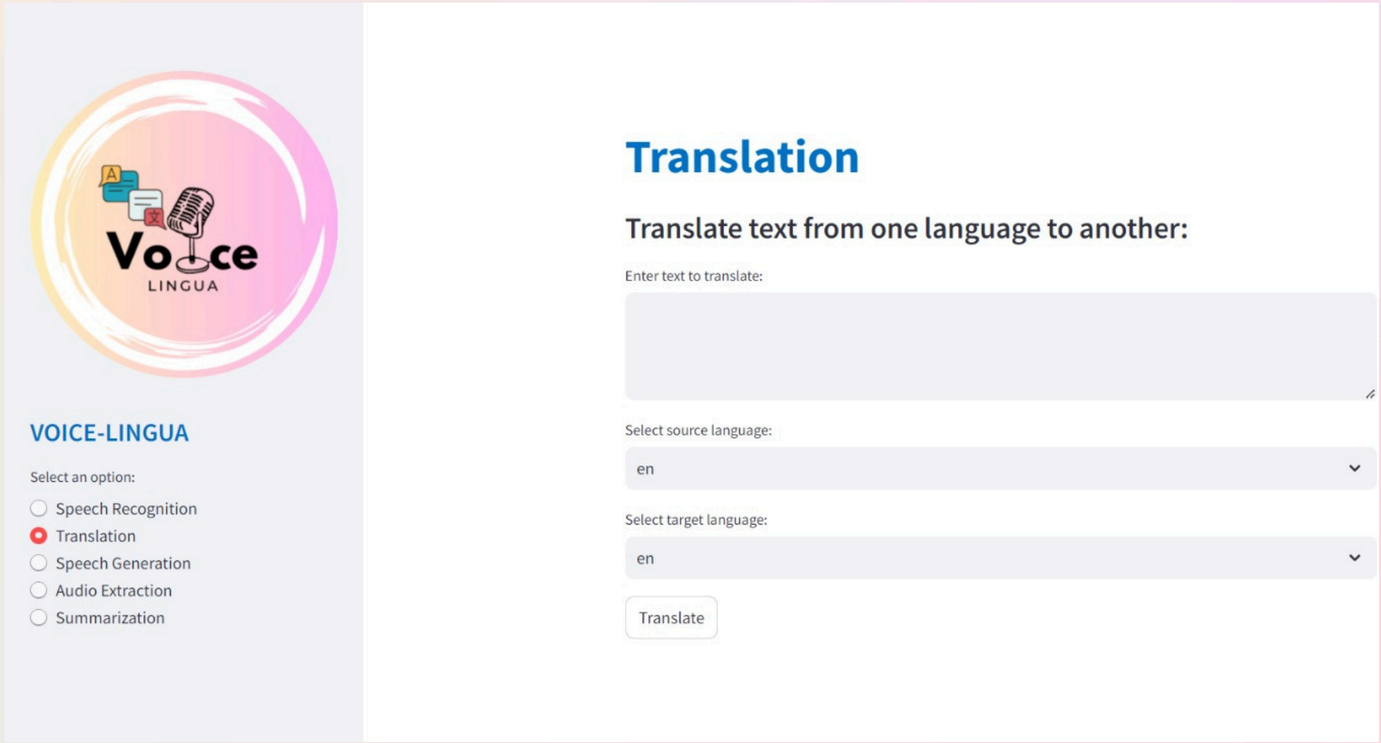
WORKFLOW

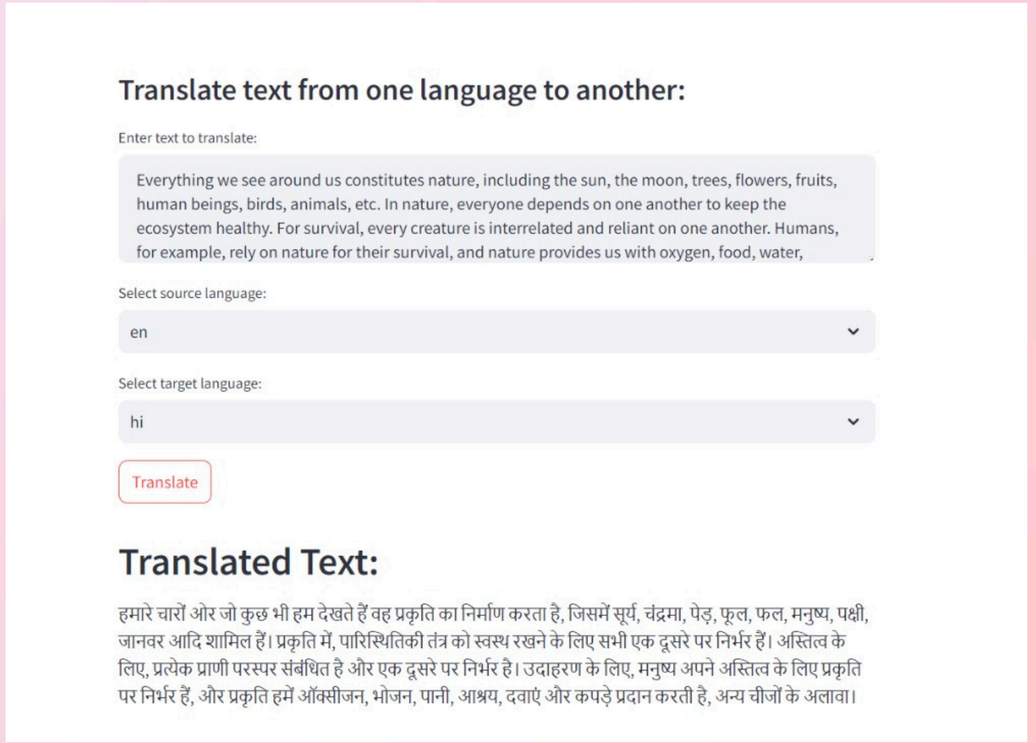
1. **Audio Preprocessing**: Convert audio signals into Mel spectrogram frames.
2. **Feature Extraction**: Extract important features from the audio frames.
3. **Positional Encoding**: Add positional information to the frames.
4. **Transformer Encoder**: Process frames to capture long-range dependencies.
5. **Transformer Decoder**: Decode frames to generate the text token-by-token.
6. **Output Generation**: Produce the final textual output



TRANSLATION

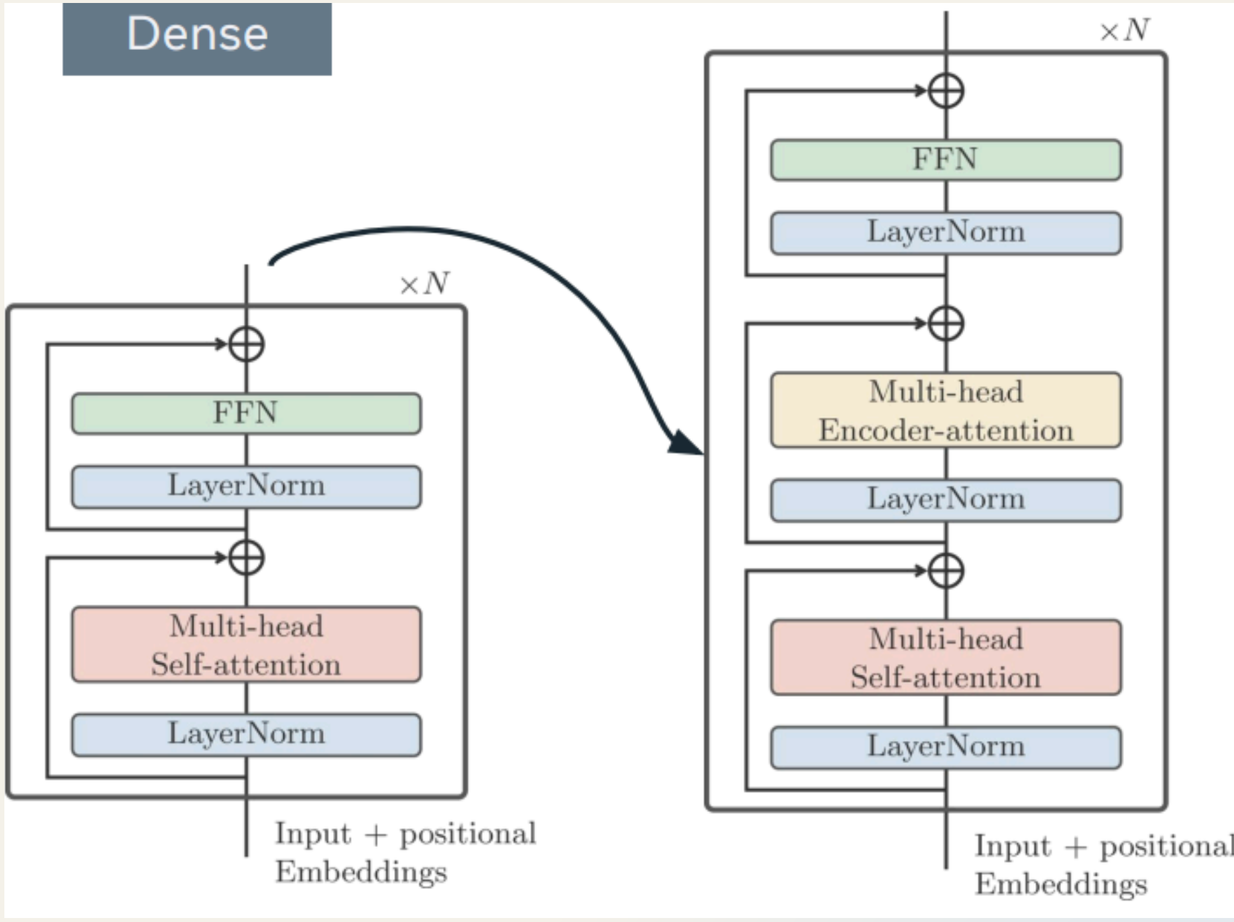
- Allows you to translate effectively between source and target languages.
- 50+ languages are supported.
- Powered by NLLB-200 model





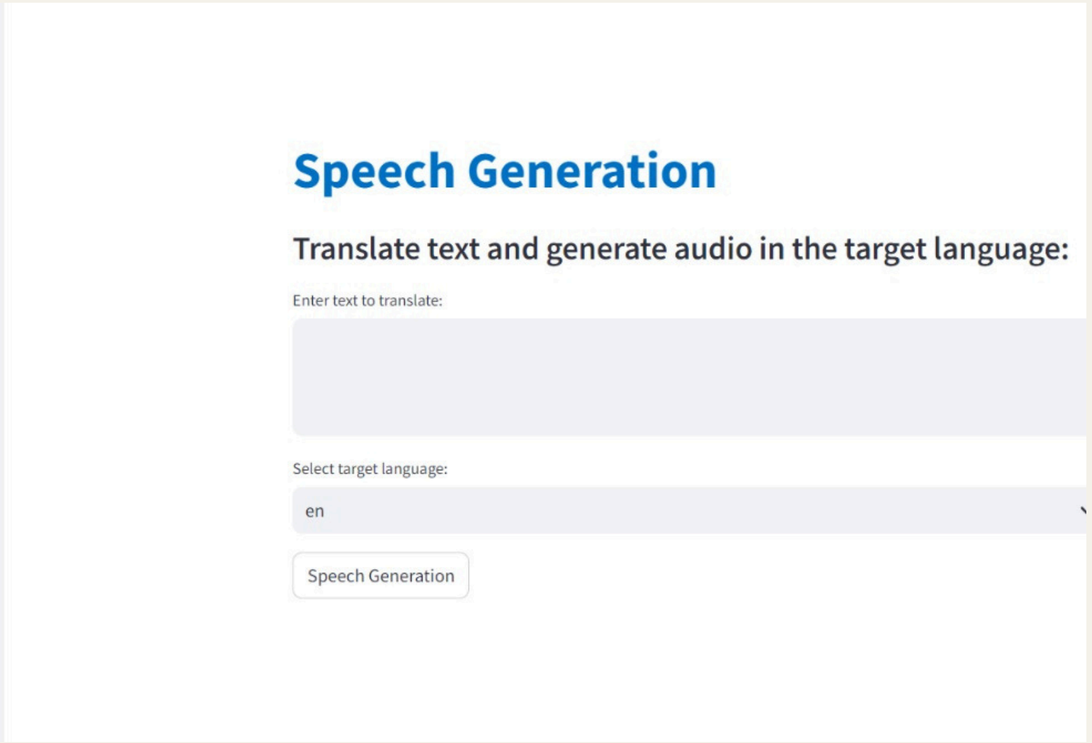
WORKFLOW

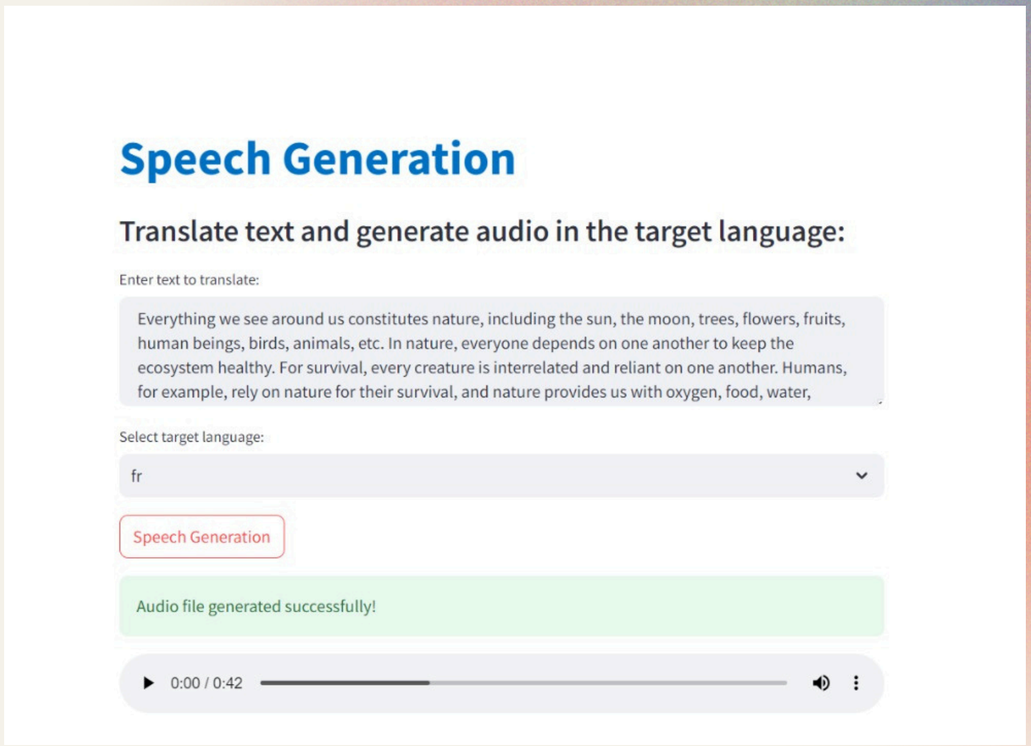
1. **Tokenization**: Convert input text into tokens.
2. **Positional Encoding**: Add positional information to the tokens.
3. **Transformer Encoder**: Process tokens to capture contextual relationships.
4. **Transformer Decoder:** Decode tokens sequentially to generate the translation.
5. **Output Generation:** Detokenize the generated tokens to produce the final translated text.



SPEECH GENERATION

- Converts text into natural-sounding speech with remarkable accuracy and clarity.
- Users can input text in any supported language and receive high-quality speech output.
- Supported by the gTTs library.



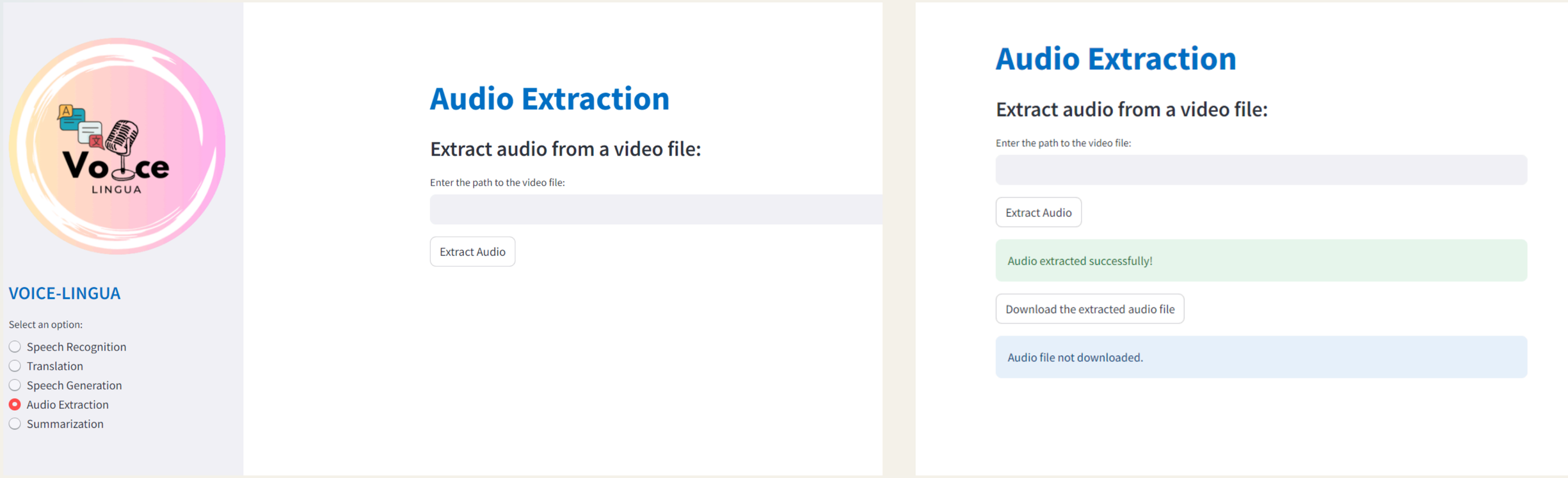


WORKFLOW

1. Text Translation: The code uses the GoogleTranslator from the googletrans library to translate the input text.
The translate method is called on the translator object, passing the input text as the argument.
The translated text is stored in the translated text variable.
The input text is translated into the target language using the Google Translator API. The source language is automatically detected, and the target language is specified by the user.
2. Audio Generation: The translated text is then used to generate an audio file using the Google Text-to-Speech (gTTS) API. The target language is specified to ensure the correct pronunciation and accent.
3. Audio File Saving: The generated audio file is saved with a specified filename.
4. Error Handling: If the target language is not valid, an error message is displayed.
5.Function Call: The `translate and generate audio` function is called with the input text, target language, and filename as arguments.
6.Output: The generated audio file is saved and can be played back.

AUDIO EXTRACTION

- Converts video files into high-quality audio.
- Supports various video file types , ensuring compatibility and ease of use.

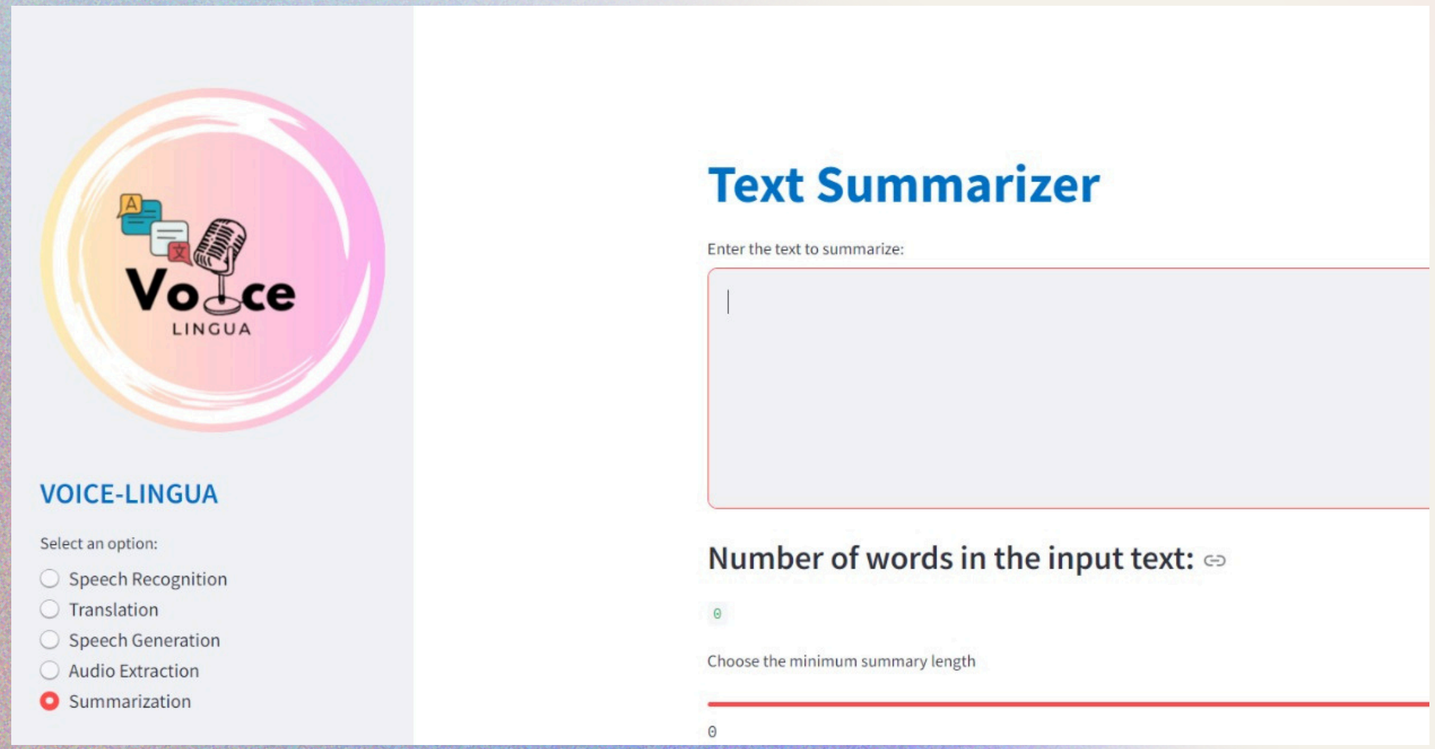


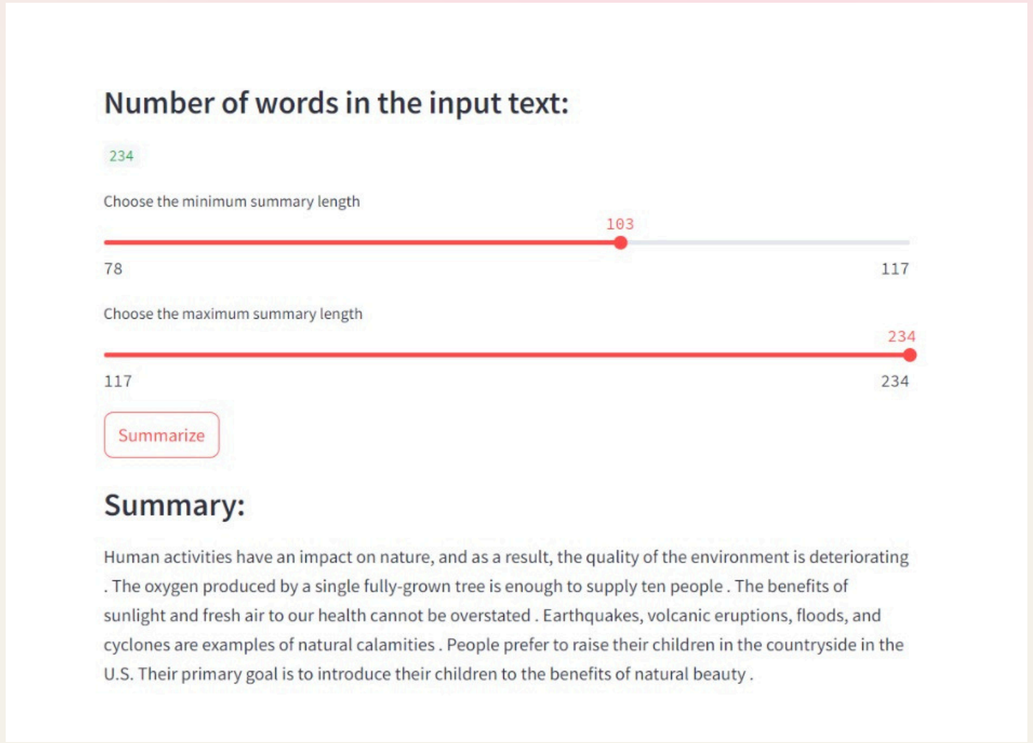
WORKFLOW

Import the MoviePy Module:
The code starts by importing the moviepy.editor module, which provides the necessary functions and classes for video editing.
Load the Video File:
The VideoFileClip class from the moviepy.editor module is used to load the video file.
The video file path is passed as an argument to the VideoFileClip function, which creates a VideoFileClip object representing the video.
Extract the Audio:
The audio attribute of the VideoFileClip object is accessed to extract the audio from the video.
This audio attribute returns an AudioFileClip object, which represents the audio component of the video.
Write the Audio to a File:
The write_audiofile method of the AudioFileClip object is used to save the extracted audio to a file.
The desired filename (e.g., "output.mp3") is passed as an argument to the write_audiofile method.

SUMMARIZATION

- Provides precise and coherent summaries of lengthy texts, reducing the content into customizable lengths.
- Users can enter any type of writing and get brief summaries that preserve important details.
- Model used : facebook/bart-large-cnn





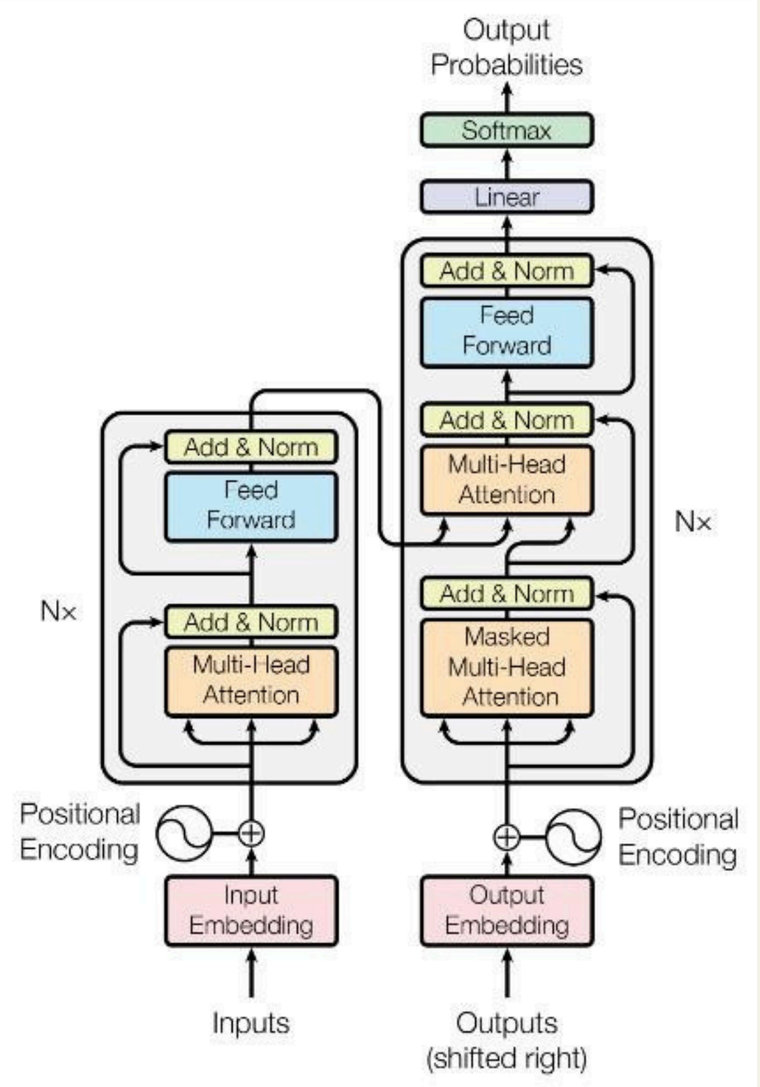
WORKFLOW

Summary Generation:

1. **Input Encoding:** the input text is tokenized and the encoder generates contextualized representation.
2. **Denoising Objective:** Training for robust representations with masked/corrupted inputs
3. **Decoding:** Generates summaries by predicting tokens based on encoded context.

Sentence and Word Selection:

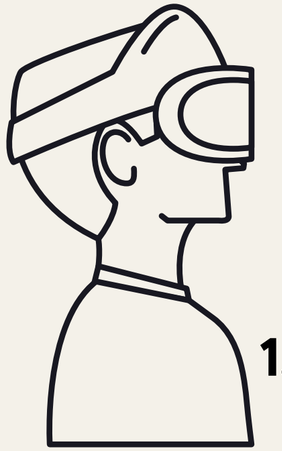
1. **Attention-Mechanism:** Self-attention and cross-attention focus on relevant parts of the input.
2. **Positional Encoding:** Maintains token order for coherent summaries.
3. **Beam Search:** Tracks multiple summaries and selects the best one.
4. **Length Penalty:** Ensures concise and to-the-point summaries.



USE CASE

VOICE LINGUA is applicable in various fields, including:

- ·**Medical**: Improving healthcare communication by transcribing doctor-patient interactions, translating medical documents, summarizing patient notes, and generating audio content for patient.
- **Education**: Facilitating transcription, translation, and summarization of study materials for students and educators.
- **Business**: Enhancing productivity with accurate meeting transcriptions, multilingual communication, and content summarization.
- **Content Creation**: Assisting creators with generating voiceovers, translating content for a global audience, and extracting audio from videos for podcasts.
- **News and Multimedia:** Supporting journalists and producers in transcribing interviews, translating reports, generating multilingual news segments, and repurposing video content.

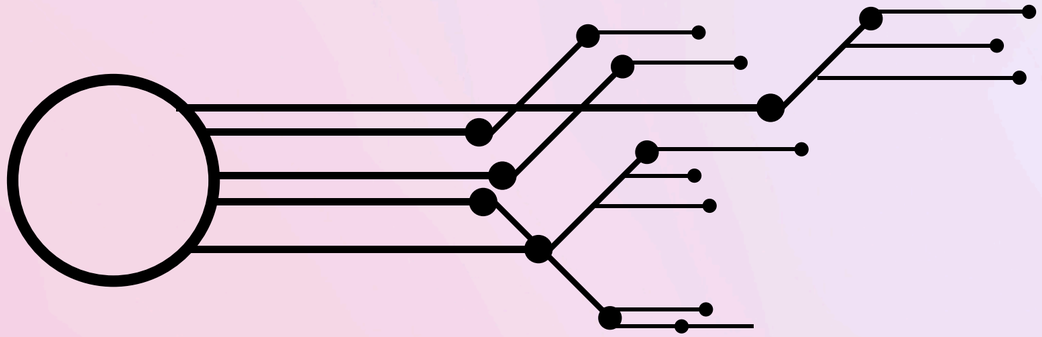


FUTURE WORK

1. **Improve Naturalness and Accuracy:** Speech Recognition: Improve speech recognition accuracy for a variety of accents and dialects, as well as in loud settings.

2. **Expansion of Language Support:** Increase the number of supported languages and dialects, focusing on underrepresented languages to make the system more inclusive.

3. **Real-time Processing:** Optimize the system for real-time processing, reducing latency in speech recognition, translation, and generation to support live interactions.



THANK-YOU