# Bhashika: Dialect-Based Text-to-Speech Model for Indic Languages

[**Github Link**: https://github.com/N-Deeps/Team2_Vocab_Vanguard/]
[**Vocab Vanguard:** Aindrila Majumder, Abhishek Dutt, N Deepika, Drishti Singh, Sarthak Sharma]

---

# Introduction

This report outlines two distinct processes aimed at enhancing the accessibility and usability of regional Hindi language content: the transcription of Hindi audio files using AI4Bharat's Automatic Speech Recognition (ASR) models and the generation of speech in a regional accent utilising the Toucan Text-to-Speech (TTS) model.

The primary objective of the transcription task is to convert audio recordings into text format, making spoken content searchable and editable. This can significantly benefit applications such as content creation, research, and accessibility for individuals who are deaf or hard of hearing.

On the other hand, the regional accent generation task focuses on providing a personalized speech synthesis experience by using user-provided voice samples as references. By translating given Hindi text into a Bihari accent, we aim to produce natural-sounding speech that reflects regional nuances. This can enhance user engagement in applications like virtual assistants, educational tools, and entertainment platforms that cater to regional audiences.

Together, these tasks contribute to a comprehensive approach in developing tools that respect linguistic diversity and improve interaction with technology in regional languages.

# Part 1: Transcription Using AI4Bharat Model

## Overview

The transcription process involves three key steps:

1. **Audio File Downloading**: Audio files are sourced from URLs provided in a CSV dataset, specifically designed for this purpose. Each entry in the dataset corresponds to a `.wav` audio file that requires transcription.
2. **Audio Transcription**: The downloaded audio files are transcribed using the AI4Bharat IndicConformer model, which is optimized for recognizing Hindi speech.
3. **Output Generation**: The transcriptions are stored in a new column called `transcript` within an output CSV file, ensuring each transcription is linked to its respective audio file.

### Requirements

To successfully execute the transcription process, the following prerequisites must be met:

- **Python Environment**: A compatible Python 3.x setup, such as Kaggle or Google Colab.
- **Required Libraries**: Essential libraries include `torch`, `nemo.collections.asr`, `pandas`, `requests`, and `soundfile`.
- **ASR Model**: The pre-trained IndicConformer model for Hindi transcription must be obtained.

### File Descriptions

The project includes several key files and directories:

- **Bihar_combined_file.csv**: This is the input dataset containing the URLs of the `.wav` audio files to be transcribed.
- **checkpoint.nemo**: This file contains the pre-trained AI4Bharat IndicConformer model, which is essential for performing automatic speech recognition on the audio data.
- **transcribed_dataset.csv**: This output file stores the transcriptions of the audio files, providing a direct correlation between each audio entry and its transcription.
- **Bihar_Dataset_wavFiles/**: This directory holds the downloaded `.wav` audio files, which are named according to their entries in the CSV file.

### Conclusion

The transcription process leverages advanced ASR technology to convert Hindi audio files into text format efficiently. By utilizing AI4Bharat's IndicConformer model, this project significantly enhances the accessibility of regional language audio content for various applications, including research, data analysis, and content creation.

---

# Part 2: Regional Accent Generation Using Toucan TTS Model

### Overview

This section details the process to generate speech in a regional accent using the Toucan TTS model. By providing a few-shot voice reference, the model adapts its output to match the user's voice characteristics while translating a given Hindi text into the specified dialect.

### Process Overview

The workflow consists of two main functions: `read_texts` for text-to-speech synthesis and `dialect_generation` for managing the overall dialect generation process.

**1. Text-to-Speech Synthesis**

The `read_texts` function performs the following tasks:

- **Model Initialization**: Initializes the Toucan TTS interface with the specified device (CPU or GPU) and model ID.
- **Language Setting**: Configures the language for the synthesis, defaulting to English but set to Hindi for this use case.
- **Speaker Reference**: Uses a user voice reference (speaker embedding) to adapt the generated speech to match the user's voice characteristics if provided.
- **Sentence Preparation**: Accepts either a single string or a list of strings, ensuring compatibility with the TTS model.
- **Speech Generation**: Synthesizes speech from the input text and saves it as a `.wav` file, applying various parameters to control duration scaling and energy variance.

**2. Dialect Generation**

The `dialect_generation` function orchestrates the process of generating speech in a Bihari accent:

- **Output Directory**: Ensures the existence of the output directory for storing audio files.
- **Input Text**: Specifies the Hindi sentence that will be translated into the Bihari accent.
- **Execution Device**: Determines whether to use a CUDA-enabled GPU or fall back to the CPU.
- **Speaker Reference**: Uses a user-provided voice sample for a few-shot prompt, allowing the model to mimic the accent and tone of the speaker.

## Execution Logic

The script is executed under the `if __name__ == '__main__':` block, which:

- **Checks Device Availability**: Identifies whether a CUDA-capable GPU is available for execution.
- **Sets Speaker Reference**: Specifies the path to the user's voice sample for accent adaptation.
- **Calls the Dialect Generation Function**: Initiates the process of generating the audio file in a Bihari accent.

# Future Work:

In future work, the focus will be on fine-tuning the Toucan TTS model using various regional voices and their corresponding transcripts obtained from the first step of the AI4 Bharat transcription process. This will involve the following key steps:

1. **Data Collection**: Gather a diverse set of regional voice samples and their transcriptions, ensuring representation from different dialects and accents.
2. **Model Fine-Tuning**: Utilize these collected voice samples to fine-tune the Toucan TTS model, enhancing its ability to generate speech in various regional dialects.
3. **Dialect Generation**: Once the model is fine-tuned, it will be tested with various text inputs to generate speech in the specified regional dialects. This will broaden the application of the TTS model, making it versatile for generating content across multiple regional languages.
4. **Evaluation and Feedback**: Collect user feedback on the generated speech to evaluate the effectiveness of the fine-tuning and make necessary adjustments to improve the model's accuracy and naturalness.

# Conclusion:

Both processes outlined in this report contribute significantly to the field of regional language processing and synthesis. The transcription of Hindi audio files allows for better accessibility and usability in content creation, while the Bihari accent generation offers personalized speech synthesis tailored to user preferences. Future efforts to fine-tune the Toucan TTS model for various regional voices will further enrich the diversity and applicability of speech synthesis technologies, enhancing communication and representation of regional dialects.

# References:

**1.https://github.com/DigitalPhonetics/IMS-Toucan**

**2.https://github.com/AI4Bharat/Indic-TTS**