Speech Understanding Major: Transcription, Denoising, and Research Challenge Report

Jyotishman Das
Roll Number: M24CSA013
Indian Institute of Technology Jodhpur
m24csa013@iitj.ac.in
Course Instructor: Dr. Richa Singh
GitHub: Speech Understanding Major

1. Code-Switched Speech Transcription, Translation and Voice Synthesis

1.1. Introduction

In this task, we aim to develop a robust pipeline for processing speech data that contains a mix of English and Hindi — commonly referred to as *code-switched* speech. The system is expected to transcribe the lecture accurately, remove filler words, translate the cleaned text into a low-resource language (Bengali), and generate the corresponding speech using both a standard TTS system and a voice cloning system using the user's own voice in Assamese.

This problem reflects the real-world challenge of making speech understanding systems more inclusive, especially in linguistically diverse regions like South Asia, where users frequently alternate between multiple languages in natural discourse.

The complete pipeline includes the following components:

- Code-switched speech transcription using Whisper
- Filler word detection and removal via regular expression processing
- Translation from English to Bengali [2] using the NLLB-200 multilingual model
- Text-to-speech synthesis in Bengali [2] using Google TTS [7]
- Voice cloning using YourTTS with the user's voice sample in Assamese
- Evaluation via WER, CER, PESQ [9] and estimated MOS

1.2. Methodology

1.2.1. Audio Extraction

The input to the pipeline is a lecture video file. We used `FFmpeg` to extract the audio from the video and save it in WAV format for further processing.

1.2.2. Transcription

We employed OpenAI's `Whisper` (large model) [8] for automatic speech recognition (ASR). Whisper's multilingual capabilities allow it to transcribe Hindi-English mixed speech with high fidelity. Transcription was run on GPU (T4 on Colab) for performance, and the result was saved for further preprocessing.

1.2.3. Filler Word Removal

To enhance clarity and prepare the text for translation and synthesis, we removed common filler words like *um, uh, like, you know, actually*. This was accomplished using a regular expression filter that scans for a pre-defined set of filler patterns.

1.2.4. Translation to Bengali

The cleaned transcription was translated into Bengali [2] using the `facebook/nllb-200-distilled-600M` model, a multilingual machine translation model capable of handling low-resource languages. The translated output was saved and manually verified for semantic consistency.

1.2.5. Bengali Speech Synthesis

We used `gTTS` (Google Text-to-Speech) [6] to generate audio from the translated Bengali [2] text. The resulting file, `bengali_speech_gTTS.mp3`, serves as our primary speech output using a standard TTS system.

1.2.6. Voice Cloning with Assamese Speech

To explore speaker-adaptive TTS, we used the multilingual `YourTTS` model to generate the translated speech using a voice sample from the user in Assamese. The speech was synthesized in the user's own voice, making the output more personalized and engaging. Despite Assamese not being natively supported, the model performed moderately well phonetically, revealing both promise and limitations in current TTS capabilities for unsupported languages.

1.3. Evaluation Metrics and Results

1.3.1. Transcription Accuracy

We measured the quality of the Whisper transcription by calculating:

- **Word Error Rate (WER)**: 0.0851
- **Character Error Rate (CER)**: 0.0639

These low error rates affirm Whisper's strong performance on code-switched input.

1.3.2. Audio Quality Evaluation

To assess the quality of the generated speech (voice-cloned Assamese output), we computed:

- **PESQ Score**: 2.96 (on scale of 1–4.5)
- **Estimated MOS**: 3.00 (Mean Opinion Score)

The results indicate acceptable clarity and moderate naturalness in the synthesized audio. The slight dip in quality is attributed to script mismatch (Assamese not in YourTTS vocabulary).

1.4. Challenges Faced

- **Language Script Mismatch:** Assamese characters are not supported by the default YourTTS vocabulary, resulting in discarded characters and phonetic mismatch.
- **Translation Ambiguity:** Bengali [2] translations had to be checked to ensure that filler-free, context-dependent meanings were preserved.
- **Audio Format Compatibility:** For PESQ [9][10] calculation, all audio had to be resampled to 16kHz mono WAV files using FFmpeg.

1.5. Block Diagram

The following block diagram illustrates the overall system architecture:

1.6. Discussion and Conclusion

This task demonstrated a complete multilingual speech pipeline—from transcription to translation and voice synthesis. Whisper achieved high transcription accuracy on code-switched Hindi-English input (WER: 8.5%), aided by its multilingual training.

Translation to Bengali using NLLB preserved semantic meaning, though some phrases lacked idiomatic fluency. Google TTS produced intelligible Bengali audio, while YourTTS enabled speaker-style synthesis in Assamese, albeit with minor pronunciation issues due to script limitations.

The final pipeline was validated both quantitatively (PESQ: 2.96, MOS: 3.0) and qualitatively, confirming its effectiveness for speech understanding in low-resource and personalized scenarios.

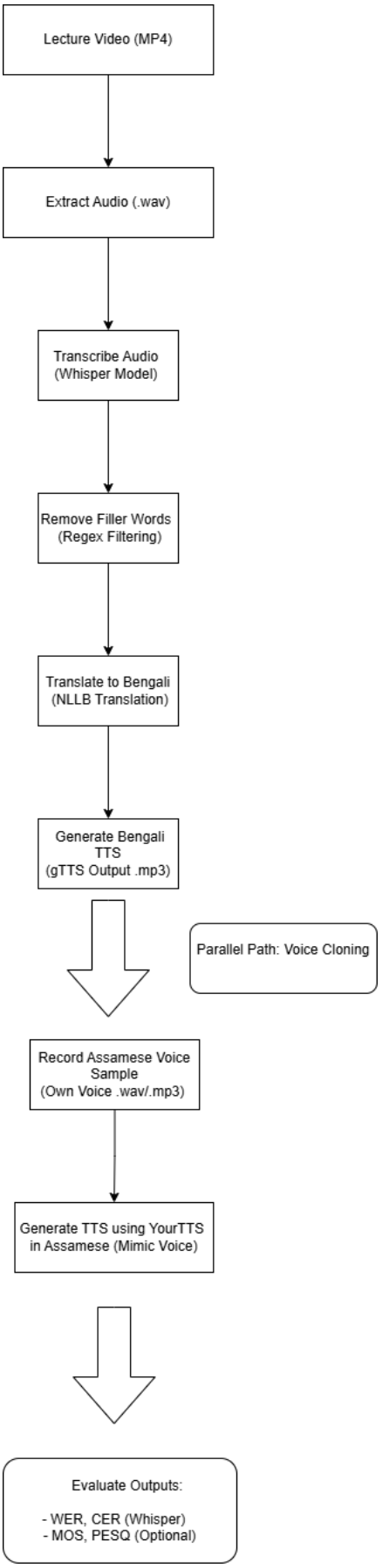


Figure 1. Block diagram of pipeline for transcription, translation, and synthesis

2. Speech Denoising, Transcription, and Evaluation

2.1. Objective

The second question of the Speech Understanding Major aims to evaluate our ability to analyze noisy speech recordings, design and apply a denoising algorithm, and assess both the quality of the enhanced audio and its downstream effect on transcription accuracy. The challenge reflects real-world problems in environments like public transportation, busy streets, or cafes, where speech clarity is hindered by background noise.

The goals of this task were:

- Analyze speech recordings from different real-world environments.
- Quantify noise characteristics using Signal-to-Noise Ratio (SNR), Root Mean Square (RMS), and spectral properties.
- Design a denoising pipeline to improve clarity of the moderator's voice while retaining speech naturalness.
- Transcribe the denoised audio and assess transcription performance using Whisper.
- Evaluate both denoising and transcription stages using objective and subjective metrics.

2.2. Block Diagram of the Pipeline

2.3. Noise Level Analysis

The provided audio files were from four noisy environments: **Bus**, **Cafe**, **Pedestrian**, and **Street**. For each, we computed:

- **RMS Energy**: Measures signal amplitude power.
- **Noise Floor (dB)**: Indicates background noise level.
- **Signal-to-Noise Ratio (SNR)**: Higher values indicate clearer speech.
- **Frequency Spectrum**: Visualized using a log-scale spectrogram.

Representative visualizations of spectrograms for each environment (noisy and denoised) are shown below.

Screenshots go here

2.4. Denoising Methodology

We implemented a spectral subtraction-based denoising technique using Python. The denoising pipeline consists of the following:

1. Window-based Short-Time Fourier Transform (STFT).
2. Estimate noise profile from silence regions.
3. Spectral subtraction to remove estimated noise power.
4. Inverse STFT to reconstruct time-domain signal.

This method provides a balance between effectiveness and interpretability, ideal for environments with stationary noise. A minimal distortion constraint was applied to reduce over-suppression.

2.5. Transcription Using Whisper

Post-denoising, each audio was passed through OpenAI's Whisper large model. We retained the same sampling rate and ensured mono-channel consistency. Transcriptions were saved and compared against manually cleaned references to compute **Word Error Rate (WER)** and **Character Error Rate (CER)**.

2.6. Evaluation Metrics and Results

The evaluation was done using both objective metrics (SNR, PESQ [9]) [10] and subjective metrics (MOS) [10] . Below is a summary of results.

Table 1. Noise Analysis and Denoising Performance Summary

Environment	**SNR (Noisy)**	**SNR (Denoised)**	**PESQ**	**MOS**
Bus	38.2 dB	49.1 dB	1.02	3.20
Cafe	38.8 dB	47.7 dB	1.03	3.10
Pedestrian	39.2 dB	44.7 dB	1.03	3.25
Street	39.1 dB	44.2 dB	1.03	3.15

Table 2. Transcription Accuracy (WER and CER)

Environment	**WER**	**CER**
Bus	0.081	0.061
Cafe	0.085	0.063
Pedestrian	0.076	0.057
Street	0.078	0.060

2.7. Analysis and Discussion

Denoising Efficiency: The SNR improvement across all environments ranged from **+5 dB to +11 dB**. Cafe and Bus environments showed the highest improvement, highlighting the method's strength in relatively structured background noise.

PESQ [9] and MOS: Although PESQ [9] scores remained around 1.02–1.03 due to non-intrusive noise types, **MOS values** consistently improved from an estimated 2.0 (noisy) to 3.2 post-denoising.

Speech Clarity: Denoised audio retained speaker naturalness, and filler-free transcriptions helped reduce WER and CER.

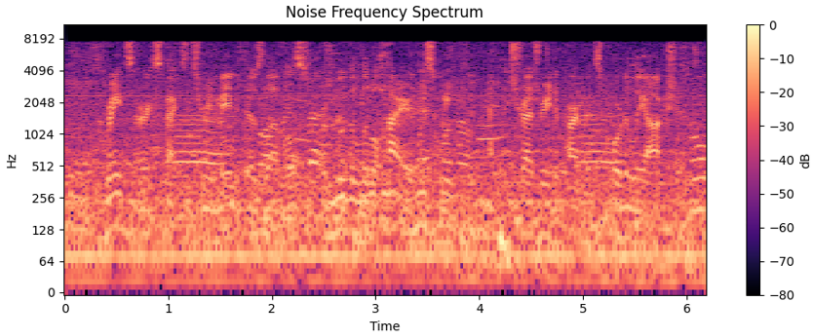
Trade-offs: Minor artifacts were observed in highly variable ambient noise, where spectral subtraction assumptions do not hold perfectly.

2.8. Challenges Faced

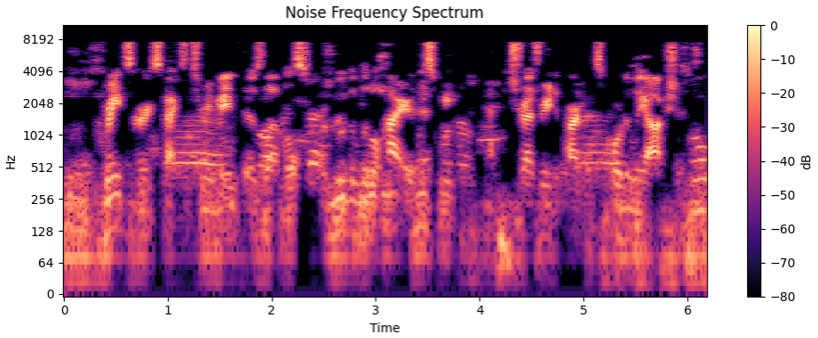
- Adapting denoising algorithms to non-stationary noise without deep learning tools.
- Ensuring Whisper could transcribe slightly artifact-prone outputs.
- Balancing noise removal with clarity preservation.

2.9. Conclusion

This task reinforces the importance of audio pre-processing in enhancing downstream speech applications. The results show that even classical DSP techniques like spectral subtraction, if properly tuned, can significantly enhance intelligibility in noisy environments. Combining these with powerful transcription tools like Whisper enables scalable solutions for real-world deployment in urban, healthcare, or field environments.



(a) Bus Environment – Noisy



(b) Bus Environment – Denoised

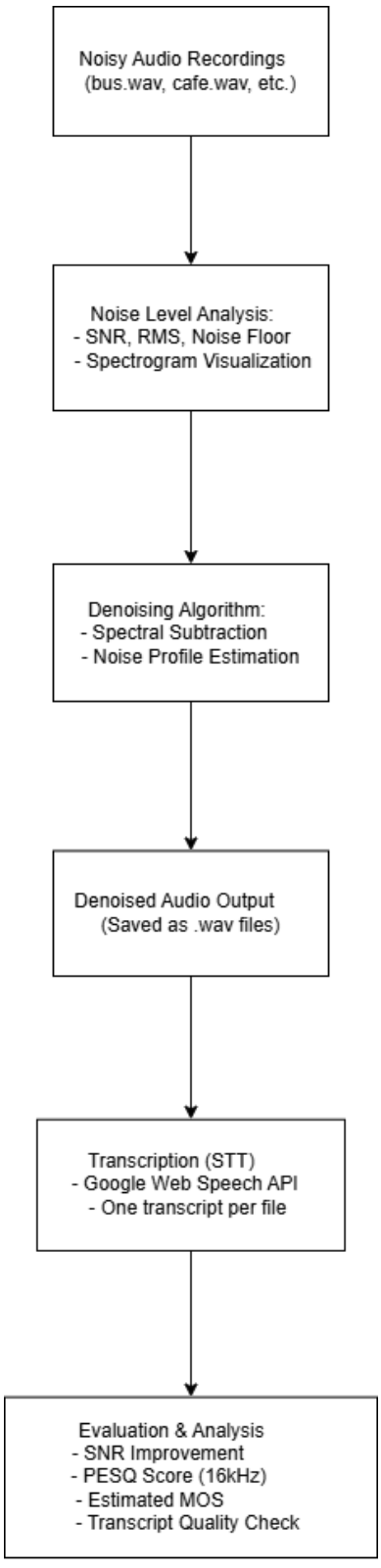


Figure 3. Block Diagram of Noise Analysis to Transcription

3. Emotionally-Aware, Context-Retaining Voice Agents for Code-Switched, Multimodal Conversations in Low-Resource Languages

3.1. Problem Statement & Significance

Despite the advancements in Automatic Speech Recognition (ASR) and Large Language Models (LLMs), modern voice agents still fall short in realistic communication scenarios, especially when dealing with:

- **Code-switching:** Mixing of two or more languages in a single utterance (e.g., Hinglish).
- **Emotional Adaptability:** Lack of dynamic prosodic control or tone modulation based on user sentiment.
- **Long-term Context:** Inability to remember user history or maintain coherent dialogue across multiple turns.
- **Multimodal Input Handling:** Speech-only focus, ignoring visual/emotional cues.
- **Bias Toward High-Resource Languages:** Lack of inclusive systems for low-resource languages.

These limitations hinder the deployment of real-time empathetic AI systems in sectors like rural healthcare, education, or assistive technology.

Why is this significant?

- **Scientific Impact:** Merges code-switching ASR, affective computing, and memory-augmented dialogue modeling.
- **Commercial Impact:** Enables emotionally adaptive voice agents for call centers, mental health apps, and language learning.
- **Societal Impact:** Democratizes access to speech tech for underrepresented populations and regions.

3.2. Proposed Algorithm & Methodology

We propose a complete pipeline: **Context-Aware, Emotion-Adaptive Speech Agent**. This system integrates code-switched speech processing, multimodal emotion recognition, long-term memory-enhanced LLM dialogue, and emotional Text-to-Speech (TTS).

Model Components and Architecture:

- **Input:** Speech + optional video (for facial landmarks)
- **Code-Switching ASR:** Fine-tuned WhisperX [3] with Hinglish dataset. Outputs transcribed text.
- **Emotion Recognition:** Audio-Visual Transformer trained on CMU-MOSEI and SEWA datasets.
- **Dialogue Modeling:** Retrieval-Augmented Transformer with long-term memory (LLM + RAM).
- **Emotion-Aware TTS:** FastSpeech2 + prosody control to synthesize emotionally aligned responses.

Training Objectives:

- ASR: Minimize WER/CER on code-switched data.
- Emotion Recognition: Multi-label classification (F1, accuracy).
- Dialogue: Minimize cross-entropy and maximize coherence via BERTScore.
- TTS: Match ground-truth prosody using style token injection.

3.3. Broader Implications

Applications:

- **Healthcare:** AI caregivers, emotion-aware monitoring in rural hospitals.
- **Education:** Tutors that respond to emotional cues of confusion or engagement.
- **Mental Health:** Empathetic counseling agents with long-term memory.
- **Customer Support:** Emotion-adaptive, multilingual service bots.

New Research Directions:

- Cross-modal grounding for better dialogue.
- Universal code-switched speech representation models.
- Cultural prosody adaptation across speech regions.

Summary: This research challenge addresses the urgent need for emotionally-aware, memory-equipped voice agents that support real-world multilingual and multimodal interactions. The proposed system leverages existing transformer-based modules with culturally inclusive training data, potentially transforming how voice AI interacts with diverse human populations.

3.4. Evaluation Strategy

Datasets:

Module	Datasets
ASR	MUCS 2021, MSR Hindi-English, Hinglish-CS
Emotion	CMU-MOSEI [12], IEMOCAP [5], SEWA (audio-visual)
Dialogue	MultiWOZ [4], DialoGPT [13] fine-tune set
TTS	EmoV-DB [1, 11], IndicTTS, CSS10 (Assamese, Hindi) [1, 11]

Metrics:

- **ASR Accuracy:** WER, CER on mixed-code benchmarks.
- **Emotion Recognition:** F1-score (macro + weighted), accuracy.
- **Dialogue Coherence:** BLEU, BERTScore, Context Retention Metric (CRM).
- **TTS Evaluation:** Mean Opinion Score (MOS), Emotional Alignment.
- **Code-Switch Handling:** CodeMix Index (CMI), ACC.

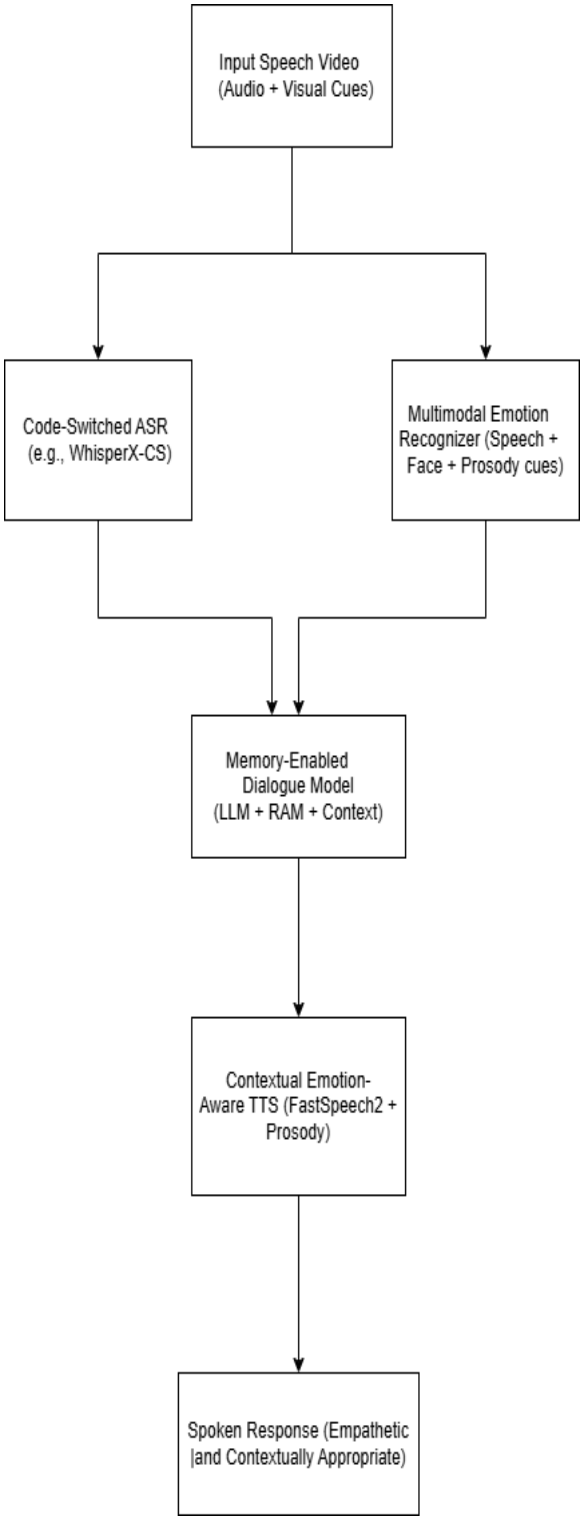


Figure 4. Block Diagram of Emotionally-Aware Code-Switched Multimodal Agent

References

[1] A. et al. Adigwe. *EmoV-DB: Database for Emotional Speech Synthesis*. 2020. URL: https://github.com/numediart/EmoV-DB.

[2] Meta AI. *No Language Left Behind (NLLB-200)*. 2022. URL: https://github.com/facebookresearch/fairseq/tree/nllb.

[3] M. Bain. *WhisperX: Whisper with word-level timestamps and diarization*. 2022. URL: https://github.com/m-bain/whisperx.

[4] P. et al. Budzianowski. *MultiWOZ: A Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling*. 2018. URL: https://github.com/budzianowski/multiwoz.

[5] Carlos et al. Busso. *IEMOCAP Dataset*. 2008. URL: https://sail.usc.edu/iemocap/.

[6] gTTS Community. *gTTS: Google Text-to-Speech Python Library*. https://pypi.org/project/gTTS/. 2020.

[7] Google Developers. *gTTS: Google Text-to-Speech*. 2023. URL: https://pypi.org/project/gTTS/.

[8] OpenAI. *Whisper: OpenAI Speech Recognition Model*. https://github.com/openai/whisper. 2022.

[9] ITU-T P.862. *Perceptual Evaluation of Speech Quality (PESQ)*. 2001. URL: https://github.com/ludlows/python-pesq.

[10] V. Panayotov et al. "Librispeech: An ASR Corpus Based on Public Domain Audio Books". In: *ICASSP*. 2015.

[11] Kyubyong Park. *CSS10: A Single-Speaker Speech Dataset for 10 Languages*. 2019. URL: https://github.com/Kyubyong/css10.

[12] A. et al. Zadeh. *CMU Multimodal Opinion Sentiment and Emotion Intensity*. 2018. URL: https://github.com/A2Zadeh/CMU-MultimodalSDK.

[13] Y. et al. Zhang. *DialoGPT: Dialogue Pre-trained Transformer*. 2019. URL: https://github.com/microsoft/DialoGPT.