From LLM Prompts to Acoustic Baselines: A Scalable Pipeline for Under-Resourced Disfluent Code-Mixed Speech

Anuran Mitra*, **Anirvan Chakravarty***, **Tapabrata Mondal***, **Sivaji Bandyopadhyay***
*Jadavpur University, Kolkata, India
{anuranm3, anirvanchakravarty39, tapabratamondal, sivaji.cse.ju}@gmail.com

Abstract

Spontaneous speech in multilingual communities often involves rapid code-mixing (CM) and natural disfluencies, yet such patterns are rarely reflected in available training data for under-resourced languages. This gap limits the development of robust automatic speech recognition (ASR) systems. To address this, we introduce BEHE-CMDisfl, a fully synthetic Bengali–English and Hindi–English disfluent code-mixed speech corpus generated through a controlled Large Language Model (LLM) and Text-to-Speech (TTS) pipeline. The dataset explicitly incorporates conversational phenomena such as filled pauses, repetitions, and restarts. We evaluate its usefulness under two ASR settings. In a micro-resource scenario ($\sim$1.3 hours), a GMM-HMM Kaldi baseline achieved a 37.74% Word Error Rate (WER) after phonetic normalization to reduce transliteration inconsistencies, and successfully retained disfluency markers in decoding. We also examined adaptation of a modern foundation model. In zero-shot testing, openai/whisper-small failed on the code-mixed speech due to severe hallucinations and looping behavior. After applying parameter-efficient fine-tuning (LoRA) for 1,000 steps, the model stabilized, reduced insertion errors, captured rapid language switching more reliably, and achieved a WER of 21.37%. These findings show that synthetic data combined with efficient fine-tuning offers a practical path for ASR development in complex low-resource disfluent CM settings.

Keywords: code-mixing (CM), disfluencies, automatic speech recognition (ASR), large language models (LLMs), text to speech (TTS), word error rate (WER)

1. Introduction

Multilingual communication is a defining feature of diverse linguistic landscapes, prominently observed in regions like South Asia. In daily interactions, speakers frequently blend languages such as Bengali, Hindi, and English within a single utterance, a phenomenon known as code-mixing or code-switching (Auer, 2013; Moyer, 2002). ***Code-mixing (CM)*** refers to the intra-sentential mixing of linguistic units (morphemes, words, or phrases) from two grammatical systems, whereas ***code-switching (CS)*** denotes inter-sentential alternation. While technically distinct, both terms are often used interchangeably in practice (Kim, 2006). This linguistic blending is highly prevalent in multilingual societies such as India (Harya, 2018), presenting a complex ”cold-start” challenge for speech technologies attempting to process rapid cross-lingual alternations.

Alongside language blending, natural speech is characterized by disfluencies such as filled pauses (“uh,” “um”), repetitions, and restarts, which arise from cognitive planning during speech production (Shriberg, 2001; Adell et al., 2006). However, most existing corpora for under-resourced Indic languages rely entirely on scripted, fluent recordings (Saranya et al., 2025; Diwan et al., 2021). Furthermore, contemporary ASR systems typically treat disfluencies as acoustic noise to be filtered out (Kundu et al., 2022; Zayats et al., 2016). This sanitization strategy limits their utility in conversational AI and behavioral modeling, where hesitation markers carry critical semantic weight. While recent text-based code-mixed resources exist, such as BnSentMix (Alam et al., 2024), they inherently lack the acoustic dimension necessary for speech modeling. Speech-level corpora capturing both code-mixing and spontaneous disfluencies remain critically scarce, leaving this domain largely unexplored (Sitaram et al., 2019).

Collecting naturalistic disfluent code-mixed speech is prohibitively expensive for micro-resource scenarios (e.g., datasets comprising roughly one hour of audio). Modern end-to-end ASR architectures generally demand hundreds of hours to generalize (Hannun et al., 2014; Watanabe et al., 2018), often failing catastrophically in cold-start settings on complex dialects. In contrast, Gaussian Mixture Model–Hidden Markov Model (GMM-HMM) systems remain practical and interpretable for severely data-constrained environments (Besacier et al., 2014; Mohri et al., 2008). To bridge this gap, we propose a scalable synthetic methodology. We construct **BEHE-CMDisfl**, generating disfluent Bengali–English and Hindi–English code-mixed utterances using OpenAI’s **ChatGPT**[1] and synthesizing them with

[1] https://chatgpt.com/

Proceedings of SIGUL 2026 Joint Workshop with ELE, EURALI, and DCLRL @ LREC 2026, pages 222–232
11–12 May 2026.

Indic Parler TTS[2] (Lacombe et al., 2024; Lyth and King, 2024). Rather than simply releasing a dataset, we validate its utility across two distinct ASR paradigms. First, we establish an efficient GMM-HMM baselines using Kaldi (Povey et al., 2011) to demonstrate the paramount importance of lexicon consistency over model size. Second, we demonstrate how a modern ASR model like Whisper (Radford et al., 2023) which initially struggles with this complex data can be efficiently adapted using a lightweight fine-tuning method called LoRA (Hu et al., 2022). This study thus provides both a disfluency-aware corpus and a proven recipe for bootstrapping under-resourced conversational ASR.

1.1. Our Contribution

- We develop **BEHE-CMDisfl**, a synthetic code-mixed Bengali–English (BE) and Hindi–English (HE) speech dataset generated through a controlled LLM–TTS pipeline that explicitly integrates filled pauses, repetitions, and restarts.
- We establish reproducible GMM-HMM baselines for micro-resource settings using Kaldi, demonstrating that transliteration normalization and lexicon consistency yield a robust WER of 37.74%.
- We show that acoustic models can retain disfluency markers in decoding outputs when explicitly modeled, supporting their inclusion in conversational ASR.
- We demonstrate that targeted adaptation of modern foundation models via LoRA resolves catastrophic zero-shot failures, reducing WER to 21.37% and accurately capturing rapid language switching in disfluent synthetic audio.

2. Related Works

Research on code-mixed language processing often targets high-resource scenarios, but multilingual environments like South Asia present severe ”cold-start” challenges for speech technologies. Early work primarily focused on textual datasets for *Hinglish*, *Banglish*, and *Tamil–English*, targeting sentiment analysis and sequence labeling tasks (Patra et al., 2018; Singh et al., 2018; Alam et al., 2024; Nishat Raihan et al., 2023). On the speech side, datasets such as *MUCS 2021* (Diwan et al., 2021) and *Prabhupadavani* (Sandhan et al., 2022) support multilingual ASR and speech translation. More recent efforts include *MediBeng* (Ghosh, 2025) and *Switchlingua* (Xie et al., 2025). While these resources advance code-switched speech research, they generally curate fluent speech and filter out spontaneous irregularities. Conversely, disfluency research is well established in English-centric corpora like *Switchboard* (Godfrey and Holliman, 1997) and *FluencyBank* (Romana et al., 2024). Although recent studies explore automatic disfluency detection (Amann et al., 2024) and synthetic augmentation (Bhat et al., 2023), speech-level corpora that jointly capture code-mixing and acoustic disfluencies for under-resourced languages remain critically scarce (Saranya et al., 2025). To overcome this data bottleneck, recent methodologies leverage neural TTS and LLMs for scalable synthetic augmentation (Thai et al., 2019; Yeo et al., 2026). For instance, *MixFluent* (Paul et al., 2025) successfully demonstrated the feasibility of generating code-mixed disfluent text using LLMs. However, it remains strictly limited to the textual modality. While MixFluent provides valuable linguistic benchmarks, its lack of acoustic realization renders it unsuitable for training end-to-end speech systems or modeling the prosodic features of disfluency. Our dataset, *BEHE-CMDisfl*, directly extends this foundational research by bridging the gap between text and speech. We not only adapt the generation methodology for broader language coverage incorporating both Bengali–English and Hindi–English pairs, but crucially, we project these textual disfluencies into the acoustic domain using the Indic Parler TTS framework (Lacombe et al., 2024; Lyth and King, 2024).

In terms of acoustic modeling, classical frameworks like Kaldi remain highly effective for micro-resource and code-switched settings (Kullmann, 2016; Yılmaz et al., 2016). While modern deep learning architectures and foundation models achieve state-of-the-art results in high-resource scenarios (Panayotov et al., 2015; Bu et al., 2017; Radford et al., 2023), their effectiveness in micro-resource conditions remains severely limited, often suffering from catastrophic hallucinations or overfitting (Dar and Pushparaj, 2026; Dhasmana et al., 2026). Addressing this requires parameter-efficient fine-tuning (PEFT) techniques, such as LoRA (Hu et al., 2022), which have emerged as vital tools for adapting massive models to niche, low-resource domains. Ultimately, existing literature either emphasizes code-mixing without disfluency (Pandey et al., 2018), text without audio, or relies on data volumes unavailable to marginalized languages. Our study fills this gap by integrating explicit synthetic disfluency modeling with both classical and foundation-model ASR pipelines in a true code-mixed micro-resource setting.

[2] https://huggingface.co/ai4bharat/indic-parler-tts

3. Synthetic Data Generation Pipeline

Rather than treating the data scarcity as a hurdle, we treat it as a controlled generation problem. The *BEHE-CMDisfl* dataset was constructed through a modular, two-stage synthetic pipeline that combines (i) *ChatGPT* based generation of code-mixed disfluent text and (ii) *Indic Parler TTS* synthesis to produce the corresponding speech. Bengali and Hindi are treated as matrix languages, with English as the embedded language. By simply adjusting the language constraints in the prompt and swapping in an appropriate target-language TTS model, researchers can readily adapt this exact methodology to bootstrap resources for other severely under-resourced code-mixed pairs. The design prioritizes controlled disfluency injection, realistic code-switching behavior, and reproducibility of the generation process.

3.1. Text Corpus Design and Preparation

To ensure a generalized domain, we curated reference text prompts covering some of the major conversational contexts:

- Daily life and informal communication (e.g., greetings, personal anecdotes, job oriented discussions etc.)
- Task-oriented dialogs (e.g., customer–service interactions)
- Education and classroom discussions among teachers and students
- Healthcare and teleconsultation contexts
- Social media–style opinions, commentary and discussions about sports and entertainment

This step is a way to explore the intersection of code-mixing and disfluency in bilingual speech and text, with a focus on understanding how LLMs handle code-mixed disfluent utterances. One of the primary objectives was to explore LLMs' capability to generate code-mixed disfluent sentences and to address the lack of high-quality code-mixed disfluent corpora, particularly for Indic languages.

3.2. LLM-Driven Text Generation

We designed our prompts as structured templates rather than open-ended requests. In our experience, without clear instructions, LLMs often produce overly formal and fluent text that lacks the natural irregularities of real conversation. To avoid this, we added explicit constraints on speaker roles, conversational setting, code-mixing level, and especially the types and approximate frequency of disfluencies. These guidelines helped generate transcripts that include pauses, repetitions, and self-corrections in a controlled yet natural way, making them more suitable for training and evaluating disfluency-aware ASR systems. To generate real and natural code-mixed disfluent text, we used a LLM, specifically, ChatGPT using zero shot and few shot prompting techniques. A typical prompt used to generate a BE-CM disfluent textual conversational data is shown below:

> You are a dialog generation assistant. Generate realistic, informal conversations between multiple speakers in a South Asian context (Bengali-English), incorporating the following instructions:
>
> 1. Speaker Roles and Identity
>
> - Each line should start with the speaker's name followed by a colon, e.g., "Arjun: ..."
> - Maintain consistent personality for each speaker:
>
> - Speaker A: friendly, informal, slightly humorous
> - Speaker B: polite, thoughtful, sometimes hesitant
>
> - Include a variety of speakers with different age, gender, and occupation backgrounds.
>
> 2. Code-Mixing Requirements
>
> - Blend Bengali with English naturally within sentences.
> - Code-mixing should occur at the word level, phrase level, or mid-sentence.
> - Ensure frequent switching for realism, but do not overuse English.
>
> 3. Disfluency Requirements
>
> - Insert natural disfluencies like:
>
> - Filled pauses: "uh", "umm", "মানে", "আচ্ছা", "you know", "hmm", "err", etc.
> - Repetitions: repeating words or phrases
> - Hesitations: restarting sentences, partial words
>
> - Include disfluencies in roughly 15–25% of utterances.
> - Disfluencies should appear randomly but contextually plausible.
>
> 4. Conversation Context

- Base dialogs on everyday scenarios:
  - Catching up with friends
  - Work or office discussions
  - Academic or student-related interactions
  - Social plans or small talk about sports, tours, etc.
- Keep conversations casual and natural.

5. Output Formatting
   - Format each utterance as: SpeakerName: <utterance>
   - Avoid narration or meta-comments.
   - Maintain proper punctuation.
   - Use Unicode Bengali characters where applicable.
   - Do not translate English words unnecessarily; retain natural code-mixing.
6. Length
   - Each conversation should contain 15-25 lines of dialog.

Example Instruction:
"Generate a casual conversation between two friends, Kabir and Aditi, where they catch up about work and weekend plans. Include filled pauses and spontaneous repetitions. Use Bengali-English code-mixing naturally."

We used this structured prompt in both zero-shot and few-shot settings, adding examples to stabilize speaker consistency and turn-taking. Explicit constraints on roles, code-mixing, and disfluency show a balanced structural control with natural flow. We performed no manual rewriting beyond basic formatting checks (e.g., script consistency and structural validation). All generated dialogs are available on this repository[3]. An example of the resulting Bengali-English code-mixed dialogue, featuring the injected disfluent markers and language switching generated using the LLM, is shown below.

Arjun: Hello? এই... তু তু...- তুই শুনছিস তো? কী যে network problem, একদম..... উফ.....!
Kabir: হ্যাঁ হ্যাঁ, এখন clear পাচ্ছি! কী হলো, এত excited লাগছে?
Arjun: আরে, শোন! মানে... তুই তো বলছিলি না যে একটা ছুটি পেলে ঘুরতে যাবি?
Kabir: হ্যাঁ রে, but... umm, এখন তো plan cancel-ই হয়ে গেল মনে হয়। মানে, এরকম sudden ছুটি পাবো মনে করি নি।
Arjun: আরে, cancel কেনো? এই দেখি, আমার-ও Friday off আছে, তাহলেই একটা short trip plan করতে পারি, কী বলিস?
Kabir: উহ... wait, wait, তুই কি serious? মানে, plan করবো তো ঠিক আছে, but... umm, last moment-এ যদি কোনো problem হয়?
Arjun: আরে না রে! এই... actually, একটা কথা বলছি, তুই... তুই তো দার্জিলিং-এর কথা বলছিলি আগে?
Kabir: Umm... হ্যাঁ, but দার্জিলিং-এ তো এখন season! মানে... booking পাওয়া যাবে?
Arjun: হ্যাঁ.... রে....! উহ...., wait, just একটা minute, হ্যাঁ, একটা hotel আছে যেখানে availability আছে মনে হয় But budget তো... wait, wait, তুই কত রাখার plan করছিলি?
Kabir: উম... budget? মানে, কিছু fixed ছিল না, but...but around Five K পড়বে না একজনের?
Arjun: উফ...উফ..... নানা....না! ওতো না। যদি train-এ যাই..... আর... umm, hotel-টা ঠিকমতো দেখতে পারি, তাহলেই চার-এ manage হয়ে যাবে!
Kabir: Hmm... এই, but... উম, একটা কথা, তুই parents-এর সাথে কথা বলেছিস তো? না হলে আবার last moment-এ...
Arjun: আরে, বলবো বলবো! এখন তো একদম confirm করছি না, যদি তুই হ্যাঁ বলিস তাহলে বলবো!
Kabir: হ্যাঁ, ঠিক আছে,... ঠিক আছে,... একটা কাজ কর, তুই একবার hotel-এ call করে confirm কর, আর ticket availability check করে দেখ!
Arjun: হ্যাঁ, একদম ঠিক! Umm... আচ্ছা, এখন phone-টা রাখি, তারপর details নিয়ে আবার call করছি okay?
Kabir: হ্যাঁ, ঠিক আছে, জলদি update দে!
Arjun: হ্যাঁ হ্যাঁ, রাখছি এখন! Bye!
Kabir: Bye!

3.3. Speech Synthesis with Indic Parler TTS

The generated dialogs were converted into speech using Indic Parler TTS developed by AI4Bharat[4], a multilingual neural TTS framework that supports 22 Indic languages. The model was selected due to its support for Indic scripts and its ability to process mixed-script inputs, which is essential for Bengali–English and Hindi–English code-mixed text. Unlike generic engines that filter out noise, Indic Parler as seen in (Sankar et al., 2025) accurately reproduces the fine-grained hesitations and conversational markers that we need for this study. Speaker profiles were defined through textual descriptions and supplied to the TTS model

[3] https://github.com/anonrpd/BEHE-CMDisf/tree/main/text_data

[4] https://ai4bharat.iitm.ac.in/

to simulate inter-speaker variability across dialog turns. This allowed consistent voice characteristics within a conversation while preserving multi-speaker structure.

Each generated text files (as described in the previous section) was processed into speech according to Algorithm 1. Utterances were segmented into chunks of at most 25 words prior to synthesis. This chunking strategy was adopted to improve waveform stability and prevent degradation in longer inputs. The synthesized segments were then concatenated to form a single audio file per dialog. No manual post-editing was performed on the generated waveforms beyond structural validation and file consistency checks. The generated speech data are available on this repository[5].

3.4. Statistical Analysis of the Dataset

To verify that our generation pipeline successfully adhered to the injected constraints, we conducted a detailed statistical analysis of both the textual and acoustic outputs. The objective here is not simply to describe the corpus, but to quantitatively show that our LLM-TTS methodology provides fine-grained, predictable control over complex, spontaneous speech phenomena. We evaluated the dataset across several metrics to ensure a balance between structural realism and acoustic stability.

3.4.1. Evaluation Metrics

Some of the evaluation metrics used are:

1. **Mean Filler Percentage (%)**: Proportion of tokens in an utterance that are filler words (e.g., *uh*, *um*, *ah*).
2. **Mean Repetition Percentage (%)**: Percentage of tokens that are repeated immediately or within a short span.
3. **Mean Restart Percentage (%)**: Fraction of utterances exhibiting restart phenomena
4. **Mean Code-Switch Percentage (%):** Ratio of words from the embedded (non-matrix) language to total words, reflecting the degree of code-switching.
5. **Speech Ratio:** Proportion of the total duration of the audio that contains active speech segments (excluding silence).
6. **Speech Ratio Range:** Difference between the maximum and minimum speech ratio across utterances in a corpus.

[5] https://github.com/anonrpd/BEHE-CMDisf/tree/main/speech_data

Algorithm 1 Text-to-Speech Generation for Disfluent Bengali-English/Hindi-English Code-Mixed dialogs

Require: `input_folder`: directory of dialog `.txt` files; `speaker_profiles`: speaker descriptions
Ensure: `output_folder`: generated `.wav` audio files

for each file F in `input_folder` **do**
  Initialize `audio_segments` list
  Read all lines from F
  for each line in F (up to MAX_LINES) **do**
    Parse line $\rightarrow$ `speaker_name`, `utterance_text`
    if `speaker_name` not in `speaker_profiles` **then**
      skip line
    end if
    `speaker_desc` = `speaker_profiles[speaker_name]`
    Split `utterance_text` into chunks ≤ 25 words
    for each chunk **do**
      Generate audio using `model.generate(tokenized_speaker_desc, tokenized_chunk)` $\rightarrow$ `audio_waveform`
      Append `audio_waveform` to `audio_segments`
    end for
  end for
  if `audio_segments` not empty **then**
    Concatenate `audio_segments` $\rightarrow$ `final_audio`
    Save `final_audio` as `.wav` in `output_folder`
  end if
end for

3.4.2. Text Data Analysis

The corpus includes 77 Bengali–English (BE-CM) and 40 Hindi–English (HE-CM) code mixed disfluent text dialog files, covering a range of conversational contexts. As shown in Table 1, Bengali–English dialogs contain an average of $\sim$13.6 words per utterance, while Hindi–English dialogs average $\sim$11.8 words. These values indicate moderately sized conversational turns across both subsets, with comparable structural organization.

Disfluency markers are present at controlled levels throughout the corpus. Filled pauses account for approximately 0.24–0.25% of tokens in both sets. Restart phenomena occur in roughly 1.22–1.35% of utterances, while repetition frequency is higher in Bengali–English (0.37%) compared to Hindi–English (0.14%). These values suggest that disfluencies are distributed across dialogues with-

Feature	BE–CM Text Data	HE–CM Text Data
Total Files	77	40
Mean Utterances per File	31.62	44.32
Mean Words/File	392.42	520.05
Mean Words per Utterances	13.59	11.77
Mean Filler %	0.25	0.24
Mean Repetition %	0.37	0.14
Mean Restart %	1.35	1.22
Mean CS %	30.94	34.46

Table 1: Summary statistics of the Bengali–English and Hindi–English disfluent code-mixed text corpora.

Feature	BE-CM Speech Data	HE-CM Speech Data
Total recordings	77	40
Duration range (s)	108–304	136–329
Average duration (s)	195.8	224.4
Mean speech ratio	0.90	0.94
Speech ratio range	0.82–0.95	0.91–0.97
Estimated words per file	438.5 ± 110	525.4 ± 120
Speech rate (words/s)	2.26 ± 0.07	2.34 ± 0.05

Table 2: Comparative summary of Bengali-English and Hindi-English speech corpora.

out dominating overall utterance structure. Code-switching constitutes a substantial portion of the corpus. On average, 30.94% of tokens in the Bengali–English dataset and 34.46% in the Hindi–English dataset originate from the embedded language. This indicates consistent bilingual alternation within utterances across both language pairs. Beyond mean statistics, variation across files is observable in the distribution of switching behavior. Figure 1 shows that the Bengali–English subset exhibits a broader switching range (4.68%–62.93%), whereas the Hindi–English subset demonstrates a comparatively narrower distribution (22.44%–55.4%). This dispersion suggests that code-mixing density varies across dialogue contexts rather than remaining uniform throughout the corpus. This contrast highlights the differential nature of the generated text, likely reflecting the underlying distribution of the LLM's training data for these specific language pairs.

3.4.3. Speech Data Analysis

The Bengali–English code mixed (BE-CM) speech corpus comprises 77 recordings with mean duration of 195.8 s, while the Hindi–English code mixed (HE-CM) set contains 40 recordings with mean duration of 224.4 s. Table 2 clearly shows that both subsets exhibit high speech ratios (>0.90), indicating minimal silence and stable conversational flow. Average speech rates fall within 2.26–2.34 words per second, consistent with natural bilingual conversational tempo. The relatively low standard deviation in speech rate suggests uniform articulation speed across synthesized speakers.

The plot shown in figure 2 shows the scatter plot of *Duration vs Estimated Words* that complements this by showing a consistent relationship between speech duration and word output for both the code-mixed disfluent datasets, with regression trends highlighting differences in speaking rate and verbosity. Together, these results indicate that our dataset maintains controlled disfluency injection at the textual level, stable acoustic realization at the speech level, and meaningful variation across the two code-mixed language pairs. This combination makes the dataset appropriate for low-resource ASR benchmarking and exploratory studies on disfluency-aware modeling.

4. Experimental Validation

To assess the utility of the BEHE-CMDisfl corpus, we evaluated it across two distinct ASR paradigms. First, we established a classical statistical baseline to determine if a model could learn to decode code-mixed speech and preserve disfluency markers under severe data constraints. Second, we studied the dataset's efficacy in adapting a modern foundation model to this complex, out-of-domain linguistic setting. We selected a 1.3-hour subset of the BE-CM data for this study to rigorously evaluate the pipeline's effectiveness under the severe data constraints typical of cold-start scenarios.

4.1. Kaldi GMM-HMM Acoustic Modeling

We utilized a GMM-HMM pipeline, which remains a robust choice for micro-resource regimes where deep learning models often fail to generalize. While the LLM generates transcripts in mixed Bengali-English Unicode as shown above previously, these were converted into a standardized Romanized script (e.g., 'tui' for তুই) for the Kaldi pipeline. This step was crucial to ensure phonetic consistency and reduce transliteration noise during acoustic modeling . The dataset was partitioned into a training set (12,006 words) and a testing set (1,236 words), maintaining an approx-

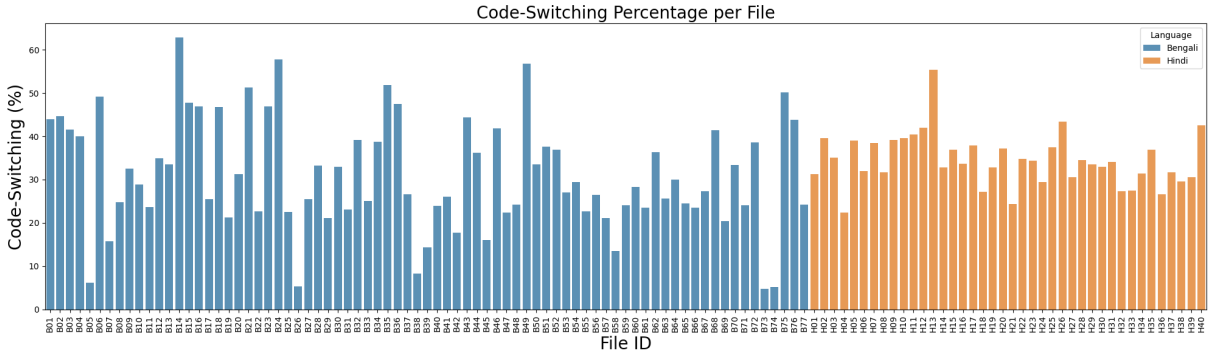

Figure 1: Code-Mixing Percentage Per Text File for Both the Languages.

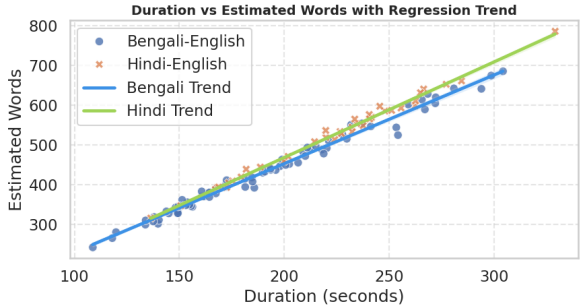

Figure 2: Relationship between speech duration and estimated word count for BE and HE datasets.

imate 90/10 split. We employed a standard feature extraction pipeline (MFCCs + deltas + delta-deltas) and trained a progression of Monophone, Triphone, and LDA-MLLT models. Regarding disfluencies, while filled pauses and repetitions consist of full lexical units, partial words (restarts) were handled at the lexicon level. Truncated words were mapped to a dedicated <UNK> or spoken-noise token to ensure the model penalizes the error without failing the decoding graph, preserving the acoustic evidence of the restart.

4.1.1. Results and Impact of Normalization

We observed that rigorous text normalization was essential at this scale. Initial experiments with raw, inconsistent transliterations (e.g., "**ar**" vs "**aar**" for the Bengali word **আর**) produced a higher WER of 42.74%. Standardizing the lexicon improved this to 38.33%, highlighting that the quality of the synthetic text prompts is as important as the audio quality itself. Table 3 summarizes the final WER performance across different acoustic modeling stages. The Kaldi lexicon was derived by combining a standard CMU Sphinx phone set with a custom pronunciation dictionary for code-mixed tokens. To resolve transliteration noise, we implemented a rule-based phonetic normalization pipeline that mapped variant spellings (e.g., "**ar**" vs "**aar**") to a single, consistent phonetic representation.

Model Configuration	WER (%)
Monophone	51.14
Triphone (Tri1)	39.23
LDA-MLLT (Tri2b)	38.33
Wide-Beam Decoding (17.0)	**37.74**

Table 3: ASR performance (WER %) on the Bengali-English disfluent code-mixed subset using the Kaldi GMM-HMM baseline. The system achieves a best-case WER of 37.74% with wide-beam decoding.

The results indicate that the synthetic data possesses sufficient phonetic consistency to train a viable acoustic model from scratch. Notably, a simpler Triphone model achieved a WER of 39.23%, suggesting that for micro-resource synthetic data, lexicon consistency is often more critical than model complexity. The best performance (WER of 37.74%) was achieved by widening the decoding beam to account for the higher perplexity at code-switching points. This 37.74% WER is particularly significant given the disfluent code-mixed micro-resource regime. For context, recent studies on similar low-resource Indic languages using advanced Wav2Vec2 architectures reported WERs as high as 47.10% even with significantly more data (10 hours) (Dar and Pushparaj, 2026). Our results demonstrate that for extremely scarce, disfluent code-mixed data, a rigorously normalized GMM-HMM pipeline performs better than the theoretical baselines of deep neural models which are prone to severe overfitting in cold-start scenarios.

4.1.2. Disfluency Retention

A key objective was to evaluate if the model could capture synthetic disfluencies. Unlike standard systems that often filter filled pauses, our GMM-HMM baseline successfully retained these tokens. Table 4 highlights two examples where the model correctly transcribed filled pauses (*"um"*), repairs (*"i mean"*), and repetitions (*"oh oh"*). This confirms that the synthetic data provides sufficient spectral evidence to treat disfluencies as valid lexical units rather than background noise.

Case	Type	Transcript Sequence
1	Ref	*... **um i mean** ekta choto re-union*
	Hyp	*... **um i mean** ekta choto re-union*
	Status: Correctly identified repair & filled pause	
2	Ref	*acha **oh oh***
	Hyp	*acha **oh oh***
	Status: Correctly identified repetition	

Table 4: Successful recognition of disfluent markers in the BE-CM disfluent synthetic test set.

4.2. Foundation Model Adaptation: Whisper and PEFT

While classical models offer a stable baseline, modern sequence-to-sequence foundation models represent the state-of-the-art for high-resource languages. We sought to determine if our synthetic data could bridge the domain gap for these models in an under-resourced, code-mixed disfluent setting. We evaluated the `openai/whisper-small` baseline on the same 1.3 hours of BE-CM disfluent audio data. In a zero-shot setting, the model performed very poorly. This failure was driven by severe generative hallucinations where the model frequently failed to predict end-of-sequence tokens, falling into infinite loops and generating repetitive, non-existent phrases. To resolve this, we applied *Parameter-Efficient Fine-Tuning (PEFT)* using *Low-Rank Adaptation (LoRA)*. By fine-tuning the model for just 1,000 steps on our synthetic subset, we achieved a final WER of 21.37%.

The adaptation completely stabilized the decoding process. Using LoRA adapters effectively decreased the hallucinatory insertion errors, teaching the model to correctly identify utterance boundaries. Also, the fine-tuned model successfully bridged the acoustic domain gap, accurately capturing the rapid Bengali-English language switching and the specific prosodic cues of the synthetic disfluencies. As a result, the primary error mode shifted from massive structural hallucinations to minor lexical substitutions. These results demonstrate that our generation methodology, combined with targeted adaptation, can transform a fundamentally unstable zero-shot baseline into a highly robust ASR system.

5. Applications and Uses

The BEHE-CMDisfl corpus and our synthetic generation methodology offer scalable solutions for under-resourced conversational disfluent code-mixed speech technologies. Based on our experiments, we identify four primary applications for this work as described below:

- **Adapting Foundation Models:** Leveraging targeted synthetic data alongside PEFT (e.g., LoRA) to bridge domain gaps and suppress zero-shot hallucinations in massive models like Whisper.
- **Micro-Resource ASR Bootstrapping:** Training robust acoustic models under extreme data constraints ($<$2 hours) to accurately recognize and retain disfluencies as valid lexical tokens.
- **Disfluency Detection and Repair:** Utilizing explicitly aligned examples of hesitations and restarts to train supervised NLP modules that improve the readability of conversational transcripts.
- **Conversational TTS Modeling:** Providing a controlled, paired text-speech environment to study and enhance synthetic voice prosody at complex code-switching boundaries.

6. Conclusion and Future Work

6.1. Conclusion

We introduced BEHE-CMDisfl, a synthetic Bengali-English and Hindi-English code-mixed corpus with disfluencies. We validated this resource across two ASR paradigms in a severe micro-resource regime for the BE-CM data. First, a rigorously normalized Kaldi GMM-HMM baseline achieved a 37.74% WER, successfully retaining disfluency markers as valid lexical units. Second, adapting the Whisper foundation model via LoRA for just 1,000 steps suppressed severe zero-shot hallucinations, thus reducing the WER to 21.37%. These results establish our LLM-TTS pipeline and targeted adaptation as a scalable, effective methodology for solving the cold-start problem in disfluent code-mixed under-resourced languages.

6.2. Future Work

Our immediate road map for extending this methodology includes:

- **Real-Speech Validation:** Addressing the generalizability of our synthetic findings, by curating a 'gold-standard' test set of 3–5 hours of real-world conversational speech. This will provide a benchmark to quantify the 'Sim-to-Real' gap and determine how well the studied models perform in natural, noisy environments.
- **Advanced Adaptation & Translation:** Extending PEFT approaches to other architectures (e.g., Wav2Vec2) and exploring the use of this data for speech-to-speech translation to normalize disfluent audio.
- **Cross-Lingual Generalization:** Testing the language-agnostic scalability of our pipeline on the Hindi-English disfluent code-mixed subset and extending it to other morphologically rich languages like Tamil and Marathi.

6.3. Final Remarks

We believe **BEHE-CMDisfl** provides a practical, reproducible resource for the community while emphasizing transparency and ethical safeguards. Researchers using the corpus should validate findings on real conversational speech before deployment in safety-critical settings.

7. Ethical Considerations and Limitations

We recognize both the benefits and potential risks of generating synthetic, code-mixed, disfluent speech. Key ethical aspects and mitigations are summarized below:

Transparency: All generation steps—including LLM prompts, TTS model versions, and post-processing scripts—should be fully documented to ensure reproducibility and accountability.

Misuse Risks: Synthetic speech can be exploited for voice spoofing or misinformation. To mitigate such misuse, dataset releases must include clear usage guidelines, watermarking procedures, and, where possible, benchmarks for deepfake detection.

Bias and Representativeness: Both text generation and TTS systems inherit demographic and linguistic biases. We report speaker coverage, avoid mimicking real individuals, and caution against overgeneralization across underrepresented dialects.

Evaluation Limits: Synthetic disfluencies may not fully capture natural conversational patterns; results obtained from synthetic data should be validated on real-world speech to ensure generalization.

Privacy and Consent: As no real voices are used, privacy risks are minimal. However, the potential for misuse (e.g., impersonation) warrants controlled access and restrictive licensing.

8. Bibliographical References

References

Jordi Adell, Antonio Bonafonte, and David Escudero. 2006. Disfluent speech analysis and synthesis: A preliminary approach. In *Proc. 3rd Int. Conf. on Speech Prosody*, pages 1–4.

Sadia Alam, Md Farhan Ishmam, Navid Hasin Alvee, Md Shahnewaz Siddique, Md Azam Hossain, and Abu Raihan Mostofa Kamal. 2024. Bnsentmix: A diverse bengali-english code-mixed dataset for sentiment analysis. *arXiv preprint arXiv:2408.08964*.

Robin Amann, Zhaolin Li, Barbara Bruno, and Jan Niehues. 2024. Augmenting automatic speech recognition models with disfluency detection. pages 224–231.

Peter Auer. 2013. *Code-switching in conversation: Language, interaction and identity*. Routledge.

L. Besacier, E. Barnard, A. Karpov, and T. Schultz. 2014. Automatic speech recognition for under-resourced languages: A survey. *Speech Communication*, 56:85–100.

Vineet Bhat, Preethi Jyothi, and Pushpak Bhattacharyya. 2023. Adversarial training for low-resource disfluency correction. *arXiv preprint arXiv:2306.06384*.

Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. 2017. Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline. In *Proc. O-COCOSDA*, pages 1–5.

M. A. Dar and J. Pushparaj. 2026. A wav2vec2 model-based automatic speech recognition system for low-resource kashmiri language. *International Journal of Speech Technology*, 29(1):2.

A. Dhasmana, A. Srivastava, and D. Chiang. 2026. Dialect matters: Cross-lingual asr transfer for low-resource indic language varieties. *arXiv preprint arXiv:2601.04373*.

Anuj Diwan, Rakesh Vaideeswaran, Sanket Shah, Ankita Singh, Srinivasa Raghavan, Shreya Khare, Vinit Unni, Saurabh Vyas, Akash Rajpuria, Chiranjeevi Yarra, et al. 2021. Multilingual and code-switching asr challenges for

low resource indian languages. *arXiv preprint arXiv:2104.00235*.

Promila Ghosh. 2025. Medibeng (revision b05b594).

J Godfrey and E Holliman. 1997. Switchboard-1 release 2: Linguistic data consortium. *Switchboard: a user's manual*.

A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

T. D. Harya. 2018. Sociolinguistics: Code switching and code mixing. *Lentera: Jurnal Ilmiah Kependidikan*, 11(1):87–98.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Eunhee Kim. 2006. Reasons and motivations for code-mixing and code-switching. *Issues in EFL*, 4(1):43–61.

E. Kullmann. 2016. Speech-to-text for swedish using kaldi. Master's thesis, KTH Royal Institute of Technology.

Rohit Kundu, Preethi Jyothi, and Pushpak Bhattacharyya. 2022. Zero-shot disfluency detection for indian languages. In *Proceedings of the 29th international conference on computational linguistics*, pages 4442–4454.

Y. Lacombe, V. Srivastav, and S. Gandhi. 2024. Parler-tts. GitHub repository.

D. Lyth and S. King. 2024. Natural language guidance of high-fidelity text-to-speech with synthetic annotations. *arXiv preprint arXiv:2402.01912*.

M. Mohri, F. Pereira, and M. Riley. 2008. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer.

Melissa G Moyer. 2002. Bilingual speech: A typology of code-mixing.

Md Nishat Raihan, Dhiman Goswami, Antara Mahmud, Antonios Anastasopoulos, and Marcos Zampieri. 2023. Sentmix-3l: A bangla-english-hindi code-mixed dataset for sentiment analysis. *arXiv e-prints*, pages arXiv–2310.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, pages 5206–5210.

A. Pandey, B. M. L. Srivastava, and S. Sitaram. 2018. Adapting monolingual resources for code-mixed hindi-english speech recognition. In *Proc. IEEE ICASSP*.

Braja Gopal Patra, Dipankar Das, and Amitava Das. 2018. Sentiment analysis of code-mixed indian languages: An overview of sail_code-mixed shared task@ icon-2017. *arXiv preprint arXiv:1803.06745*.

Aryan Paul, Tapabrata Mondal, Dipankar Das, and Sivaji Bandyopadhyay. 2025. Generating and analyzing disfluency in a code-mixed setting. *Recent Advances in Natural Language Processing (RANLP)*, pages 915–924.

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, et al. 2011. The kaldi speech recognition toolkit. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.

Amrit Romana, Minxue Niu, Matthew Perez, and Emily Mower Provost. 2024. Fluencybank timestamped: An updated data set for disfluency detection and automatic intended speech recognition. *Journal of Speech, Language, and Hearing Research*, 67(11):4203–4215.

Jivnesh Sandhan, Ayush Daksh, Om Adideva Paranjay, Laxmidhar Behera, and Pawan Goyal. 2022. Prabhupadavani: A code-mixed speech translation data for 25 languages. *arXiv preprint arXiv:2201.11391*.

Ashwin Sankar, Yoach Lacombe, Sherry Thomas, Praveen Srinivasa Varadhan, Sanchit Gandhi, and Mitesh M Khapra. 2025. Rasmalai: Resources for adaptive speech modeling in indian languages with accents and intonations. *arXiv preprint arXiv:2505.18609*.

S Saranya, B Bharathi, S Gomathy Dhanya, and Aishwarya Krishnakumar. 2025. Real-time continuous tamil dialect speech recognition and summarization. *Circuits, Systems, and Signal Processing*, 44(4):2855–2881.

Elizabeth Shriberg. 2001. To 'errrr'is human: ecology and acoustics of speech disfluencies. *Journal of the international phonetic association*, 31(1):153–169.

Vinay Singh, Deepanshu Vijay, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. Named entity recognition for hindi-english code-mixed social media text. In *Proceedings of the seventh named entities workshop*, pages 27–35.

S. Sitaram, K. R. Chandu, S. K. Rallabandi, and A. W. Black. 2019. A survey of code-switched speech and language processing. *arXiv preprint arXiv:1904.00784*.

Bao Thai, Robert Jimerson, Dominic Arcoraci, Emily Prud'hommeaux, and Raymond Ptucha. 2019. Synthetic data augmentation for improving low-resource asr. In *2019 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, pages 1–9.

S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, et al. 2018. Espnet: End-to-end speech processing toolkit. In *Proc. Interspeech*.

Peng Xie, Xingyuan Liu, Tsz Wai Chan, Yequan Bie, Yangqiu Song, Yang Wang, Hao Chen, and Kani Chen. 2025. Switchlingua: The first large-scale multilingual and multi-ethnic code-switching dataset. *arXiv preprint arXiv:2506.00087*.

Y. H. Yeo, Y. Hu, S. Gopal, Y. Peng, H. Liu, and E. S. Chng. 2026. Improving code-switching speech recognition with tts data augmentation. *arXiv preprint arXiv:2601.00935*.

Emre Yılmaz, M. Andringa, S. Kingma, J. Dijkstra, F. van der Kuip, H. van de Velde, et al. 2016. Longitudinal speaker clustering and identification for frisian-dutch code-switching. In *Proc. Interspeech*, pages 3668–3672.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional lstm. *arXiv preprint arXiv:1604.03209*.