CLEAR: Code-Mixed ASR with LLM-Driven Rescoring

Shivam Kumar, Md. Shad Akhtar
Department of CSE, IIIT-Delhi, New Delhi, India
{shivam23004,shad.akhtar}@iiitd.ac.in

Abstract

Code-mixing presents significant challenges for Automatic Speech Recognition (ASR), especially for Indian languages, due to homophone ambiguity, domain-specific word identification, and data scarcity. Traditional ASR models struggle with these complexities, often failing to differentiate between phonetically similar words in multilingual contexts. To address this, we propose CLEAR, a novel rescoring model that integrates descriptive prompting and LLM-based rescoring while analyzing the impact of n-best hypotheses across multiple beam widths. CLEAR enhances ASR performance, achieving S-WER of 26.9, P-WER of 26.46, and T-WER of 25.04—improving by 6.9%, 13.47%, and 4.42%, respectively, over the best baseline TDNN. These findings demonstrate that CLEAR effectively resolves homophone ambiguities and refines transcriptions, leading to a 13.56% S-WER and 7.77% T-WER reduction over decoder only fine-tuned Whisper.

1 Introduction

Code-mixing and code-switching are prevalent linguistic phenomenon in multilingual communities, where speakers alternate between languages within a single discourse. The terms code-switching and code-mixing are often used interchangeably[1]; however, they carry distinct linguistic meanings. Code-switching is intersentential where the language/code switch occurs at the utterance level; while code-mixing is intrasentential where the switch happens at the word or phrase level within a sentence (Thara and Poornachandran, 2018; Setiawan, 2023; Winata et al., 2022).

Developing automatic speech recognition (ASR) systems for code-switched speech presents unique challenges due to the nature of the language(Çetinoğlu et al., 2016). Code-switching introduces linguistic complexities such as cross-lingual homophone disambiguation (e.g., Bill in English means a receipt or a piece of paper; however, बिल(Bill) in Hindi means a hole or a burrow) (Yu et al., 2024), code-switching point detection (Wang et al., 2019), and the identification of embedding and matrix languages, which are crucial for determining the correct syntactic structure (e.g., Subject-Verb-Object (SVO) in English vs. Subject-Object-Verb (SOV) in Hindi) (Iakovenko and Hain, 2024). Accurately modeling these aspects is essential for generating grammatically coherent transcriptions.

To mitigate these challenges, recent advancements (Prabhavalkar et al., 2023) in ASR have been driven by innovations in neural architectures, training strategies, and robust learning techniques. Many explorations in end-to-end ASR leveraged transformer-based encoders and self-supervised pretraining to learn rich representations from raw acoustic signals (Baevski et al., 2020; Hsu et al., 2021; Chadha et al., 2022). Whisper (Radford

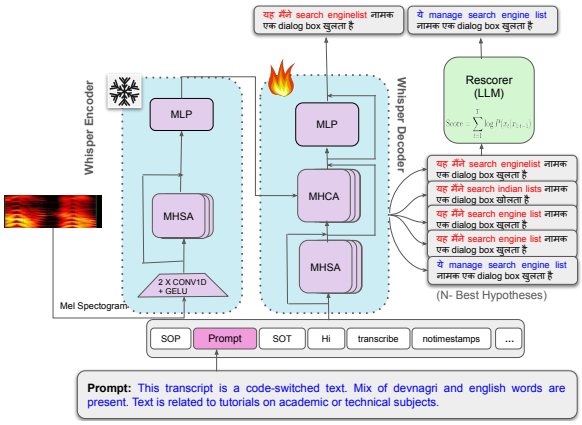

Figure 1: Proposed architecture of CLEAR.

[1]We will also use these two terms interchangeably to refer our input setting in this paper.

et al., 2023) is one of most popular state-of-the-art multilingual ASR models, trained on 630K hours of data, that can transcribe, translate, and detect speeches across 99 languages. Peng et al., 2023 incorporated a mixture of language tags, i.e., <zh><en> for Mandarin-English code-switched dataset; while Yang et al., 2024c extended Whisper with a separate language tag <|en−zh|> to uniquely refer to Mandarin-English code-mixed sentences.

With the inception of LLMs, recent studies have leveraged prompting strategies (Liu et al., 2023b) in Whisper-based architectures. Suh et al., 2024 explored LLMs to generate contextual descriptions, which were then used to prompt the Whisper for transcription, demonstrating the potential of LLM-guided ASR.

Although previous methods produced good results, they required generating different mixes of language tags or using an LLM to create prompts for each utterance, we do not aim to do that also these works have been experimented on Mandarin-English code-switchted dataset belongs to different domain and not on Hindi-English code-switched which has its own linguistic complexities. Our research builds upon these works but takes a distinct approach. Unlike prior studies, we do not introduce new language tags or rely on LLM-generated prompts. Instead, we propose CLEAR to demonstrate that descriptive prompting alone can yield high-quality code-mixed transcriptions by refining ASR outputs through LLM-based rescoring. We hypothesize that LLM-based rescoring can mitigate the issue of homophone disambiguation by scoring the fluent sentences with higher scores. We obtain n-beams of potential outputs from Whisper and utilize LLM-based scorer to measure their linguistic fluency and coherency. We fine-tune the Whisper decoder while keeping the encoder in a frozen state. Specifically, we experiment with leading LLMs – GPT-2, LLaMA 3.1 (8B), LLaMA 3.2 (1B), DeepSeek R1[2], Qwen-2 (7B), Mistral (7B), and GPT-4 (Radford et al., 2019; Dubey et al., 2024; Guo et al., 2025; Yang et al., 2024a; Jiang et al., 2023; Achiam et al., 2023) – while varying beam widths to assess their impact on ASR performance. We employ MUCS 2021 dataset to evaluate CLEAR. Our analysis across six competitive baselines signifies the importance of CLEAR on code-switched ASR outputs. Our contributions are summarized below:

- We present CLEAR and contribute novel insights into the role of descriptive prompting and LLM-based scoring in improving code-switched ASR systems, paving the way for more effective transcription models in multilingual settings.
- We extensively evaluate CLEAR against 6 baseline methods. We present CLEAR significant improvement evaluated across evaluation metrics.

Reproducibility: https://github.com/flamenlp/CLEAR

2 Related Work

Another line of work have explored approaches to improve language modeling and context understanding. For instance, Aditya et al., 2024 investigated attention mechanisms within transformer layers, identifying attention heads that effectively capture language identities and guiding them accordingly. To mitigate multilingual context confusion, Zhang et al., 2022 proposed attention weight recomputation to better differentiate languages within speech. Further, Liu et al., 2023a, 2024a introduced language biases at both the token and frame levels to enhance the model's ability to handle language switching effectively. Song et al., 2022 proposed a language-specific characteristic assistance (LSCA) method to mitigate the problem caused by lanugage-specific encoders (LSEs) since most existing methods did not have language constraints; they introduced a language-specific loss to do that. To disambiguate homophones Srivastava and Sitaram, 2018 used a WX-based common pronunciation scheme for mixed language pairs and unification of homophones during training, resulting in a lower word error rate for systems built using this data. Chung et al., 2022 proposed a novel homophone extension method to integrate human knowledge of the homophone lexicon into the beam search decoding process with language model re-scoring. Some of the recent work has also explored Mixture of Experts

[2]DeepSeek-R1-Distill-Llama-8B

(MoE) architectures for code-switched ASR. Ye et al., 2024 proposed using separate encoders as language experts, while Yang et al., 2024b introduced a disentanglement loss to enable lower encoder layers to capture inter-lingual acoustic information while reducing linguistic confusion in higher layers. Liu et al., 2024b introduced a language alignment loss in ASR training to align acoustic features to pseudo-language labels learned from the decoder and also employs LLM via generative error correction to tackle the problem caused by complex token alternatives for language modeling in bilingual scenarios.

3 Methodology

In this section, we describe **CLEAR** model to enhance transcription of Hindi-English code-switched speech. Our primary objective is to use a descriptive prompting strategy that provides contextual guidance to the decoder, improving transcription accuracy without requiring extensive fine-tuning of the entire model. Additionally, we incorporate LLMs for rescoring to further refine the final transcription output.

Proposed Pipeline: Whisper, unlike conventional ASR models, allows for prompting (Suh et al., 2024) through special tokens that guide the transcription process. These tokens include `<|sop|>` (*start-of-previous*), `<|sot|>` (*start-of-transcript*), `<|en|>` or `<|hi|>` (language tags), `<|transcribe|>` (specifies the task as transcription), and `<|notimestamps|>` (to disable word-level timestamps). In our proposed pipeline, we strategically place our custom prompt after the `<|sop|>` token. This placement provides contextual information to the decoder while ensuring compliance with Whisper's input constraints, as only 224 tokens[3] can be used as a prompt. We also experiment with different prompts to check which is working best, more details will be discussed in section 4. Our constructed prompt follows the format:

`<|sop|><|prompt|><|hi|><|transcribe|><|notimestamps|>`

We fine-tune Whisper by integrating our designed prompt to influence the decoder's behavior.

LLM-Based Scorer: To further refine transcription quality, we introduce a rescoring mechanism utilizing LLMs. LLMs are trained on vast multilingual corpora (Gurgurov et al., 2024), effectively capture semantic structures and can assist in selecting the most plausible transcription candidate. Given a beam width of n, the Whisper decoder generates n transcription hypotheses $\{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$. The rescoring process involves computing the sum of log probability of each hypothesis $x^{(i)}$ based on the LLMs ouput logits, we call this sum as score. The log probability of a candidate sequence is given by:

$$\begin{aligned}
\log P(\mathbf{x^{(i)}}) &= \sum_{t=1}^{T} \log P(x_t | x_{1:t-1}, \theta) \\
&= \sum_{t=1}^{T} \log \left(\frac{\exp(z_t^{(x_t)})}{\sum_j \exp(z_t^{(j)})} \right) \\
&= \sum_{t=1}^{T} \left(\log \left(\exp(z_t^{(x_t)}) \right) - \log \sum_j \exp(z_t^{(j)}) \right) \\
&= \sum_{t=1}^{T} \left(z_t^{(x_t)} - \log \sum_j \exp(z_t^{(j)}) \right) \qquad (1)
\end{aligned}$$

where t is current time step, and T is the total number of steps. The sequence $x = (x_1, x_2, \ldots, x_T)$ consists of tokens x_t, each assigned a logit $z_t^{(x_t)}$ by the LLM. The LLMs parameters are denoted by θ. j is the index in the output vocabulary.
The best transcription x^* is then selected as the one with the highest log probability among all n hypotheses:

$$\mathbf{x}^* = \arg \max_{i \in 1,2,\ldots,n} \log P(\mathbf{x}^{(i)}) \qquad (2)$$

We conduct experiments with different LLMs and beam widths to assess their impact on transcription quality.

Fine-Tuning: Our fine-tuning process focuses solely on the decoder while keeping the encoder frozen. Since Whisper has been pre-trained on 630K hours of multilingual speech data, its encoder already possesses a strong understanding of the acoustic properties of speech. Freezing the encoder prevents overfitting and ensures that the model retains its

[3] https://cookbook.openai.com/examples/whisper_prompting_guide

Prompt-1	Prompt-2	Prompt-3
This transcript is a code-switched text. Mix of devnagri and english words are present. Text is related to tutorials on academic or technical subjects.	This transcript is a code-switched text. Mix of Devanagari and English words are present. Text is related to tutorials on academic or technical subjects. Few examples look like this: तो electricity bill option पर click करें कुछ गलत password दीजिए और enter प्रेस करें	The transcript comprises telephone quality speech data in Hindi. Transcript is mixed of Hindi and English words like this: hi hi hi hi en hi hi hi hi hi en hi hi hi hi. Transcribe the speech in this format.

Table 1: Prompts use to fine-tune the CLEAR is listed here. We experiment with many prompts, few of them are shown in this table.

general ASR capabilities while adapting its decoder for improved handling of code-switched speech. Previous studies (Yang et al., 2023) have shown that Whisper with a frozen encoder can achieve superior performance on certain ASR tasks. We will also prove that decoder-only finetuning works in the section 5. Our methodology, which integrates Whisper's ASR capabilities with descriptive prompting and LLM-based rescoring, presents an efficient approach for improving code-switched speech recognition without altering language tags (Yang et al., 2024c).

4 Experiments

Dataset Description: We use the Hindi-English code-switched dataset[4] from the MUCS Challenge, Interspeech 2021 (Diwan et al., 2021), derived from spoken tutorials on technical topics. All audio files are sampled at 16 kHz with 16-bit encoding. The dataset comprises spontaneous speech from educational settings, making it particularly challenging due to variations in speaker accents, speech disfluencies, and technical terms. It consists of ~100 hours of data and splitted into train (89.86 hrs), test (5.18 hrs), and blind (6.24 hrs) sets. Moreover, the dataset has code-switching percentage[5] of 85.88%, 81.88%, and 95.55% in train, test, and blind sets, respectively. The dataset also contains enunciated punctuation (e.g., "<" for "lesser"). Furthermore, the train and test sets have ~33.9% overlaps; however, blind test-train sentence overlap is 2.1%. Therefore, we evaluate CLEAR on blind set only.

Evaluation Metric: For evaluation, we compute three variants of word-error-rate (WER), i.e., strict-WER (S-WER), punctuation-WER (P-WER), and transliterated-WER (T-WER). S-WER is the standard WER metric which computes the transcription error rate at the surface level. In comparison, P-WER accounts for variations in punctuation by applying a predefined punctuation mapping before computing WER (replace "greater" by ">"); thus, ensuring consistency in evaluation. On the other hand, T-WER assesses errors in cross-linguistic transcription by replacing words in the predicted transcriptions with their corresponding transliterated forms (replace डिस्क्रिप्शन by description). These three WER variants provide a comprehensive assessment of the ASR model, capturing both standard transcription errors and linguistic variations of code-mixed languages.

Descriptive Prompt Details: We experiment with different numbers of prompts in different styles to guide the decoder; a few of them worked and some did not. Some examples of prompts are listed in Table 1. The prompts that gives the best performance are Prompt-1 and Prompt-2 and also in in CLEAR architecure we have used this Prompt-1, you can see in the Fig. 1. Prompt-1 is very simple

[4] https://www.openslr.org/104/

[5] code-switched utterances upon total no. of utterances

Model	S-WER	P-WER	T-WER
TDNN	28.90	–	26.20
E2E Transformer	33.65	–	29.80
PromptingWhisper	42.10	41.72	34.75
Whisper (ZS)	266.5	265.84	263.30
Whisper (FT)	32.16	31.64	28.54
Whisper (FE)	31.12	30.58	27.15
CLEAR	**26.9** (↓ 6.9%)	**26.46** (↓ 13.47%)	**25.04** (↓ 4.42%)

Table 2: Comparative results on MUCS dataset. ZS = Zero Shot, FT = Full finetuning, and FE = Finetune by Frozen Encoder

to tell the decoder that text is code-switched and guiding what code-switching means by mentioning about type of languages involed and also telling the decoder that dataset is related to technical subject and tutorial taught in academic setting, basically it is giving the contextual cues about the dataset. Similary Prompt-2 is doing the same, additionally it is also giving the examples to make it more clear. And Prompt-3 even tells the decoder when the language switch is happening, which will allow the decoder to predict the language switch even better, one of the major problems of code-switching. But sometimes these Prompt-2, 3 and some more which are not listed here do not work. One of the problems we find is that these are longer prompts, which may be longer for some utterances, and Whisper can understand only 224 tokens as a prompt and anything longer that will be truncated. Therefore, Prompt-3 mostly will fail due to its dynamic nature for each utterance, Prompt-2 can work if it is within the token limits.

Baselines: We compare the performance of our CLEAR model with following baselines. DNN-HMM (Diwan et al., 2021) is a neural network created using the Kaldi toolkit[6], consisting of 8 TDNN (Time-Delay Neural Network) blocks (Peddinti et al., 2015) with a dimension of 768. End-to-End (E2E) Transformer (Diwan et al., 2021) is a hybrid CTC-Attention model (Watanabe et al., 2017) with a 12-layer encoder and a 6-layer decoder, each with 2048 units and 0.1 dropout rate. It employs a CTC weight of 0.3 and an attention weight of 0.7, using eight 64-dimensional attention heads per layer. PromptingWhisper (Peng et al., 2023) is a Whisper-large model tested in a zero-shot setup by changing the language tag. Here, we use <|hi|><|en|> as the language tag. Additionally, we employ Whisper in both zero-shot and fine-tuning settings. For CLEAR we utilize Whisper-small[7] (12 self-attention layers in both the encoder and the decoder) which is capable of processing 30-second audio segments and generates text autoregressively.

[6] https://kaldi-asr.org/

Scorer	Beam	S-WER	P-WER	T-WER
–	1	28.09 (↓ 9.73%)	28.42 (↓ 7.06%)	26.06 (↓ 4.01%)
GPT2	5*	**26.9**	**26.46**	**25.04**
	10	27.11	26.67	25.47
LLaMA 3.1 (8B)	5	28.26	27.78	25.02
	10	28.39	27.91	25.09
LLaMA 3.2 (1B)	5	27.75	27.30	25.69
	10	27.86	27.42	26.03
DeepSeek	5	27.60	27.14	25.55
	10	28.20	27.74	26.35
Qwen-2 (7B)	5	27.48	27.01	25.40
	10	27.89	27.44	26.06
Mistral (7B)	5	27.55	27.11	25.49
	10	27.91	27.50	26.09
GPT-4	5	27.85	27.29	24.82

Table 3: Ablation on different beam widths. Star(*) signifies the CLEAR model.

Training Details: We fine-tune CLEAR for 10 epochs using a learning rate of 1e−4. The training was conducted on an NVIDIA A100 GPU with a batch size of 16. We employ AdamW as the optimizer with a weight decay of 0.01 to regulate parameter updates. The training procedure leverages mixed-precision training to improve efficiency and reduce memory consumption while maintaining numerical stability. During inference, we adopt beam search decoding (beam-width = n) to improve transcription accuracy. We also experiment with temperature scaling to prevent overly confident incorrect predictions.

5 Results and Analysis

Table 2 presents the comparative analysis of CLEAR against other baselines on the blind set of the MUCS dataset. Our initial evaluation

[7] https://huggingface.co/openai/whisper-small

of Whisper in a zero-shot (ZS) setting revealed a drastic degradation in performance, with S-WER exceeding 260%. This excessively high WER is primarily caused by repetitive character sequences and a lack of domain adaptation, leading to significant transcription errors. The model struggles with both code-switching regions and the specific linguistic patterns of the dataset, underscoring the limitations of the out-of-the-box Whisper-small model in handling code-mixed speech. To mitigate these issues, we fine-tune Whisper in two settings, first full fine-tuning (FT) and second fine-tuning by frozing encoder (FE), results are shown in Table 2. Whisper (FT) shows improvement in S-WER (32.16), P-WER (31.64), and T-WER (28.54) as compared to ZS setting. We further fine-tune Whisper by frozing encoder (FE) on our dataset, allowing it to adapt to the linguistic characteristics of code-switched speech. This results in a substantial improvement in S-WER (31.12) compared to ZS setting and this is also an improvement over Whisper(FT), as the model exhibit a better understanding of language semantics, reduced character repetition, and improved handling of code-switching boundaries. The results of Whisper(FT) and Whipser(FE) shows that fine-tuning by the frozing encoder works very well as compared to full fine-tuning. Therefore, for our proposed pipeline we will use Whipser (FE) for all our experiments. Despite these gains, domain-specific challenges persisted, particularly for low-frequency words, homophone inconsistency, and unseen terms that were not well represented in the corpus.

To further enhance transcription accuracy, we use descriptive prompt-based fine-tuning approach (Peng et al., 2023), where the Whisper decoder was fine-tuned while keeping the encoder frozen. This strategy led to a 9.73% reduction in S-WER and a 4.01% reduction in T-WER (beam $n = 1$ in Table 3) compared to Whisper (FE). These results indicate that prompt-based fine-tuning is an effective adaptation strategy for code-switching ASR, significantly improving performance without requiring architectural modifications to the Whisper model. Building on this, we employ our scoring mechanism to grade the n-best hypothesis from the fine-tuned Whisper model. `CLEAR` reports a reduced score of S-WER (26.9), P-WER (26.46), and T-WER (25.04) – a significant reduction of +6.9%, +13.47% and +4.42% in S-WER, P-WER, and T-WER, respectively, over the best baseline (i.e TDNN).

As reported in Table 3, we employ multiple open-source LLMs, such as GPT-2, LLaMA 3.1 (8B), LLaMA 3.2 (1B), DeepSeek, Qwen-2 (7B), Mistral (7B), and GPT-4 for transcription scoring. We systematically analyzed the impact of beam width on WER, experimenting with $n = 5, 10, 15$ and 20[8]. Our findings indicate that a beam width of 5 consistently yielded the best results across models. In particular, compared to fine-tune (FE) Whisper-small, GPT-2 achieved a 13.56% reduction in strict WER and a 7.77% reduction in transliterated WER; LLaMA 3.1 (8B) yielded a 9.19% reduction in strict WER and a 7.84% reduction in transliterated WER; LLaMA 3.2 (1B) yielded a 10.82% reduction in strict WER and a 5.37% reduction in transliterated WER; Deepseek yielded a 11.31% reduction in strict WER and a 5.89% reduction in transliterated WER; Qwen yielded a 11.69% reduction in strict WER and a 6.44% reduction in transliterated WER; Mistral yielded a 11.47% reduction in strict WER and a 6.11% reduction in transliterated WER; GPT-4 yielded a 10.50% reduction in strict WER and a 20.24% reduction in transliterated WER, It outperforms GPT2 by 0.87% only in T-WER. However, we would like to highlight that GPT-4 incurred a significant 3$/100 hypotheses during inference. This justifies the use of GPT-2 based scorer in `CLEAR` as a budget effective solution. Also rescoring mechanism with GPT-4 was different compared to other LLMs because we can't extract the logits from GPT-4 therefore we rank the hypothesis based on their accuracy, coherency, and fluency on a scale of -10 (very inaccurate) to 10 (perfectly accurate). Calculating sum of log probabilites of logits will not work in the case of GPT-4.

These results highlight the effectiveness of LLM-based rescoring techniques, demonstrating their ability to refine transcription outputs and further reduce errors in code-mixed ASR

[8]We do not report n=15 and n=20 due to inferior results. We observe performance drops across multiple scorers with beam>5.

	Model	**Text**	**Remarks**
Sentence 1	GT	अब इस method पर आते हैं	–
	W-(ZS)	अब इस मेफ पर आते	"method" is missing, and substitution leads to loss of meaning
	W-(FE)	और formula का स्टेटमेंट दें	Extraneous words introduced, altering the intended meaning.
	CLEAR–R	अब इस method पर आते हैं	Matches GT exactly, correct transcription.
	CLEAR	अब इस method पर आते हैं	Matches GT exactly, correct transcription.
Sentence 2	GT	get noise profile पर click करें	–
	W-(ZS)	अगर तो तो तो तो तो तो तो तो तो	Repeated words result in meaningless output.
	W-(FE)	cat noise profile पर click करें	Incorrect word "cat" instead of "get", missing domain-specific knowledge.
	CLEAR–R	get noise profile पर click करें	Correctly retains "get" and follows GT structure.
	CLEAR	get noise profile पर click करें	Matches GT exactly, preserving domain-specific knowledge.
Sentence 3	GT	अब वापस IDE पर आते हैं	–
	W-(ZS)	अब वापस आईडी पर आते	Misrecognition of "IDE" as **"आईडी"** changes the meaning.
	W-(FE)	अब वापस ID पर आते हैं	Homophones is not correctly identified
	CLEAR–R	अब वापस IDE पर आते हैं	Correctly identifies "IDE" and follows GT.
	CLEAR	अब वापस IDE पर आते हैं	Matches GT exactly, resolving homophone ambiguity
Sentence 4	GT	चिंता न करें यदि class diagram view में नहीं खुलता है	–
	W-(ZS)	जितना न करें ये दिखाएगा जगे व्यू में नहीं खुलता है	Unintelligible phrase with extra words.
	W-(FE)	चिंता न करें यदि, टगग्राम डायग्राम में नहीं खुलता है	Incorrectly replaces "class diagram view" with an unrelated term.
	CLEAR–R	चिंता न करें यदि class tag view में नहीं खुलता है	Partially correct, but "class tag view" is incorrect.
	CLEAR	चिंता न करें यदि class diagram view में नहीं खुलता है	CLEAR Matches GT exactly, ensuring correct code-switching.
Sentence 5	GT	1123 put insulin . fasta file के लिए contents	–
	W-(ZS)	अप्रोट इंसुलिन ड़ प्रश्टा फॉईल के लिए, ख़न्टेंस दिखाता है.	Misrecognized words distort the sentence meaning.
	W-(FE)	आउटपुट insulin dot first फाइल के लिए कंटेंट्स दिखाता है	Incorrect segmentation of "fasta file" as "dot first file."
	CLEAR –R	default insulin dot firster file के लिए कंटेंट्स दिखाता है	Better than W-(FE), but still modifies "fasta file."
	CLEAR	default installation dot firster file के लिए कंटेंट्स दिखाता है	CLEAR Corrects some errors but still alters "fasta file."

Table 4: Comparison of ASR outputs among competitive models. CLEAR–R: CLEAR without scorer (i.e., beam=1 and Whisper fine-tuned with descriptive prompt).

tasks. Overall, our findings establish that a combination of prompt-based fine-tuning and LLM-based rescoring substantially enhances the performance of the code-mixed ASR task.

Complexity of LLMs: While large language models (LLMs) are often associated with high computational costs, we carefully designed CLEAR to avoid these burdens. Rather than fine-tuning the LLMs, which would require significant GPU hours and memory, we utilized them solely for inference to score ASR hypotheses. This approach is lightweight - each hypothesis takes only about 0.6 to 0.7 seconds to evaluate — making it both efficient and practical for real-world post-processing scenarios. Interestingly, we also observed significant improvements in transcription quality even when using relatively smaller LLMs such as LLaMA-3.2 (1B). This suggests that even modest-sized language models can capture contextual nuances well enough to resolve ambiguities and correct ASR output, particularly in challenging settings such as code-mixed speech. This balance between performance gain and computational overhead suggests that we can use LLMs as scorers for post-processing the ASR outputs and makes our method feasible for broader deployement.

Qualitative and Error Analysis:
To further assess the quality of the generated transcriptions, we conduct a detailed qualitative and error analysis, as shown in Table 4. This analysis highlights the advantages of our proposed pipeline over the zero-shot and fine-tuned models in handling various linguistic complexities in code-switched ASR. Our pipeline exhibits significant improvements in multiple aspects, including word ordering, domain-specific terminology, homophone disambiguation, and rescoring-based refinement.

- **Word Ordering:** One notable enhancement is the model's ability to predict words in the correct order. As observed in Sentence 1, while the zero-shot model produces an incomplete and incorrect phrase, and the fine-tuned model introduces extraneous words, both the CLEAR–R and CLEAR versions successfully reconstruct the correct sentence structure. This sug-

gests that our approach effectively learns the sequential dependencies within code-mixed speech, leading to more coherent and grammatically accurate outputs.

- **Domain-specific terminology:** Another key improvement is in handling domain-specific terminology, as illustrated in Sentence 2. The fine-tuned model incorrectly transcribes "cat" instead of "get", demonstrating its struggle with differentiating both similar-sounding words and domain-specific words. CLEAR–R approach, however, accurately transcribes "get", showcasing a better understanding of contextual cues. This improvement is crucial in technical and specialized domains, where precise word recognition significantly impacts the usability of transcriptions.

- **Homophone Disambiguation:** In Sentence 3, CLEAR effectively tackles homophone disambiguation, particularly distinguishing between "id" and "ide". The fine-tuned model fails to capture this distinction, incorrectly predicting "id" instead of "ide", whereas the CLEAR–R and CLEAR accurately recognize the correct term. This capability is essential in technical environments where similar-sounding words carry distinct meanings, ensuring that transcripts remain contextually relevant.

- **Rescoring Refinements:** In sentence 4, demonstrates the effectiveness of rescoring in refining transcriptions. While the CLEAR–R output closely aligns with the ground truth, it incorrectly transcribes "class tag view" instead of "class diagram view". The CLEAR output successfully corrects this error, emphasizing the impact of our rescoring mechanism in enhancing transcription accuracy. This step ensures that even when the initial transcription is suboptimal, the model can refine its predictions to achieve greater alignment.

- **Pitfall of CLEAR:** It is not like our CLEAR model is giving good performance of every utterances. In some scenarios it is failing. As we can see in the sentence 5, it is not correctly predicting many words. One reason for this may there are two many code-switches available, puctuation "." is present, and domain specific word is also present. There may be possibility because the presence of these challenges causing the problem to the ASR.

6 Conclusion

This study introduces CLEAR, a novel approach to enhancing Hindi-English code-mixed ASR by integrating descriptive prompt-based fine-tuning and LLM-based rescoring. Our findings reveal that fine-tuning the Whisper decoder while freezing the encoder is a highly effective strategy for code-switching transcription (refer 2^{nd} last row of Table 2), yielding substantial reductions in various word error rates.Extensive ablation and qualitative analysis establishes LLM-based rescoring as an efficient refinement mechanism, effectively disambiguating homophones, and also enhances overall readability and domain-specific accuracy without requiring explicit language tags or specialized pre-training. These insights pave the way for more adaptable and resource-efficient LLM-guided ASR systems, particularly in low-resource and multilingual settings.

Limitations

This work has also certain limitations. Due to GPU memory constraints, we did not explore the impact of larger batch sizes, which could potentially influence model performance. Additionally, there is a lack of a diverse and high-quality code-mixed dataset for Indian languages. Our experiments are limited to MUCS code-switching data. To the best of our knowledge, this is the only publicly available dataset. We did not evaluate the approach on other pairs of Indian languages. The limited dataset diversity hinders the robustness, highlighting the critical need for high-quality datasets in this research area. However, we believe that the overall pipeline of our CLEAR model is highly adaptable to other language pairs as well. Another challenge is tackling too many code-switching points, availability of punctuation, and domain specific words all these togther if present in a sentence. sometimes CLEAR fails to handle this. In future we also plan to handle this limitaion of our model.

Acknowledgment

The authors acknowledge the support of the Infosys Foundation through CAI at IIIT-Delhi.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bobbi Aditya, Mahdin Rohmatillah, Liang-Hsuan Tai, and Jen-Tzung Chien. 2024. Attention-guided adaptation for code-switching speech recognition. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10256–10260. IEEE.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations.

Özlem Çetinoğlu, Sarah Schulz, and Ngoc Thang Vu. 2016. Challenges of computational processing of code-switching. *arXiv preprint arXiv:1610.02213*.

Harveen Singh Chadha, Priyanshi Shah, Ankur Dhuriya, Neeraj Chhimwal, Anirudh Gupta, and Vivek Raghavan. 2022. Code switched and code mixed speech recognition for indic languages. *arXiv preprint arXiv:2203.16578*.

HoLam Chung, Junan Li, Pengfei Liu, Wai-Kim Leung, Xixin Wu, and Helen Meng. 2022. Improving rare words recognition through homophone extension and unified writing for low-resource cantonese speech recognition. In *2022 13th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 26–30. IEEE.

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*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Daniil Gurgurov, Tanja Bäumel, and Tatiana Anikina. 2024. Multilingual large language models and curse of multilinguality. *arXiv preprint arXiv:2406.10602*.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units.

Olga Iakovenko and Thomas Hain. 2024. Methods for automatic matrix language determination of code-switched speech. *arXiv preprint arXiv:2410.02521*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Hexin Liu, Leibny Paola Garcia, Xiangyu Zhang, Andy WH Khong, and Sanjeev Khudanpur. 2024a. Enhancing code-switching speech recognition with interactive language biases. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 10886–10890. IEEE.

Hexin Liu, Haihua Xu, Leibny Paola Garcia, Andy WH Khong, Yi He, and Sanjeev Khudanpur. 2023a. Reducing language confusion for code-switching speech recognition with token-level language diarization. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Hexin Liu, Xiangyu Zhang, Haoyang Zhang, Leibny Paola Garcia, Andy WH Khong, Eng Siong Chng, and Shinji Watanabe. 2024b. Aligning speech to languages to enhance code-switching speech recognition. *arXiv preprint arXiv:2403.05887*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Interspeech*, volume 2015, pages 3214–3218.

Puyuan Peng, Brian Yan, Shinji Watanabe, and David Harwath. 2023. Prompting the hidden talent of web-scale speech models for zero-shot task generalization. *arXiv preprint arXiv:2305.11095*.

Rohit Prabhavalkar, Takaaki Hori, Tara N Sainath, Ralf Schlüter, and Shinji Watanabe. 2023. End-to-end speech recognition: A survey. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:325–351.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Budi Setiawan. 2023. Code-mixing vs code-switching: a study of grammatical perspective through code-switching varieties. *KnE Social Sciences*, pages 47–57.

Tongtong Song, Qiang Xu, Meng Ge, Longbiao Wang, Hao Shi, Yongjie Lv, Yuqin Lin, and Jianwu Dang. 2022. Language-specific characteristic assistance for code-switching speech recognition. *arXiv preprint arXiv:2206.14580*.

Brij Mohan Lal Srivastava and Sunayana Sitaram. 2018. Homophone identification and merging for code-switched speech recognition. In *Interspeech*, pages 1943–1947.

Jiwon Suh, Injae Na, and Woohwan Jung. 2024. Improving domain-specific asr with llm-generated contextual descriptions. In *Interspeech 2024*, pages 1255–1259.

S Thara and Prabaharan Poornachandran. 2018. Code-mixing: A brief survey. In *2018 International conference on advances in computing, communications and informatics (ICACCI)*, pages 2382–2388. IEEE.

Qinyi Wang, Emre Yılmaz, Adem Derinel, and Haizhou Li. 2019. Code-switching detection using asr-generated language posteriors. *arXiv preprint arXiv:1906.08003*.

Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. 2017. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

Genta Indra Winata, Alham Fikri Aji, Zheng-Xin Yong, and Thamar Solorio. 2022. The decades progress on code-switching research in nlp: A systematic survey on trends and challenges. *arXiv preprint arXiv:2212.09660*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Hao Yang, Jinming Zhao, Gholamreza Haffari, and Ehsan Shareghi. 2023. Investigating pre-trained audio encoders in the low-resource condition. *arXiv preprint arXiv:2305.17733*.

Tzu-Ting Yang, Hsin-Wei Wang, Yi-Cheng Wang, Chi-Han Lin, and Berlin Chen. 2024b. An effective mixture-of-experts approach for code-switching speech recognition leveraging encoder disentanglement. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11226–11230. IEEE.

Yuhang Yang, Yizhou Peng, Hao Huang, Eng Siong Chng, and Xionghu Zhong. 2024c. Adapting openai's whisper for speech recognition on code-switch mandarin-english seame and asru2019 datasets. In *2024 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1–6. IEEE.

Shuaishuai Ye, Shunfei Chen, Xinhui Hu, and Xinkang Xu. 2024. Sc-moe: Switch conformer mixture of experts for unified streaming and non-streaming code-switching asr. *arXiv preprint arXiv:2406.18021*.

Tengfei Yu, Xuebo Liu, Liang Ding, Kehai Chen, Dacheng Tao, and Min Zhang. 2024. Speech sense disambiguation: Tackling homophone ambiguity in end-to-end speech translation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8020–8035.

Shuai Zhang, Jiangyan Yi, Zhengkun Tian, Jianhua Tao, Yu Ting Yeung, and Liqun Deng. 2022. Reducing language context confusion for end-to-end code-switching automatic speech recognition. *arXiv preprint arXiv:2201.12155*.