Post-ASR Correction in Hindi: Comparing Language Models and Large Language Models in Low-Resource Scenarios

Rishabh Kumar[1], Amrith Krishna[2], Ganesh Ramakrishnan[1], Preethi Jyothi[1]

[1]IIT Bombay, [2]Oriflow

Correspondence: {krrishabh, ganesh, pjyothi}@cse.iitb.ac.in , ak2329@cantab.ac.uk

Abstract

Automatic Speech Recognition (ASR) systems for low-resource languages like Hindi often produce erroneous transcripts due to limited annotated data and linguistic complexity. Post-ASR correction using language models (LMs) and large language models (LLMs) offers a promising approach to improve transcription quality. In this work, we compare fine-tuned LMs (mT5, ByT5), fine-tuned LLMs (Nanda 10B), and instruction-tuned LLMs (GPT-4o-mini, LLaMA variants) for post-ASR correction in Hindi. Our findings reveal that smaller, fine-tuned models consistently outperform larger LLMs in both fine-tuning and in-context learning (ICL) settings. We observe a n-shaped inverse scaling trend under zero-shot ICL, where mid-sized LLMs degrade performance before marginal recovery at extreme scales, yet still fall short of fine-tuned models. ByT5 is more effective for character-level corrections such as transliteration and word segmentation, while mT5 handles broader semantic inconsistencies. We also identify performance drops in out-of-domain settings and propose mitigation strategies to preserve domain fidelity. In particular, we observe similar trends in Marathi and Telugu, indicating the broader applicability of our findings across low-resource Indian languages.

1 Introduction

Automatic Speech Recognition (ASR) systems enable seamless human-computer interaction (Zierau et al., 2023), especially in linguistically diverse countries such as India. ASR technology is increasingly adopted across domains such as agriculture, education, e-commerce, and governance, helping to bridge digital accessibility gaps (Javed et al., 2022; Bhogale et al., 2023b). However, building robust ASR systems for Hindi, the most widely spoken Indian language, remains a significant challenge due to its low-resource nature, limited availability of high-quality annotated speech data (Adiga et al., 2021), and complex linguistic characteristics, including regional variations, code-mixing, and orthographic diversity (Kachru, 2006; Kumar et al., 2025).

To address these challenges, post-ASR correction has emerged as an effective strategy (Kumar et al., 2024; Ma et al., 2025). It involves using LMs trained in large text-only corpora, which are more widely available than speech-text pairs, to refine noisy ASR outputs in low-resource languages such as Hindi (Kumar et al., 2022). This task can be framed as a high-overlap text-editing problem (Malmi et al., 2022), where the goal is to minimally modify an ASR hypothesis to align it more closely with the correct transcript, handling phonetic, grammatical and semantic errors[1].

Recent advances in pre-trained language models span both smaller models (with $\leq$ 580M parameters), such as mT5 (multilingual) and ByT5 (byte-level) built on the T5 architecture (Raffel et al., 2020; Xue, 2020; Xue et al., 2022), and larger models (with $\geq$ 3B parameters), such as GPT (Brown et al., 2020) and LLaMA (Touvron et al., 2023). These models enable post-ASR error correction through supervised fine-tuning or zero-shot/few-shot ICL (Ma et al., 2025). Although larger models are often expected to generalize better due to their scale and exposure to large training corpora, it remains unclear whether they actually outperform smaller, task-specific models for domain-sensitive post-ASR correction, particularly for low-resource and morphologically rich languages.

This leads to two key research questions:

RQ1: How does model performance scale with size in the context of Hindi ASR post-correction?

RQ2: How do ICL approaches using LLMs compare with fine-tuned smaller models, particularly in handling source-specific ASR errors?

[1]For illustrative examples of Hindi ASR error types, please refer to Appendix B

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 2: Short Papers, pages 636–645
March 24-29, 2026

To answer these questions, we performed a systematic comparison of fine-tuned language models and LLMs in both fine-tuned and ICL configurations. Our study benchmarks these models with the Lahaja Hindi ASR dataset (Javed et al., 2024a) using ASR hypotheses generated by open-source models such as IndicWav2vec (Javed et al., 2022) and IndicConformer (Javed et al., 2024a).

We find that smaller fine-tuned models consistently outperform much larger LLMs in both in-domain and out-of-domain scenarios. Surprisingly, we also observe an n-shaped inverse scaling trend in zero-shot ICL settings when plotting the word error rate (WER), where mid-sized LLaMA models (3B-10B) degrade performance compared to both smaller and extremely large models, yet even the largest models fail to match the performance of fine-tuned mT5. This highlights the importance of source-specific inductive biases in modeling ASR errors over general-purpose linguistic knowledge.

We further show that ByT5 is especially effective at correcting character-level errors, such as transliteration mistakes, numeric misrecognitions, and compound word splits. At the same time, mT5 better handles broader semantic inconsistencies and domain-level shifts. Preliminary experiments on Marathi and Telugu ASR outputs also reveal similar trends, indicating that our findings may generalize to other low-resource Indian languages[2].

Our key contributions are:

1. We observed an **inverse scaling trend** in Hindi post-ASR correction, where mid-sized LLMs underperform compared to both smaller LMs and larger LLMs under zero-shot ICL.

2. **Systematic benchmarking of fine-tuned LMs and instruction-tuned LLMs**, demonstrating that fine-tuned small models (mT5, ByT5) significantly outperform larger LLMs such as the GPT-4o mini and LLaMA variants in both ICL and fine-tuning settings.

3. **Granular error-type analysis**, showing ByT5's strength in fine-grained character-level corrections, and mT5's robustness in semantic error correction and domain generalization.

4. **Proposal and evaluation of mitigation strategies for domain adaptation** in ASR correction, including domain-replicative training and in-domain/out-of-domain data mixing.

5. **Multi-language generalizability**, empirically demonstrating that our post-ASR correction approach works across multiple Indic languages, including Hindi, Marathi, and Telugu.

[2]Code and Model: https://github.com/cyfer0618/Post-ASR-Correction-in-Hindi

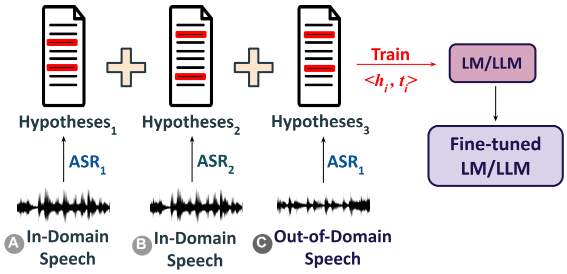

Figure 1: Overview of the data preparation process illustrating how in-domain and out-of-domain speech are used for fine-tuning LM and LLM models.

2 Methodology

We frame post-ASR correction as a text-editing task, where the goal is to transform a noisy ASR hypothesis into a corrected transcript using LMs or LLMs. This section outlines our dataset formulation, training setup, and the strategy used to handle domain variation in low-resource settings.

Let $D^{id}_{train} = \{(a_i, t_i) \mid 1 \leq i \leq n\}$ denote an in-domain speech-text dataset (share similar speaker distributions, topics, vocabulary, and contextual characteristics with the target evaluation set), where a_i is a speech utterance and t_i is the corresponding ground truth transcript. This dataset is used to train an ASR model A^{id}_1. To create training data for post-ASR correction, we decode the speech with A^{id}_1, generating a 1-best hypothesis h_i for each a_i, resulting in a dataset $H^{id}_{train} = \{(h_i, t_i) \mid 1 \leq i \leq n\}$. Here, h_i serves as the noisy input and t_i as the reference transcript.

Moreover, due to the limited availability of in-domain speech-text data, we also consider out-of-domain (*OOD*) datasets to augment training. Let D^{ood}_{train} be a dataset with different characteristics (differ in speaker distribution, style, topic), used to train another ASR model A^{ood}_2. The resulting hypotheses form H^{id}_{train} and H^{ood}_{train} as shown in Figure 1.

We fine-tune both the LMs and LLMs in H^{id}_{train} and H^{ood}_{train}, enabling them to learn typical ASR error patterns and their corrections. At inference time, ASR hypotheses from a held-out set D^{id}_{test} are corrected using these models. In addition, we also use the ASR hypotheses H^{id}_{train} for the evaluation of the ICL.

Training Dataset	Dataset Size	IndicWav2Vec ASR Hyp.	IndicWav2Vec ByT5	IndicWav2Vec mT5	IndicWav2Vec LLaMA	IndicConformer ASR Hyp.	IndicConformer ByT5	IndicConformer mT5	IndicConformer LLaMA
D1: In-Domain Speech with ASR model	63500		**24.17**	32.92	76.72		18.22	**17.50**	76.04
D2: + In-Domain Speech with Diff. ASR model	127306	28.60	**26.62**	29.09	**27.80**	18.02	18.07	**16.75**	23.24
D3: + Out-of-Domain Speech with ASR model	1021472		**25.14**	**23.74**	**26.03**		**17.52**	**16.31**	21.49

Table 1: WER (%) comparison for various fintuned LMs (ByT5-small, mT5-base) and LLM (LLaMA).

3 Experiment and Results

Datasets: We evaluate post-ASR correction models using the Lahaja dataset (Javed et al., 2024a), which comprises 12.5 hours of Hindi speech from 132 speakers across 83 districts. It includes read, extempore, and conversational speech, making it suitable for evaluating domain-specific correction performance. For fine-tuning, we use the IndicVoice corpus (Javed et al., 2024b) (65 hours, 287 speakers), selected for its domain and vocabulary overlap with Lahaja. To assess generalization, we include two out-of-domain (*OOD*) datasets: Kathbath (Javed et al., 2023) (read speech) and Shrutilipi (Bhogale et al., 2023a) (conversational radio broadcasts), offering diverse linguistic and stylistic characteristics.

Baseline: We generate ASR hypotheses using two open-source Hindi ASR models. **IndicWav2Vec** (Javed et al., 2022), based on the wav2vec 2.0 architecture. **IndicConformer** (Javed et al., 2024a), based on the conformer architecture. As detailed in Appendix D, preliminary comparisons showed these models consistently outperform other Hindi ASRs in both WER and CER in the Lahaja dataset. These hypotheses serve as the input for post-ASR correction models.

Model Configurations: We compare the following LMs and LLMs:

1. **ByT5** (Xue et al., 2022): A tokenizer-free, byte-level T5 variant, effective for character-level corrections.
2. **mT5** (Xue, 2020): A multilingual T5 variant using SentencePiece tokenization, trained on Common Crawl data including Hindi.
3. **LLaMA-3-Nanda-10B-Chat** (Choudhury et al., 2025): A 10B bilingual LLM adapted from LLaMA-3-7B with continued pretraining on 65B Hindi tokens.
4. **GPT-4o mini** (OpenAI, 2024): A closed-weight instruction-tuned model, evaluated in zero and few-shot ICL settings.

ByT5 and mT5 are fine-tuned on the 1-best ASR hypotheses paired with reference transcripts. GPT-4o-mini is evaluated using a few-shot prompt[3], with examples drawn from IndicVoice through random sampling and sentence embedding similarity (Joshi et al., 2023) to ensure contextual relevance. We also carried out a pilot experiment with n = 5 hypotheses for ByT5 in D1, observing a WER of 45, indicating that while multi-hypothesis inputs may provide additional signal, the improvements are limited without targeted modeling.

3.1 Results of Fine-tuned LMs and LLMs

Training Dataset	Dataset Size	IndicWav2Vec ByT5	IndicWav2Vec mT5	IndicConformer ByT5	IndicConformer mT5
D1: IndicVoice [IC]	63500	**24.17**	32.92	18.22	**17.50**
IndicVoice [W2V]		**26.00**	**26.67**	18.37	**16.81**
Shrutilipi [IC]	127306	31.37	29.67	24.18	22.19
Kathbath + Shrutilipi [IC]		30.45	27.76	23.34	19.48
Shrutilipi [W2V]		30.10	29.96	25.04	22.55
Kathbath + Shrutilipi [W2V]		28.84	29.05	22.30	20.75
D2: IndicVoice [IC+W2V]		**26.62**	29.09	18.07	**16.75**
D3: **D2** + *OOD* [IC]	1021472	**25.14**	**23.74**	**17.52**	**16.31**
D2 + *OOD* [W2V]		**23.66**	**22.97**	**17.55**	**16.45**
D4: **D2** + *OOD* [IC + W2V]		**23.36**	**23.00**	**17.46**	**16.17**

Table 2: Performance comparison of ByT5-small (ByT5) and mT5-base (mT5) models on the Lahaja test dataset trained with different training datasets. The WER (%) of the IndicWav2Vec (W2V) model is 28.6, while the IndicConformer (IC) model is 18.02 . (*OOD* = Out-of-Domain)

Table 1 reports WER (%) for fine-tuned post-correction models under progressively richer training conditions. Moving from D1 (in-domain, single ASR) to D3 (in-domain + out-of-domain, single ASR) consistently reduces WER for both ByT5 and mT5 across the hypotheses of IndicWav2Vec and IndicConformer (e.g., on IndicConformer hypotheses, mT5 improves from 17.50 in D1 to 16.31 in D3). This trend suggests that incorporating out-of-domain hypotheses exposes the model to a broader range of ASR error patterns and improves robustness. However, since D3 also increases the number of training pairs, we interpret these gains as the combined effect of additional training scale and

[3] For prompt, see Appendix H

hypothesis diversity, and we further analyze this trade-off in subsequent experiments.

Table 2 isolates the contribution of (i) the ASR source used to generate hypotheses (IndicConformer vs. IndicWav2Vec) and (ii) in-domain vs. out-of-domain training composition. Adding more diverse data improves correction (Kathbath+Shrutilipi vs. Shrutilipi), and mixing hypotheses from multiple ASR systems further helps (IndicVoice IC+W2V). When OOD data are added, the multi-ASR setting (D4: D2+*OOD*) produces the best mT5 result on IndicConformer, slightly better than single-ASR *OOD* training, supporting multi-ASR, multi-domain augmentation for robust post-ASR correction.

3.2 Results of ICL

Experiment	Shots	IndicWav2Vec	IndicConformer
-	0-Shot	28.60 → 31.77	18.02 → 25.14
SE Similarity	1-Shot	28.60 → 29.22	18.02 → 22.88
	3-Shot	28.60 → 28.18	18.02 → 22.04
	5-Shot	**28.60 → 27.14**	18.02 → 20.89

Table 3: WER (%) comparison for various shot settings using GPT-4o mini (ICL)

In Table 3, we evaluate the ICL capability of a LLM, GPT-4o mini. We assess post-ASR correction in both zero-shot, 1-shot and few-shot settings. Our findings demonstrate the adaptability of few-shot learning, leveraging sentence embeddings (SE) to improve ASR correction. However, in the case of IndicConformer, this approach resulted in an increase in the WER of the ASR hypothesis.

Smaller fine-tuned models, such as **mT5** and **ByT5** consistently, outperform larger LLMs such as GPT-4o-mini and LLaMA-3. This suggests that task-specific inductive bias and domain adaptation are more effective than sheer model scale for post-ASR correction in low-resource settings[4].

We also evaluated the impact of model size on Hindi post-ASR correction under a zero-shot ICL setup, relying solely on the pre-trained knowledge of each model without additional fine-tuning. As shown in Figure 2, increasing the number of parameters (3.1 1B → 3.2 3B → 3.1 8B → Nanda-10B-Chat 10B → 3.3 70B) reveals an n-shaped trend in the word error rate (WER): performance improves initially, then worsens and may slightly recover at higher scales. This inverse scaling behavior indicates that larger models do not necessarily guarantee better correction accuracy.

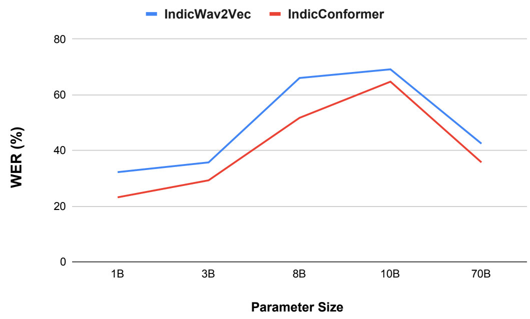

Figure 2: Inverse scaling phenomenon in Hindi post-ASR correction across varying LLaMA model sizes.

We further analyzed why this inverse-scaling trend emerges and found three consistent failure modes. First, **an over-correction effect becomes more pronounced with scale in ICL**: larger LLMs more often "edit beyond necessity", introducing hallucinated or stylistic substitutions that are semantically plausible but lexically misaligned with the reference, thereby increasing WER. Second, we observe **clear error-type sensitivity differences**: fine-tuned ByT5/mT5 models tend to correct local perturbations (e.g., transliteration noise, word segmentation/merging) more conservatively and faithfully, whereas ICL LLMs apply broader generalizations that can overshoot the intended minimal correction. Third, **domain fidelity degrades with scale**: larger LLMs exhibit stronger out-of-domain drift, amplifying corrections that move the output away from in-domain lexical choices, consistent with known brittleness in generative editing behavior under distribution shift.

4 Ablation Study

Training Dataset	Ratio	Dataset Size	IndicWav2Vec ByT5	IndicWav2Vec mT5	IndicConformer ByT5	IndicConformer mT5
D2 + *OOD* [IC]	3:7	381415	22.44	25.89	17.19	**16.03**
D2 + *OOD* [W2V]			**22.26**	25.81	17.65	16.51
D2 + *OOD* [IC]	2:8	571962	22.32	26.51	17.74	**16.02**
D2 + *OOD* [W2V]			22.29	26.68	17.58	16.62

Table 4: Evaluation of post-ASR correction on Lahaja dataset mixing the in-domain and out-of-domain (*OOD*) dataset in fixed ratio.

We observe residual degradation at higher out-of-domain proportions, highlighting the limitations of fixed-ratio scheduling alone. Table 4 shows

[4] For additional analysis, see Appendix E.

that a 3:7 sampling ratio from in-domain to out-of-domain per batch yields the best post-ASR correction performance, suggesting that batch composition is key to retain in-domain error patterns. This points to the need to incorporate techniques such as domain-aware regularization fine-tuning to improve domain fidelity in low-resource settings.

Experiments	IW → CW	CW → IW	No Change
Word Segmentation	224	216	498
Compound Words	75	74	215
English Words	637	283	3180
English Number	7	17	131
Hindi Number	36	24	94
Underrepresented Character	2254	1129	3296

Table 5: Analysis of errors in the Lahaja Dataset using mT5=16.17 model train on D4 dataset. IW = Incorrect Word and CW = Correct Word

Experiments	IW → CW	CW → IW	No Change
Word Segmentation	241	253	722
Compound Words	84	97	206
English Words	730	456	3087
English Number	19	22	119
Hindi Number	33	28	97
Underrepresented Character	2287	1798	3263

Table 6: Analysis of errors in the Lahaja Dataset using ByT5=17.46 model train on D4 dataset. IW = Incorrect Word and CW = Correct Word

Table 5 and Table 6 show that ByT5 consistently corrects more character-centric errors, code-mixed tokens, compound-word splits, word-segmentation errors, numeric misrecognitions, and under-represented graphemes than mT5. This comes from ByT5's byte-level tokenization, which provides finer granularity for detecting single-character perturbations. In contrast, mT5's sub-word vocabulary provides stronger semantic coverage, but makes it less sensitive to very fine-grained character variations.

ByT5-small	ByT5-base	mT5-small	mT5-base	LLaMA	GPT-4o mini
2.29	2.79	**0.97**	**1.84**	10.17	2.03

Table 7: Latency (in seconds) of different models for post-ASR correction.

In Table 7, we summarize the latency of different LMs/LLMs, indicating that *mt5-small* performed the fastest post-ASR correction. Thus, mT5 achieves significant performance gains while being significantly faster than larger LLMs.

Language	Hypothesis	ByT5-small	ByT5-base	mT5-small	mT5-base
Marathi	**25.55**	26.32	26.02	25.76	**25.12**
Telugu	**23.28**	24.51	24.72	**22.68**	**22.05**

Table 8: Evaluation of post-ASR correction on Marathi and Telugu IndicTTS datasets.

4.1 Additional Languages

Our approach was tailored to Hindi, focusing on lexical and multiword interventions involving both lexical and morphemic-level knowledge. However, we have conducted evaluations for Marathi and Telugu as well. Table 8 shows the performance of various post-correction models on Marathi and Telugu subsets of the IndicTTS dataset. We compare ASR hypotheses against corrected outputs from ByT5 and mT5 models of both small and base sizes. The mT5-base model achieves a lower WER across both languages. We use the IndicTTS dataset for this evaluation as it closely resembles the Lahaja dataset in linguistic characteristics and is in-domain with the IndicVoice dataset, ensuring consistent domain relevance for low-resource ASR evaluation.

5 Conclusion

In this work, we explore the effectiveness of LMs and LLMs for post-ASR correction in Hindi, highlighting the surprising result that smaller, fine-tuned models such as mT5 and ByT5 consistently outperform much larger LLMs like GPT-4o-mini and LLaMA variants. Our findings reveal a n-shaped inverse scaling trend, observed under zero-shot in-context learning, where increasing model size initially degrades performance before marginal improvements at extreme scales, yet still falls short of the smaller models. ByT5 excels at fine-grained character-level corrections, while mT5 is more effective at capturing broader semantic inconsistencies. We also identify significant performance degradation in high out-of-domain settings and propose mitigation strategies to preserve domain-specific fidelity in post-ASR correction. Preliminary experiments on Marathi and Telugu also reflect similar patterns, indicating that our findings may generalize across other low-resource Indian languages. These results underscore the importance of source-specific inductive biases and demonstrate that lightweight, fine-tuned models are often better suited than general-purpose LLMs for improving ASR quality in such contexts.

6 Acknowledgments

We acknowledge BharatGen for providing compute and resources. We deeply appreciate the insights on this work from Advait Dhopeshwarkar, Abhinav Painuli, and Chriss Philip Saji. We thank Josh Talks Private Limited for providing a travel grant to support the first author's attendance at EACL 2026.

Limitations

We acknowledge the following limitations of our work:

- This study focuses mainly on Hindi. Although preliminary evaluations are conducted in Marathi and Telugu, they lack detailed analysis. In addition, the absence of linguistic experts for these languages limits the depth of error categorization and qualitative interpretations.
- ICL results are limited to GPT-4o mini and evaluated under only a few-shot setting and SE-based prompting. A comparison to GPT-4o is missing due to limited funds.

References

Devaraja Adiga, Rishabh Kumar, Amrith Krishna, Preethi Jyothi, Ganesh Ramakrishnan, and Pawan Goyal. 2021. Automatic speech recognition in sanskrit: A new speech corpus and modelling insights. *arXiv preprint arXiv:2106.05852*.

Loïc Barrault, Yu-An Chung, Mariano Coria Meglioli, David Dale, Ning Dong, Mark Duppenthaler, Paul-Ambroise Duquenne, Brian Ellis, Hady Elsahar, Justin Haaheim, et al. 2023. Seamless: Multilingual expressive and streaming speech translation. *arXiv preprint arXiv:2312.05187*.

Kaushal Bhogale, Abhigyan Raman, Tahir Javed, Sumanth Doddapaneni, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra. 2023a. Effectiveness of mining audio and text pairs from public data for improving asr systems for low-resource languages. In *Icassp 2023-2023 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 1–5. IEEE.

Kaushal Santosh Bhogale, Sai Sundaresan, Abhigyan Raman, Tahir Javed, Mitesh M Khapra, and Pratyush Kumar. 2023b. Vistaar: Diverse benchmarks and training sets for indian language asr. *arXiv preprint arXiv:2305.15386*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Monojit Choudhury, Shivam Chauhan, Rocktim Jyoti Das, Dhruv Sahnan, Xudong Han, Haonan Li, Aaryamonvikram Singh, Alok Anil Jadhav, Utkarsh Agarwal, Mukund Choudhary, et al. 2025. Llama-3-nanda-10b-chat: An open generative large language model for hindi. *arXiv preprint arXiv:2504.06011*.

Yassir Fathullah, Chunyang Wu, Egor Lakomkin, Junteng Jia, Yuan Shangguan, Ke Li, Jinxi Guo, Wenhan Xiong, Jay Mahadeokar, Ozlem Kalinli, et al. 2024. Prompting large language models with speech recognition abilities. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13351–13355. IEEE.

Tahir Javed, Kaushal Bhogale, Abhigyan Raman, Pratyush Kumar, Anoop Kunchukuttan, and Mitesh M Khapra. 2023. Indicsuperb: A speech processing universal performance benchmark for indian languages. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12942–12950.

Tahir Javed, Sumanth Doddapaneni, Abhigyan Raman, Kaushal Santosh Bhogale, Gowtham Ramesh, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra. 2022. Towards building asr systems for the next billion users. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10813–10821.

Tahir Javed, Janki Nawale, Sakshi Joshi, Eldho George, Kaushal Bhogale, Deovrat Mehendale, and Mitesh M Khapra. 2024a. Lahaja: A robust multi-accent benchmark for evaluating hindi asr systems. *arXiv preprint arXiv:2408.11440*.

Tahir Javed, Janki Atul Nawale, Eldho Ittan George, Sakshi Joshi, Kaushal Santosh Bhogale, Deovrat Mehendale, Ishvinder Virender Sethi, Aparna Ananthanarayanan, Hafsah Faquih, Pratiti Palit, et al. 2024b. Indicvoices: Towards building an inclusive multilingual speech dataset for indian languages. *arXiv preprint arXiv:2403.01926*.

Ananya Joshi, Aditi Kajale, Janhavi Gadre, Samruddhi Deode, and Raviraj Joshi. 2023. L3cube-mahasbert and hindsbert: Sentence bert models and benchmarking bert sentence representations for hindi and marathi. In *Science and Information Conference*, pages 1184–1199. Springer.

Yamuna Kachru. 2006. Hindi.

NJ Karthika, Adyasha Patra, Nagasai Saketh Naidu, Arnab Bhattacharya, Ganesh Ramakrishnan, and Chaitali Dangarikar. 2025. Semantically cohesive word grouping in indian languages. *arXiv preprint arXiv:2501.03988*.

Rishabh Kumar, Devaraja Adiga, Rishav Ranjan, Amrith Krishna, Ganesh Ramakrishnan, Pawan Goyal, and Preethi Jyothi. 2022. Linguistically informed post-processing for asr error correction in sanskrit. In *INTERSPEECH*, pages 2293–2297.

Rishabh Kumar, Devaraja Adiga, Rishav Ranjan, Amrith Krishna, Ganesh Ramakrishnan, Pawan Goyal, and Preethi Jyothi. 2025. Linguistically informed automatic speech recognition in sanskrit. *Computer Speech & Language*, page 101861.

Rishabh Kumar, Sabyasachi Ghosh, and Ganesh Ramakrishnan. 2024. Beyond common words: Enhancing asr cross-lingual proper noun recognition using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6821–6828.

Sheng Li, Chen Chen, Chin Yuen Kwok, Chenhui Chu, Eng Siong Chng, and Hisashi Kawai. 2024. Investigating asr error correction with large language model and multilingual 1-best hypotheses. In *Proc. Interspeech*, pages 1315–1319.

Vasista Sai Lodagala, Sreyan Ghosh, and Srinivasan Umesh. 2023. data2vec-aqc: Search for the right teaching assistant in the teacher-student training setup. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Rao Ma, Mengjie Qian, Mark Gales, and Kate Knill. 2025. Asr error correction using large language models. *IEEE Transactions on Audio, Speech and Language Processing*.

Rao Ma, Mengjie Qian, Potsawee Manakul, Mark Gales, and Kate Knill. 2023. Can generative large language models perform asr error correction? *arXiv preprint arXiv:2307.04172*.

Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar, and Aliaksei Severyn. 2022. Text generation with text-editing models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*, pages 1–7, Seattle, United States. Association for Computational Linguistics.

Ashish Mittal, Darshan Prabhu, Sunita Sarawagi, and Preethi Jyothi. 2024. Salsa: Speedy asr-llm synchronous aggregation. *arXiv preprint arXiv:2408.16542*.

OpenAI. 2024. Gpt-4o mini: Advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. Accessed: 2025-10-07.

Jing Pan, Jian Wu, Yashesh Gaur, Sunit Sivasankaran, Zhuo Chen, Shujie Liu, and Jinyu Li. 2023. Cosmic: Data efficient instruction-tuning for speech in-context learning. *arXiv preprint arXiv:2311.02248*.

Srijith Radhakrishnan, Chao-Han Huck Yang, Sumeer Ahmad Khan, Rohit Kumar, Narsis A Kiani, David Gomez-Cabrero, and Jesper N Tegner. 2023. Whispering llama: A cross-modal generative error correction framework for speech recognition. *arXiv preprint arXiv:2310.06434*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

L Xue. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Naim Zierau, Christian Hildebrand, Anouk Bergner, Francesc Busquet, Anuschka Schmitt, and Jan Marco Leimeister. 2023. Voice bots on the frontline: Voice-based interfaces enhance flow-like consumer experiences & boost service outcomes. *Journal of the Academy of Marketing Science*, 51(4):823–842.

A Appendix

In the Appendix, we provide:

1. Section B: Illustrative Example of Hindi ASR Errors
2. Section C: Related Works
3. Section D: Model Comparison
4. Section E: Additional Analysis
   (a) Section E.1: LM/LLM comparison
   (b) Section E.2: Effect of domain-specific regularization
5. Section F: Compound Word Error Detection Algorithm
6. Section G: Compute Infrastructure
7. Section H: Prompts

GT: rathayātrā ke lie jānabūjhakara vana ṭūrisṭa dvārā taiṃtālīsa minaṭa kī derī kī gaī hai
N-GT: rathayātrā ke lie jānabūjhakara vana [One] ṭūrisṭa [Tourist] dvārā taiṃtālīsa minaṭa kī derī kī gaī hai
Hypothesis: **ratha yātrā** ke lie jānabūjhakara **vāna** **ṭyūresṭa** dvārā **taitālīsa** minaṭa kī derī kī **gaīhai**
Transcript: **rathayātrā** ke lie jānabūjhakara **vana** **ṭūrisṭa** dvārā **taitālīsa** minaṭa kī derī kī **gaī hai**

English Word ['ṭyūresṭa']
Number ['vāna', 'taitālīsa']
Word Segmentation ['gaīhai']
Compound Words ['ratha yātrā']
Under Represented Characters ['taitālīsa']
} ASR Error Types

Figure 3: Example of ASR hypothesis errors in Hindi, categorized by error types: English word transliteration (*ṭyūresṭa*), number transcription (*vāna*, *taitālīsa*), word segmentation (*gaīhai*), compound word splitting (*ratha yātrā*), and underrepresented character errors (*taitālīsa*).

B Illustrative Example of Hindi ASR Errors

Compound words, such as (/rathayātrā/), which refers to an annual Hindu chariot festival, erroneously the word can split into (/ratha yātrā/) (ratha means chariot, yātrā means travel, journey), thus altering their meaning. Word segmentation errors are also common, particularly with derivational and infectional word groups (Karthika et al., 2025), where phrases like (/kē liē/) or (/gaī hai/) can become incorrectly merged. Misrecognition of numbers further complicates Hindi ASR. For instance, the English numbers, such as "one" (expected as (/ vana /)), are often phonetically transcribed as (/ vāna /), and native Hindi numbers, like (/taitālīsa/) (taitālīsa means forty three), can be distorted due to inadequate training data. Code-mixed content, such as(/ rathayātrā kē liē jānabūjhakara vana ṭūrisṭa dvārā taitālīsa minaṭa kī dērī kī gaī hai /)[5], further complicates ASR tasks, as systems struggle to manage transitions between Hindi and English seamlessly. Lastly, phonetic and orthographic variability arising from regional accents, dialects, and optional diacritics or conjunct consonants leads to systematic recognition errors as shown in Figure 3.

C Related Works

LLMs have been integrated into ASR systems through various approaches. ASR error correction uses LLM to re-score the N-best lists of potential transcriptions, refining the predictions (Ma et al., 2023; Radhakrishnan et al., 2023). Speech ICL fine-tunes LLMs with speech inputs, enabling them to handle diverse tasks (Kumar et al., 2024), while deep LLM fusion (Fathullah et al., 2024) employs LLMs as decoders in ASR architectures, integrating language modelling capabilities through mechanisms like gated cross-attention. However, both ICL speech (Pan et al., 2023) and deep LLM fusion (Fathullah et al., 2024) are computationally intensive, requiring significant resources and large labelled speech datasets, which are scarce for low-resource languages such as Hindi. Similarly, LLM re-scoring of N-best lists often underperforms compared to using a single 1-best hypothesis (Li et al., 2024; Kumar et al., 2024), which is sufficient to address common errors such as word segmentation, underrepresented characters, and compound word handling.

[5]means "For the chariot procession, a tourist intentionally caused a delay of forty-three minutes".

D Model Comparison

Model	WER (%)	CER (%)
IndicWav2vec (Javed et al., 2022)	28.605	10.54
IndicWhisper (Bhogale et al., 2023b)	32.17	19.86
IndicConformer (Javed et al., 2024a)	18.015	6.458
Seamless M4T (Barrault et al., 2023)	52.63	29.89
data2vec_aqc (Lodagala et al., 2023)	29.63	10.6
SALSA (Mittal et al., 2024)	74.43	54.54

Table 9: Performance Comparison of Open-Source Hindi ASR Models on Hindi Lahaja dataset

Table 9 presents a comparative evaluation of open-source Hindi ASR models on the Hindi Lahaja dataset in terms of WER and Character Error Rate (CER). Among the evaluated systems, IndicConformer (Javed et al., 2024a) achieves the best performance with a WER of 18.015% and a CER of 6.458%, significantly outperforming other models. IndicWav2Vec (Javed et al., 2022) also demonstrates strong performance with a WER of 28.605% and CER of 10.54%, while IndicWhisper and Seamless M4T show higher error rates, reflecting their limitations in capturing the linguistic nuances of Hindi. Notably, SALSA (Mittal et al., 2024) performs the worst, with a WER of 74.43% and CER of 54.54%, suggesting it is less suitable for Hindi ASR. These results reinforce the effectiveness of IndicConformer as a robust baseline for downstream post-ASR correction tasks in Hindi.

Moreover, Table 1 demonstrates how the use of larger and diverse training datasets improves model. Specifically, IndicWav2Vec and IndicConformer, combined with LM like ByT5 and mT5, exhibit marked improvements in the Lahaja test set, underscoring the effectiveness of leveraging diverse error patterns for ASR post correction training. Although fine-tuned LLaMA decline the ASR hypothesis quality.

E Additional Analysis

E.1 LM/LLM comparison

We have experimented with LMs (mT5 and ByT5) and LLMs (LLaMA-3-Nanda-10B-Chat) under comparable condition in terms of Hindi token used for pre-training them in absolute terms, relative terms to their size, and relative to overall presence of Hindi within the rest of the languages present to pre-train the model. We find that our observation still holds. Given that many experiments have shown that the fine-tuned model substantially updates their weights and hence the performance improvement is substantial, we empirically observe that finetuning has substantially improved the performance.

Experiment	Shots	IndicWav2Vec	IndicConformer
-	0-Shot	28.60 → 31.77	18.02 → 25.14
Random	1-Shot	28.60 → 30.95	18.02 → 24.51
	3-Shot	28.60 → 29.84	18.02 → 22.13
	5-Shot	28.60 → 29.27	18.02 → 22.19
SE Similarity	1-Shot	28.60 → 29.22	18.02 → 22.88
	3-Shot	28.60 → 28.18	18.02 → 22.04
	5-Shot	**28.60 → 27.14**	18.02 → 20.89

Table 10: WER (%) Comparison for Various Shot Settings using GPT-4o mini (ICL)

The dataset (60K-1M examples) is large for T5 families (ByT5, mT5) but relatively small for a model of 10B parameters. With 60K examples, the model does not converge towards high overlap text editing behaviour and instead continues to behave like a generative LLM.

E.2 Effect of domain-specific regularization

While fixed-ratio training helps mitigate domain forgetting by ensuring consistent exposure to limited in-domain data, an open research question remains: Can incorporating regularization techniques alongside fixed-ratio training further enhance model retention of in-domain knowledge during post-ASR correction? As shown in Table 11, fine-tuning the ByT5 and mT5 variants with a controlled ratio from the in-domain to the out-of-domain results in noticeable gains in correction performance across both IndicWav2Vec and IndicConformer outputs. However, despite these improvements, subtle performance degradation is still observed in some configurations with higher proportions out-of-domain (OOD). This suggests that additional mechanisms, such as domain-aware regularization, rehearsal-based constraints, or importance-weighted loss, could potentially reinforce in-domain retention even further. Investigating such methods in conjunction with fixed-ratio scheduling presents a promising direction for improving robustness and domain fidelity in low-resource post-ASR correction.

F Compound Word Error Detection Algorithm

To systematically identify compound word errors in ASR hypotheses, we propose an algorithm that leverages a trie-based structure built from a vocabulary dictionary. As outlined in Algorithm 1, the process involves tokenizing both the ground truth (GT) and hypothesis (Hyp) utterances, generating valid substrings from GT tokens, and validating these against the constructed trie. The algorithm then checks whether the valid compound words from the ground truth appear intact in the hypothesis. If a compound word is absent or split incorrectly in the hypothesis, it is flagged as an error. This approach is particularly effective for detecting errors in morphologically rich languages like Hindi, where compound word splitting significantly alters meaning. By identifying such errors, the algorithm supports more fine-grained post-ASR correction and helps evaluate model performance on preserving lexical integrity.

G Compute Infrastructure

Compute details: For all our pre-training and fine-tuning experiments, we used two NVIDIA A100-SXM4-80GB GPUs. Each training requires 4-48 hours.
Software and Packages details: We implement all our models in PyTorch[6]
Models
mT5: mT5-small (300M parameters), mT5-base (580M parameters)
ByT5: ByT5-small (300M parameters), ByT5-base (580M parameters)
Nanda: LLaMA3-10B
GPT-4o mini: 8B parameter

H Prompt

[6] https://pytorch.org/

Training Dataset	Ratio	Dataset Size	byt5-small		byt5-base		mt5-small		mt5-base	
			W2V	IC	W2V	IC	W2V	IC	W2V	IC
IndicVoice [IC+W2V] + other ASR dataset [IC]	3:7	381415	0.2620	0.1778	0.2244	0.1719	0.2817	0.1689	0.2589	**0.1603**
IndicVoice [IC+W2V] + other ASR dataset [W2V]			0.2300	0.1760	**0.2226**	0.1765	0.2600	0.1713	0.2581	0.1651
IndicVoice [IC+W2V] + other ASR dataset [IC]	2:8	571962	0.2358	0.1729	0.2232	0.1774	0.2735	0.1688	0.2651	**0.1602**
IndicVoice [IC+W2V] + other ASR dataset [W2V]			0.2310	0.1787	0.2229	0.1758	0.2591	0.1758	0.2668	0.1662
IndicVoice [IC+W2V] + other ASR dataset [IC]	1:9	993155	0.2442	0.1774	0.2443	0.1774	0.2512	0.1710	0.2588	**0.1614**
IndicVoice [IC+W2V] + other ASR dataset [W2V]			0.2333	0.1829	0.2234	0.1762	0.2388	0.1712	0.2549	0.1638

Table 11: Evaluation of post-ASR correction mixing the in-domain and out-of-domain dataset in fixed ratio

Algorithm 1 Detecting Compound Word Errors Using a Trie

Require: Dict: Vocabulary dictionary, GT: Ground Truth utterance , Hyp: Hypothesis utterance

Ensure: Er_{CW}: List of compound word errors

Step 1: Build the Trie
Initialize an empty Trie T
for each word $\in$ Dict **do**
  Traverse T character by character
  if character does not exist in T **then**
    Create a new node
  end if
  Mark the end of word as isEndOfWord $\leftarrow$ True
end for
Step 2: Preprocess Input
Tokenize GT: $GT_{tokens} \leftarrow split(GT)$
Tokenize Hyp: $Hyp_{tokens} \leftarrow split(Hyp)$
Step 3: Generate Substrings
for each word $\in GT_{tokens}$ **do**
  Splits $\leftarrow$ splits(word)
  Store valid splits as $Splits_{valid}$
end for
Step 4: Validate Substrings
for each split $\in Splits_{valid}$ **do**
  if all substrings subsplit $\in$ split exist in T **then**
    Add split to $CompoundWords_{valid}$
  end if
end for
Step 5: Check for Errors
for each word $\in CompoundWords_{valid}$ **do**
  if word $\notin Hyp_{tokens}$ **then**
    Add word to Er_{CW}
  end if
end for
Step 6: Output Results
Save Er_{CW} for further analysis

ChatGPT Prompt

Example 1:
You are given an ASR hypothesis of a spoken utterance. The hypothesis may contain misrecognized words, incorrect word segments, or code-switching mistakes. Your job is to produce the best possible corrected text, relying on your knowledge of grammar and typical usage
Please correct any errors in
1. Incorrect transliteration of English words
2. Incorrect transliteration of English numbers
3. Incorrect transcription of native Hindi numbers
4. Misrecognition of underrepresented characters
5. Splitting of compound words
6. Incorrect word segmentation

There may be more than two errors in the ASR hypothesis. Output only the final corrected output (no extra commentary)
Hypothesis: ratha yātrā ke lie jānabūjhakara vāna ṭyūresṭa dvārā taitālīsa minaṭa kī derī kī gaīhai

Predicted Output: ratha yātrā ke lie jānabūjhakara vana ṭyūrisṭa dvārā taiṃtālīsa minaṭa kī derī kī gaī hai.