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

Anonymous ACL submission

Abstract

Automatic Speech Recognition (ASR) systems for low-resource languages like Hindi often suffer from transcription errors due to limited training data and linguistic complexity. Post-ASR correction is a key strategy to address these issues, particularly given the prevalence of code-mixing, compounding, and segmentation errors. We evaluate both fine-tuned language models (LMs) and large language models (LLMs) for this task, treating it as a high-overlap text editing problem. Experiments on the Lahaja dataset reveal a U-shaped inverse scaling trend: smaller models like mT5 and ByT5 outperform larger LLMs such as LLaMA (1B-70B) and GPT-4o-mini, even in in-context learning (ICL) settings. While ByT5 excels at character-level corrections, mT5 performs better on semantic errors. We also observe significant degradation in out-of-domain settings and propose mitigation strategies to retain domain fidelity. Our results highlight that inductive bias and task alignment often outweigh model scale for effective post-ASR correction in low-resource contexts.

1 Introduction

Automatic Speech Recognition (ASR) systems enable seamless human-computer interaction (Zierau et al., 2023), especially in linguistically diverse countries like India. These systems are increasingly adopted across domains such as agriculture, education, e-commerce, and governance, helping 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 challenging due to its low-resource nature, limited availability of high-quality annotated speech data (Adiga et al., 2021), and complex linguistic characteristics (Kachru, 2006).

Post-ASR correction has shown to improve transcription accuracy by using language models (LM) trained on large text-only corpora, which are often more widely available than speech-text data, for a low-resource language like Hindi (Kumar et al., 2022). Unlike traditional n-gram based LMs, modern LMs such as T5 (Raffel et al., 2020) and Large LMs (LLMs) such as GPT (Brown et al., 2020) and LLaMA (Touvron et al., 2023), capture rich semantic and contextual information, enabling them to effectively tackle both linguistic and domain-specific nuances. The post-ASR correction task typically addresses a wide range of errors arising out of ASR systems, including text edit errors, phonetic, grammatical and higher order semantic errors. LLMs, by virtue of their larger parameter counts and exposure to massive and diverse training data, are expected to generalize better in distributional semantics than traditional language models.

Post-ASR correction can be seen as a text editing task (Malmi et al., 2022), with high overlap in the input and the predicted text. Here, such a model should ideally leverage the linguistic, semantic, and world knowledge encoded in such language models while maintaining an inductive bias toward source-specific error patterns, from an ASR in this case, for better error correction.

The trade-off between source specific inductive bias and inherent LM characteristics leads to two fundamental questions. One, how does model performance scale with size? Two, how competitive are incontext learning methods (ICL) with no parameter updation in comparison to fine tuning models in learning source specific error patterns? ASRs, regardless of the language, typically induce error patterns pertaining to phonetic similarities, homophones, grammatical inconsistencies, contextual misinterpretations,punctuation omissions, etc. Further, Hindi poses unique challenges for ASR development due to its linguistic richness, complex phonetic structure, diverse accents, and significant regional variations[1].

[1] For more detail Appendix A

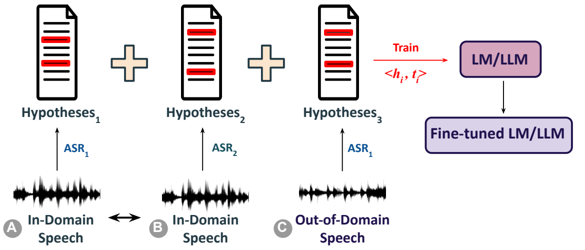

Figure 1: Performance comparison of fine-tuned and ICL-based LM/LLM models on Hindi post-ASR correction.

We address both the fundamental questions from our experiments. We observe that Post-ASR correction in Hindi can be seen as an inverse scaling task, (McKenzie et al., 2023), where model performance degrades as model size increases. More specifically. we observe that the task exhibits a U-shaped scaling where task performance decreases upto a certain model size and then has shown to increase for the largest model we evaluated (Wei et al., 2022). This suggests that smaller models such as mT5 and byT5 with 580 million and 300 million parameters respectively outperform LLMs as high as 70 Billion and even GPT-4o-mini. While these smaller models are fine-tuned, they outperform the LLMs in both fine-tuned and ICL configurations. These results further illustrate the need for having stronger source-error specific inductive biases in the model over the inherent general knowledge that these LMs possess. Here, Llama variants ranging from 1, 3, 8, 10 and 70 billion parameters have shown performance degradation resulting in worse error rates than the original ASR hypothesis.

In this work, we investigate the effectiveness of LLMs for Post-ASR correction. Surprisingly, we observe that fine-tuned T5 variants, with parameter sizes in the few hundred millions, consistently outperform much larger LLMs. We evaluate both ICL and fine-tuned configurations of several LLMs, including the LLaMA family (ranging from 1B to 70B parameters) and GPT-4o-mini. Across these settings, smaller models such as mT5 and ByT5 demonstrate superior performance. Notably, the task exhibits a U-shaped inverse scaling trend, error rates initially increase with model size before improving for the largest models, yet they still fall short of matching the T5 variants. Among the smaller models, mT5 achieves the best overall performance in both in-domain and out-of-domain scenarios, while ByT5 excels at fine-grained character-level corrections, and mT5 is more effective at capturing broader semantic errors.

We benchmark our models on the Hindi Lahaja dataset (Javed et al., 2024a), demonstrating improvements in Word Error Rate (WER) of up to 5.63 for IndicWav2vec (Javed et al., 2022) and 1.85 for IndicConformer (Javed et al., 2024a). Our 1-best hypothesis strategy (Li et al., 2024) proves effective in improving both linguistic and domain-specific transcription accuracy in Hindi.

Contributions:

- We demonstrate that Hindi post-ASR correction exhibits an **inverse scaling effect**, where mid-sized LLMs underperform compared to both smaller and extremely large models.
- **Fine-tuned small models** (mT5, ByT5) significantly outperform large LLMs such as GPT-4o-mini and LLaMA variants in both ICL and fine-tuning settings.
- **ByT5 excels at fine-grained character-level corrections** (e.g., transliteration, numeric misrecognition, compound splitting), while mT5 generalizes better across semantic error types and domain variations.
- We identify performance degradation in high out-of-domain settings and propose **mitigation strategies** to retain domain-specific fidelity in ASR post-correction.

2 Methodology

We define an in-domain dataset $D^{id}_{train} = \{(a_i, t_i), 1 \leq i \leq n\}$, consisting of n pairs of speech a_i and corresponding transcripts t_i, used to fine-tune LMs and LLMs. Unless explicitly stated otherwise, D^{id}_{train} is sourced from a single dataset rather than being aggregated from multiple sources. Let A^{id}_1 represent an ASR model trained on D^{id}_{train} and A^{ood}_2 denote a different ASR model trained on an out-of-domain dataset D^{ood}_{train}. The objective is to correct errors in the ASR hypotheses using fine-tuned LMs and LLMs. We generate the 1-best hypothesis for each instance in D^{id}_{train}, resulting in $H^{id}_{train} = \{(h_i, t_i), 1 \leq i \leq n\}$, where h_i is the ASR hypothesis and t_i is the reference transcript. The LMs and LLMs are fine-tuned on this data, with t_i serving as the target for correcting errors in h_i. During inference, for each audio sample in the test set D^{id}_{test}, the ASR model generates hypotheses that are subsequently processed by the fine-tuned models to produce accurate transcripts.

Developing a post ASR correction model requires a large dataset that contains pairs of error and gold transcriptions. However, creating such a dataset is particularly challenging for Hindi, as it is a low-resource language with limited availability of high-quality annotated data. To address this limitation, we construct a fine-tuning dataset comprising 1-best ASR hypotheses paired with their corresponding gold transcripts for each utterance (Li et al., 2024). This dataset will enable the generation of corrected transcriptions (H_{train}) from error hypotheses. Resulting, this approach aims to enhance the model's error correction performance on Hindi test data, improving overall transcription accuracy.

2.1 Training Data Creation

Figure 1 illustrates our proposed methodology for generating a suitable dataset to fine-tune LMs and LLMs for ASR error correction. We focus on creating training corpora in both in-domain and out-of-domain scenarios.

In-domain speech refers to data that shares similar speaker distributions, topics, vocabulary, and contextual characteristics with the target evaluation set, such as Lahaja. This alignment enables LMs and LLMs to more effectively learn domain-specific error patterns for ASR post-correction. In contrast, out-of-domain speech differs in style, source, or vocabulary, e.g., Kathbath (structured read speech) or Shrutilipi (news domain), and is less representative of the test conditions.

To address the scarcity of in-domain resources for low-resource languages like Hindi, we propose constructing domain-replicative training datasets, denoted as H^{id}_{train}. These can be used to fine-tune LMs/LLMs or for ICL. Similarly, for out-of-domain scenarios, we define H^{ood}_{train}, with utterances first transcribed by the ASR model. Note that D^{ood}_{train} is used for adapting the ASR model to these out-of-domain conditions.

3 Experiment and Results

Datasets: We evaluate LMs and LLMs performance on the Lahaja (Javed et al., 2024a) Hindi ASR dataset (12.5 hours, 132 speakers, 83 districts), which includes read, extempore, and conversational speech. For fine-tuning, we use the IndicVoice (Javed et al., 2024b) dataset (65 hours from 287 speakers), selected for its domain and vocabulary overlap with Lahaja. For out-of-domain evaluation, we include Kathbath (Javed et al., 2023) (read speech from IndicCorp) and Shrutilipi (Bhogale et al., 2023a) (conversational radio broadcasts). These datasets offer diverse speech styles and linguistic complexity.

Baseline: We use IndicWav2Vec (Javed et al., 2022) and IndicConformer (Javed et al., 2024a), two india specific multilingual ASR systems developed by AI4Bharat. These models follow wav2vec 2.0 and conformer architecture respectively. Our preliminary experiments showed these two models perform the best amongst other Hindi ASRs, detailed in Appendix C.

Model Configurations In this work, we evaluate our hypothesis on pre-trained LM ByT5, mT5 along with open-weight and close-weight LLMs, Llama-3 and ChatGPT-4o-Mini, respectively. ByT5 (Xue et al., 2022) is a T5 variant, an encoder-decoder based LM. It is a tokenizer-free, byte-level model. In contrast, mT5 (Xue, 2020) is a T5 (Raffel et al., 2020) variant that uses SentencePiece tokenization (Kudo, 2018). It is a multilingual model trained on common crawl data, including in Hindi. For an open-weight LLM, we use Llama-3-Nanda-10B-Chat, a 10B-parameter, bilingual English-Hindi LLM adapted from Llama-3-7B through architectural changes and continued pretraining on a 65B-token Hindi corpus. Lastly, for a closed-weight LLM, we adopt ChatGPT-4o-mini, which offers strong zero- and few-shot reasoning abilities at a favorable cost-performance balance.

3.1 Training and Evaluation

The training data for fine-tuned models comprise 1-best ASR hypotheses generated by various Hindi ASR systems, paired with their corresponding ground truth transcriptions. In contrast, ChatGPT-4o mini leverages few-shot learning with prompts designed using random and similar sentence embedding (Joshi et al., 2023) examples. These sentence embedding examples are constructed from Hindi ASR errors in the IndicVoice dataset, with corrections selected based on sentence embeddings to ensure semantic similarity and contextual relevance across the examples and the ASR hypotheses. Furthermore, we also trained the ByT5 on D1 dataset using an n-best hypothesis (n = 5) and observed a WER of 45, indicating the impact of multiple hypotheses in refining ASR post-correction.

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	28.60	**24.17**	32.92	76.72	18.02	18.22	**17.50**	76.04
D2: + In-Domain Speech with Diff. ASR model	127306	28.60	**26.62**	29.09	**27.8**	18.02	18.07	**16.75**	23.24
D3: + Out-of-Domain Speech with ASR model	1021472	28.60	**25.14**	**23.74**	**26.03**	18.02	**17.52**	**16.31**	21.49

Table 1: WER Comparison for Various fintuned LMs (ByT5-small, mT5-base) and LLM (Llama)

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]	63500	**26.00**	**26.67**	18.37	**16.81**
Shrutilipi [IC]	127306	31.37	29.67	24.18	22.19
Kathbath + Shrutilipi [IC]	127306	30.45	27.76	23.34	19.48
Shrutilipi [W2V]	127306	30.10	29.96	25.04	22.55
Kathbath + Shrutilipi [W2V]	127306	28.84	29.05	22.30	20.75
D2: IndicVoice [IC+W2V]	127306	**26.62**	29.09	18.07	**16.75**
D3: **D2** + other ASR dataset [IC]	1021472	**25.14**	**23.74**	**17.52**	**16.31**
D2 + other ASR dataset [W2V]	1021472	**23.66**	**22.97**	**17.55**	**16.45**
D2 + other ASR dataset [IC + W2V]	1021472	**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 Word Error Rate (WER) of the IndicWav2Vec (W2V) model is 28.6, while the IndicConformer (IC) model is 18.02 .

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 3: WER Comparison for Various Shot Settings using ChatGPT (ICL)

3.2 Results of Finetuned LMs and LLMs

Table 1 and Table 2, evaluate the performance of Finetuned LMs and LLMs under three training scenarios: in-domain speech with an ASR model (D1), in-domain speech using a different ASR model (D2), and out-of-domain speech with an ASR model (D3). **In-Domain Training:** When finetuned LMs are trained on datasets from the same domain (D^{id}_{train},H^{id}_{train}) as the test set (D^{id}_{test}), performance improves significantly. For example, *ByT5* and **mT5** performs better when training dataset is IndicVoice compared to Shrutilipi or other. We hypothesize that this is because the finetuned LMs encounter error patterns at test time that are similar to those it has been exposed to during training. **Out-of-Domain Training:** When finetuned LMs are trained on datasets from different domains (D^{ood}_{train},H^{ood}_{train}) than the test set (D^{id}_{test}), performance improvements are not observed. This is likely because the errors encountered at test time differ significantly from those seen in the training data, limiting the ability of the finetuned LMs to generalize effectively.

3.3 Results of ICL

In Table 3, we evaluate the ICL capability of a LLM, ChatGPT-4o mini. We assess ASR post-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 ASR hypothesis WER.

Our experiments evaluating the impact of model size under a zero-shot ICL setting, leveraging the inherent knowledge from the pre-trained versions of each model without further fine-tuning mentioned in the Appendix D.

4 Conclusion

In this work, we explored the effectiveness of language models (LMs) and large language models (LLMs) for post-ASR correction in Hindi, highlighting the surprising result that smaller, finetuned models such as mT5 and ByT5 consistently outperform much larger LLMs like GPT-4o-mini and LLaMA variants. Our findings reveal a U-shaped inverse scaling trend, where increasing model size initially degrades performance before marginal improvements at extreme scales, yet still falling 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 performance degradation in high out-of-domain settings and propose mitigation strategies to preserve domain-specific fidelity in ASR post-correction. These results underscore the importance of source-specific inductive biases and suggest that lightweight, fine-tuned models are better suited than general-purpose LLMs for improving ASR quality in low-resource language contexts.

Limitations

As part of future work, we would like to work on the following limitations of our work:

- While the study primarily focuses on Hindi, this language-specific scope may constrain the generalizability of the findings to other low-resource Indian languages with distinct linguistic characteristics. Although preliminary evaluations are conducted on Marathi and Telugu, they lack detailed analysis. Moreover, the absence of linguistic experts for these languages limits the depth of error categorization and interpretation.

- ICL results are limited to GPT-4o-mini and evaluated under only a few-shot and SE-based prompting. The comparison of 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.

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*.

T Kudo. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

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, 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, 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.

Ian R McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, et al. 2023. Inverse scaling: When bigger isn't better. *arXiv preprint arXiv:2306.09479*.

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

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*.

Jason Wei, Najoung Kim, Yi Tay, and Quoc V Le. 2022. Inverse scaling can become u-shaped. *arXiv preprint arXiv:2211.02011*.

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 Illustrative Example of Hindi ASR Errors

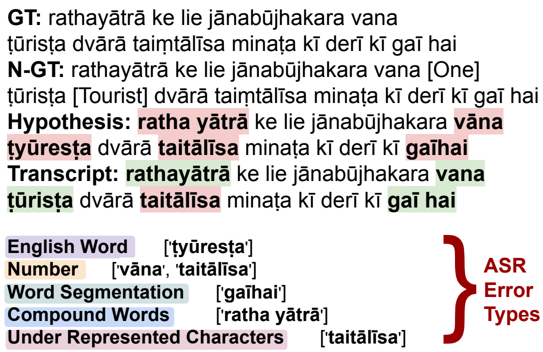

Figure 2: Example of ASR hypothesis errors in Hindi, categorized by error types: English word transliteration (*tyūresṭa*), number transcription (*vāna*, *taitālīsa*), word segmentation (*gaīhai*), compound word splitting (*ratha yātrā*), and underrepresented character errors (*taitālīsa*).

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 /)[2], 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 2.

B Related Works

LLMs have been integrated into ASR systems through various approaches. ASR error correction utilizes LLMs to rescore N-best lists of potential transcriptions, refining predictions (Ma et al., 2023; Radhakrishnan et al., 2023). Speech in-context learning 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 speech in-context learning (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 like Hindi. Similarly, LLM rescoring of N-best lists often underperforms compared to using a single 1-best hypothesis (Li et al., 2024), which is sufficient for addressing common errors such as word segmentation, underrepresented characters, and compound word handling.

C 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 4: Performance Comparison of Open-Source Hindi ASR Models on Hindi Lahaja dataset

Table 4 presents a comparative evaluation of open-source Hindi ASR models on the Hindi Lahaja dataset in terms of Word Error Rate (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 ASR post-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 finetuned Llama decline the ASR hypothesis quality.

D Ablation Study

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 Lahaja Dataset by mT5=16.17 model train on Lahaja 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 Lahaja Dataset by ByT5=17.46 model train on Lahaja 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 mistakes, numeric misrecognitions, and underrepresented graphemes, than mT5. This stems from

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

ByT5's byte-level tokenization, which provides finer granularity for detecting single-character perturbations. In contrast, mT5's sub-word vocabulary affords stronger semantic coverage but makes it less sensitive to very fine-grained character variations.

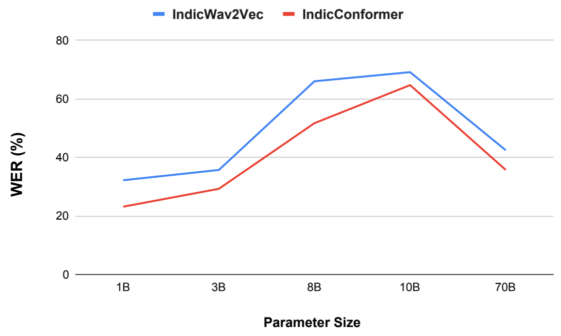

Figure 3: Inverse scaling phenomenon in Hindi post-ASR correction across varying LLaMA model sizes.

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

ByT5-small	ByT5-base	mT5-small	mT5-base	Llama	ChatGPT-4o mini
2.29	2.79	**0.97**	**1.84**	10.17	2.03

In Table 7, we summarize the latency of different LMs/LLMs, indicating that *mt5-small* performed the fastest post-ASR correction. It also points to the fact smaller models like mT5 not only achieve significant performance gains but also are faster than larger LLMs. Hence, we incorporate mT5 for post-ASR correction is advantageous for both performance wise and latency, enabling robust ASR correction in low-resource settings.

We evaluate the impact of model size on Hindi post-ASR correction under a zero-shot in-context learning setup, relying solely on the pre-trained knowledge of each model without additional fine-tuning. As shown in Figure 3, increasing parameter counts (3.1 1B $\rightarrow$ 3.2 3B $\rightarrow$ 3.1 8B $\rightarrow$ Nanda-10B-Chat 10B $\rightarrow$ 3.3 70B) reveals an n-shaped trend in 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.

E 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.

F Additional Languages

Language	Hypothesis	ByT5 small	ByT5 base	mT5 small	mT5 base
Marathi	25.556	26.324	26.018	25.761	25.122
Telugu	23.284	24.51	24.725	22.68	22.05

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

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.

G 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 ASR post-correction and helps evaluate model performance on preserving lexical integrity.

H 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[3]

Models

mT5: mT5-small (300M parameters), mT5-base (580M parameters)

ByT5: ByT5-small (300M parameters), ByT5-base (580M parameters)

Nanda: Llama3-10B

ChatGPT: 8B parameter

I 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 ASR post-correction? As shown in Table 9, fine-tuning ByT5 and mT5 variants with a controlled in-domain to out-of-domain ratio 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 out-of-domain proportions. 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 ASR post-correction.

J Prompt

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: $\text{GT}_{\text{tokens}} \leftarrow \text{split(GT)}$
Tokenize Hyp: $\text{Hyp}_{\text{tokens}} \leftarrow \text{split(Hyp)}$
Step 3: Generate Substrings
for each word $\in \text{GT}_{\text{tokens}}$ **do**
  Splits $\leftarrow$ splits(word)
  Store valid splits as $\text{Splits}_{\text{valid}}$
end for
Step 4: Validate Substrings
for each split $\in \text{Splits}_{\text{valid}}$ **do**
  if all substrings subsplit $\in$ split exist in T **then**
    Add split to $\text{CompoundWords}_{\text{valid}}$
  end if
end for
Step 5: Check for Errors
for each word $\in \text{CompoundWords}_{\text{valid}}$ **do**
  if word $\notin \text{Hyp}_{\text{tokens}}$ **then**
    Add word to Er_{CW}
  end if
end for
Step 6: Output Results
Save Er_{CW} for further analysis

---

[3] 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]	3:7	381415	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]	2:8	571962	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]	1:9	993155	0.2333	0.1829	0.2234	0.1762	0.2388	0.1712	0.2549	0.1638

Table 9: Evaluation of ASR post-correction on Lahaja dataset mixing the in-domain and out-of-domain dataset in fixed ratio

ChatGPT Prompt based on error-types

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 own 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 gaī hai

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