KrishiVaani: A Conversational Hindi Speech Corpus through Automatic ASR Post-Correction and Accelerated Refinement

Anonymous ACL submission

Abstract

Building robust, real-world Automatic Speech Recognition (ASR) datasets for Hindi remains challenging due to linguistic complexity, accent variations, and domain-specific vocabulary, particularly in agricultural contexts. This paper presents **KrishiVaani**, a conversational Hindi speech corpus that addresses the issues mentioned above. Additionally, we provide a comparative analysis of various LMs and LLMs for automatic ASR post-correction, aiding in model selection to minimize annotation effort. We also extend an open-source tool, VAgyojaka, to accelerate data validation and verification processes by 6x and 2x, respectively. This enhancement streamlines the creation of large-scale Hindi speech corpora, ensuring high-quality data through efficient annotation and error detection. Experimental results show that using KrishiVaani significantly improves accuracy across diverse speaker accents, environmental noise levels, and agricultural terminology.

1 Introduction

Advances in speech technology have transformed human-AI interaction, driving the need for diverse voice user interfaces (VUIs) (Deshmukh and Chalmeta, 2024). These VUIs depend on ASR systems to enable smooth communication in low-resource languages. This is particularly significant in India, where many languages pose challenges in written form (Ghosh et al., 2010). However, linguistic diversity, accent variation, code-mixing, and environmental noise remain major challenges. Despite progress in ASR, development for low-resource Indian languages like Hindi is hindered by the lack of high-quality, domain-specific, real-world speech data (Adiga et al., 2021; Javed et al., 2024b).

To address this, we introduce the KrishiVaani Hindi speech corpus, which aims to address the existing gaps and foster research in Hindi ASR. KrishiVaani stands out as a real-world agricultural domain corpus, capturing speech in noisy environments, making it a valuable asset. The corpus also encompasses diverse vocabulary and conversations from both urban and rural speakers, enhancing its linguistic and demographic diversity.

Developing high-quality ASR systems for Indian languages requires diverse representation and real-world speech datasets, yet existing resources present notable limitations. The Shrutilipi dataset (Bhogale et al., 2023a), despite offering an extensive 2.35K hours of Hindi speech sourced from All India Radio (AIR) archives, suffers from OCR errors, misaligned text, and non-transcribed content, making it challenging for precise audio-text alignment. Similarly, Kathbath (Javed et al., 2023) includes test sets (*Kathbath Test-Unknown* and *Kathbath Test-Known*) but primarily consists of read speech from the IndicCorp (Kakwani et al., 2020) monolingual corpora, failing to capture conversational and code-switched speech. The Spring-inx dataset (Gangwar et al., 2023), a comparatively large resource of Hindi speech, ensures gender and dialectal diversity, but has limited representation of code-mixed and code-switched speech (only 10%). IndicVoice (Javed et al., 2024b) covers read, extempore, and conversational speech, but features very short conversational segments (2-3 minutes) to be considered suitable for dialog-centric ASR systems. These limitations highlight the need for better-curated, large-scale datasets that include spontaneous speech, diverse accents, longer conversational structures, and robust code-switching representation. To address these challenges and ensure high-quality transcriptions, we incorporate VAgyojaka (Kumar et al., 2022a) an open-source ASR post-correction tool, which we have extended to support data validation and data verification.

ASR systems often use language models (LMs) (Kumar et al., 2022b; Wang et al., 2022) and/or large language models (LLMs) (Kumar et al., 2024;

Ma et al., 2023; Li et al., 2024b) for automatic post-correction. The initial hypotheses generated by ASR systems can be post-corrected using these LMs/LLMs. This minimizes the number of errors to be corrected by human annotators beforehand. We leverage this paradigm using different LMs/LLMs through two approaches: fine-tuning (Li et al., 2024a; Hu et al., 2024), which requires a large training dataset, and in-context learning (Ghosh et al., 2024), which uses a small set of in-domain examples for correction. By using LMs/LLMs as automatic ASR post-correctors, we reduce the annotation effort. Our approach is tailored to Hindi, focusing on lexical and multiword interventions involving both lexical and morphemic-level knowledge, helping in post-correction of ASR results. This entire corpus creation setup can be seamlessly extended to other languages as well.

Our contribution can be summarized as follows:

- We develop a speech corpus curation pipeline leveraging the VAgyojaka tool, significantly accelerating data validation and verification by factors of 6x and 2x, respectively.
- We showcase this pipeline to curate **KrishiVaani**, a conversational Hindi speech corpus for real-world agricultural scenarios.
- We present a comparative analysis of various smaller LMs (mT5, ByT5) and LLMs (Llama, ChatGPT) for automatic post-correction in ASR, minimizing the annotation effort.
- We have released the code, our proposed models, KrishiVaani dataset[1], modified VAgyojaka tool[2], and associated annotation guidelines to facilitate further research.

2 Pipeline for Corpus Construction

We construct the KrishiVaani speech corpus through a multi-stage pipeline comprising three key stages: (i) Data Collection, (ii) Corpus Construction, and (iii) Data Refinement. The complete pipeline is shown in Figure 1, while the detailed step-by-step description is provided in Appendix A.

2.1 Data Collection

We begin by creating a domain-specific keyword list using Hindi and English dictionaries and Wikipedia articles. These keywords are used to query and download Hindi YouTube videos, which are then converted into mono-channel audio files with a uniform sampling rate (16 kHz). Since YouTube metadata often includes English tags, incorporating both languages ensures better domain coverage.

2.2 Speech Corpus Construction

The audio files are segmented and processed through a series of steps:

Voice Activity Detection (VAD) is applied to trim silence and identify speech-active segments.

Speaker Diarization and segmentation are performed using pre-trained models to separate and label speech by speaker.

ASR Transcription is conducted using multiple Hindi ASR models. We prioritize character-level accuracy to reduce annotation load.

Forced Alignment links each word in the transcript to its temporal location in the audio, which is crucial in correcting transcripts via the VAgyojaka Tool.

The aligned, speaker-wise utterances form our initial *audio-text corpus*. We further improve transcript quality by automatic ASR post-correction using fine-tuned LMs (e.g., ByT5, mT5).

2.3 Data Refinement

We use the VAgyojaka tool to streamline three post-processing steps:

Data Curation: Annotators correct ASR outputs, segment utterances, and ensure diarization quality following our internal guidelines.

Transcript Post-Processing: Text normalization and English-to-Hindi transliteration are carried out with model-assisted verification.

Validation and Verification: A two-layer quality check ensures correctness and consistency. A reward-based scheme incentivizes high-quality annotations.

We observed that the use of LMs, LLMs, and the VAgyojaka tool together reduced the annotation time by nearly half, making this process scalable for other Indian languages.

3 Experimentation

In this section, we describe the KrishiVaani dataset and KVWav2Vec, our proposed model, followed by the various baselines used in our experiments. Additionally, we outline the LM/LLM configurations

[1] https://anonymous.4open.science/r/KrishiVaani-2463
[2] https://github.com/vagyojaka/VAgyojaka

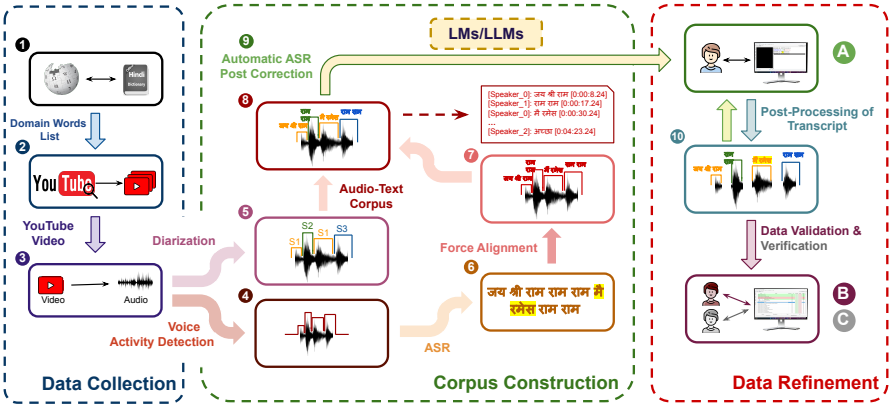

Figure 1: Overview of the multi-stage pipeline used to develop the **KrishiVaani** Hindi speech corpus. The process begins with domain-specific Data Collection (left), Speech Corpus Construction (center), and Data Refinement (right) to ensure the generation of a high-quality, domain-specific Hindi speech dataset.

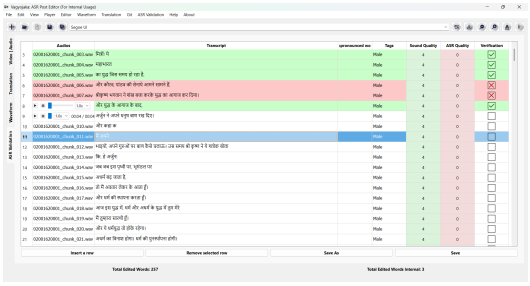

Figure 2: Performing data validation and data verification through the open-source VAgyojaka tool

applied to our fine-tuned and in-context learning models used for automatic ASR post-correction.

3.1 KrishiVaani Dataset

The KrishiVaani corpus comprises 20 hours of transcribed audio from YouTube videos, primarily focused on the agriculture domain, while also encompassing various other fields. This corpus is divided into several subsets, including 10 hours for training and 2 hours for validation data. Apart from that, there are also test sets having 1 hour for the KrishiVaani-Known (which shares common speakers with the training set), 3 hours for the KrishiVaani-UnKnown (including speakers absent from training data), and 4 hours for the out-of-domain (OOD) KrishiVaani-OOD (including speech from multiple domains excluding agriculture). Notably, the *KrishiVaani-OOD* dataset encompasses diverse domains, ensuring broader applicability and enhancing the dataset's utility for domain-agnostic ASR research. To assess the quality of the KrishiVaani corpus, we trained **KVWav2Vec**, an IndicWav2Vec (Javed et al., 2023) model for the Hindi language. The training dataset combined 65 hours from IndicVoice (Javed et al., 2024b) with 10 hours from the KrishiVaani training split data.

3.2 Baselines

KVWav2Vec is compared against the following ASR baselines:

1. IndicWav2Vec (Javed et al., 2023): Multilingual wav2vec 2.0 model fine-tuned on Hindi dataset with CTC loss.

2. IndicWhisper (Bhogale et al., 2023b): Whisper-based ASR model fine-tuned on Hindi dataset with byte-level BPE.

3. SALSA (Mittal et al., 2024): Hybrid model combining Whisper encoder with Llama2-7B(Touvron et al., 2023) decoder for low-resource Hindi ASR.

4. IndicConformer (Javed et al., 2024a): The Conformer-based model fine-tuned on Hindi using subword-character tokenization.

5. SeamlessM4T (Barrault et al., 2023): A large-scale multilingual ASR and translation model trained on 4.5M hours of speech.

3.3 LM/LLM Configurations

This section highlights how we apply both fine-tuning and in-context learning approaches for ASR post-correction using LMs and LLMs.

1. mT5-small (Xue, 2020): Multilingual encoder-decoder with subword tokenization.

2. ByT5-small (Xue et al., 2022): Tokenizer-free variant of mT5 using UTF-8 bytes.

3. Llama-3-Nanda-10B (Touvron et al., 2023): Hindi-focused LLM trained on 65B Hindi tokens.

4. ChatGPT-4o mini (Ma et al., 2023): Used

Model	IndicWav2Vec	IndicWhisper	IndicConformer	SeamlessM4T	SALSA	KVWav2Vec
KrishiVaani-Known (WER)	23.7	23.42	24.2	49.94	87.59	**22.38**
KrishiVaani-Known (CER)	**8.50**	10.65	10.3	31.23	67.35	8.58
KrishiVaani-UnKnown (WER)	28.57	30.08	26.84	41.5	95.50	**26.04**
KrishiVaani-UnKnown (CER)	12.69	16.62	12.92	25	75.20	**11.98**
KrishiVaani-OOD (WER)	**22.41**	49.78	28.56	42.73	79.09	24.61
KrishiVaani-OOD (CER)	**8.51**	35.96	17.34	27.73	61.24	9.51

Table 1: Models Comparison across different KrishiVaani datasets

Model	KVWav2Vec	ByT5	mT5	Llama
KrishiVaani-Known	22.38	24.33	**22.29**	26.37
KrishiVaani-UnKnown	26.04	26.77	**25.70**	37.75
KrishiVaani-OOD	24.61	25.12	**24.35**	30.08

Table 2: Comparison of different WER (%) scores of KVWav2Vec with other fine-tuned LMs/LLMs.

for zero-, one-, and few-shot in-context correction; SBERT embeddings (Joshi et al., 2023) guide selection of examples.

4 Results and Discussions

Experiment	Shots	KrishiVaani-Known	KrishiVaani-UnKnown	KrishiVaani-OOD
KVWav2Vec	-	22.38	26.04	24.61
ChatGPT-4o mini	0-Shot	29.33	31.89	27.65
With Random	1-Shot	28.36	30.26	26.43
	3-Shot	27.64	30.14	25.65
	5-Shot	26.37	28.95	25.27
With SE Similarity	1-Shot	27.64	30.26	26.16
	3-Shot	26.81	29.90	25.74
	5-Shot	23.59	26.35	**22.62**

Table 3: Comparative overview of different WER (%) scores for different settings using in-context learning through ChatGPT

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

Table 4: Latency (in seconds) of different models for ASR post-correction.

In this section, we analyze and compare the performance of KVWav2Vec with various other ASR baselines. We perform Beam Search Decoding on the KrishiVaani test sets and calculate the WER and CER. As established earlier, for Indian languages, CER serves as a much better metric to quantify the errors in ASR systems. Referring to Table 1, we observe that our KVWav2Vec model outperforms all existing baselines on the KrishiVaani-UnKnown dataset and performs on par with them for the KrishiVaani-Known dataset. Among the baseline models, IndicWav2Vec achieved the best performance on the KrishiVaani-OOD dataset, with KVWav2Vec trailing by a narrow margin. This shows that KVWav2Vec excels in conversational ASR when the speaker or domain is familiar.

Table 2 presents a comparative analysis of our KVWav2Vec hypotheses (before automatic ASR correction) and our fine-tuned LMs/LLMs after correction. Smaller models (ByT5, mT5) perform better in correction than the larger Llama model. Table 3 shows the performance of in-context learning for ASR systems using ChatGPT-4o mini as a correction model. Without context (0-shot), it fails to correct ASR errors. With random context, it improves using linguistic knowledge, and with sentence embeddings, it even restores missing tokens. The 5-shot settings with sentence embeddings-based similarity showed improvement on the KrishiVaani-OOD dataset as ChatGPT's knowledge extends in several other domains. However, as compared to KVWav2Vec, it showed no improvement on the other two test sets. This is likely due to limited in-context examples for Indian languages in ChatGPT-4o mini's training for that domain. As seen in Table 4, we summarize the latency of different LMs/LLMs, indicating that *mt5-small* performed the fastest automatic ASR post-correction. It also points to the fact that smaller models like mT5 not only achieve significant performance gains but also are faster than larger LLMs. Hence, we incorporate mT5 for automatic ASR correction in our pipeline.

5 Conclusion

We have thus demonstrated the effectiveness of our custom pipeline in curating the KrishiVaani Hindi speech corpus. KrishiVaani addresses key challenges in Hindi ASR by providing a high-quality conversational speech dataset for real-world applications, particularly agriculture. Our proposed model KVWav2Vec also shows impressive performance for known and unknown scenarios. Our comparative analysis of various LMs/LLMs for ASR post-correction offers valuable insights into model selection for reducing annotation effort. Furthermore, leveraging the VAgyojaka tool significantly accelerates the data refinement stage, facilitating efficient large-scale corpus creation.

Limitations

The pipeline leverages models like IndicWav2Vec, ByT5, and Llama-3-Nanda, which are either trained on large corpora or Hindi-specific datasets. For languages that lack such pretrained models, this pipeline may not deliver comparable results unless similar resources are first developed, limiting its plug-and-play potential across languages. Moreover, the pipeline is not effective in handling a code-switched speech dataset. The pipeline can be modified to handle a code-switched dataset.

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*.

Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. 2020. Pyannote. audio: neural building blocks for speaker diarization. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7124–7128. IEEE.

Alexis Conneau, Min Ma, Simran Khanuja, Yu Zhang, Vera Axelrod, Siddharth Dalmia, Jason Riesa, Clara Rivera, and Ankur Bapna. 2023. Fleurs: Few-shot learning evaluation of universal representations of speech. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 798–805. IEEE.

Akshay Madhav Deshmukh and Ricardo Chalmeta. 2024. User experience and usability of voice user interfaces: A systematic literature review. *Information*, 15(9):579.

Arjun Gangwar, S Umesh, Rithik Sarab, Akhilesh Kumar Dubey, Govind Divakaran, Suryakanth V Gangashetty, et al. 2023. Spring-inx: A multilingual indian language speech corpus by spring lab, iit madras. *arXiv preprint arXiv:2310.14654*.

Debashis Ghosh, Tulika Dube, and Adamane Shivaprasad. 2010. Script recognition—a review. *IEEE Transactions on pattern analysis and machine intelligence*, 32(12):2142–2161.

Sreyan Ghosh, Mohammad Sadegh Rasooli, Michael Levit, Peidong Wang, Jian Xue, Dinesh Manocha, and Jinyu Li. 2024. Failing forward: Improving generative error correction for asr with synthetic data and retrieval augmentation. *arXiv preprint arXiv:2410.13198*.

Yuchen Hu, Chen Chen, Chao-Han Huck Yang, Ruizhe Li, Chao Zhang, Pin-Yu Chen, and EnSiong Chng. 2024. Large language models are efficient learners of noise-robust speech recognition. *arXiv preprint arXiv:2401.10446*.

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, 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.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, NC Gokul, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar. 2020. Indicnlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961.

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, Mayank Kothyari, Jatin Dalal, Ganesh Ramakrishnan, and Preethi Jyothi. 2022a. Vagyojaka: An annotating and post-editing tool for automatic speech recognition. In *INTERSPEECH*, pages 857–858.

Rishabh Kumar, Devaraja Adiga, Rishav Ranjan, Amrith Krishna, Ganesh Ramakrishnan, Pawan Goyal, and Preethi Jyothi. 2022b. 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. 2024a. Investigating asr error correction with large language model and multilingual 1-best hypotheses. In *Proc. Interspeech*, pages 1315–1319.

Yuang Li, Jiawei Yu, Min Zhang, Mengxin Ren, Yanqing Zhao, Xiaofeng Zhao, Shimin Tao, Jinsong Su, and Hao Yang. 2024b. Using large language model for end-to-end chinese asr and ner. *arXiv preprint arXiv:2401.11382*.

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*.

Yash Madhani, Sushane Parthan, Priyanka Bedekar, Gokul Nc, Ruchi Khapra, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra. 2023. Aksharantar: Open indic-language transliteration datasets and models for the next billion users. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 40–57.

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

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*.

Ehsan Variani, Tom Bagby, Kamel Lahouel, Erik McDermott, and Michiel Bacchiani. 2018. Sampled connectionist temporal classification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4959–4963. IEEE.

Chengyu Wang, Suyang Dai, Yipeng Wang, Fei Yang, Minghui Qiu, Kehan Chen, Wei Zhou, and Jun Huang. 2022. Arobert: An asr robust pre-trained language model for spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1207–1218.

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.

A Detailed Pipeline for KrishiVaani

This section describes the generation of the KrishiVaani speech corpus through our custom pipeline. Figure 1 highlights the design of this multi-stage pipeline involving three stages of data collection, speech corpus construction, and data refinement, respectively. The following subsections give a detailed overview of each stage:

A.1 Data Collection

This stage gathers the required data to be processed further. It is sequentially broken down into the following steps:

Domain Words List (1): We generate different keyword lists (e.g., "khetī" means farming) containing Hindi and English terms from different dictionaries and Wikipedia articles on these domains. We refine it using a keyword filtering module relevant to YouTube Hindi videos[3]. YouTube Hindi videos have English keywords associated with them. Therefore, it is essential to have both Hindi and English keyword lists.

YouTube Search Engine (2): We use the YouTube search engine to find the corresponding videos on these domains from the respective keyword lists. We de-duplicate the retrieved videos. Our analysis of these videos revealed that many of these videos contained missing or inaccurate subtitles. Therefore, we generated the transcripts for these videos.

Converting Video to Audio (3): We download the video and convert it into audio using single channel *wav* format and 16 kHz sampling rate.

[3]We follow YCC licensing.

A.2 Speech Corpus Construction

Once the data collection stage is complete, the speech corpus construction process begins, involving the following steps:

Voice Activity Detection (4): We conduct voice activity detection (VAD) with a limited maximum duration of 20 sec and average silence detection using mp3split[4]. We also use speaker segmentation from Pyannote to separate the speaker's audio and ensure the integrity of speech content as much as possible.

Diarization (5): For speaker diarization, we segment and label an audio recording using Pyannote (Bredin et al., 2020). The goal of this step is to determine *"who spoke when"* within a multi-speaker audio stream. This enables cleaner and more reliable training data for ASR models.

ASR (6): In this step, we use Hindi ASR to generate the transcripts of the converted audio files. We considered multiple ASR models such as IndicWhisper (Bhogale et al., 2023b), IndicWav2Vec (Javed et al., 2023), etc., to maintain flexibility. Though IndicWhisper showcases a lower Word Error Rate (WER), we prefer IndicWav2Vec owing to its lower Character Error Rate (CER). Since IndicWav2Vec captures fine-grained details at the character level, it also helps to reduce further annotation effort.

Forced Alignment (7): This step maps each word in the utterance to its corresponding audio timestamp. We use a pre-trained IndicWav2Vec-Hindi model based on the Connectionist Temporal Classification (CTC) criterion (Variani et al., 2018). This step also benefits the upcoming data curation stage.

Audio-Text Corpus (8): After performing diarization and forced alignment, we combine the timestamps of the speaker(s) and the transcripts to create the final speaker-wise utterances.

Automatic ASR Correction (9): We use the best one of different fine-tuned LMs (ByT5 (Xue et al., 2022), mT5 (Xue, 2020)) and LLMs (Llama (Touvron et al., 2023), ChatGPT) to perform ASR post-correction. This step enhances both transcript accuracy and annotation efficiency. The output of this step is provided to the data refinement stage.

A.3 Data Refinement

For the entire stage of data refinement, we use the VAgyojaka tool (Kumar et al., 2022a) for each of the data curation, data validation, and data verification steps. Before VAgyojaka, ASR curation was a time-intensive process, where X hours of speech data required 8X hours for curation. This time was spent correcting ASR transcripts, validating forced alignments, and checking speaker diarization. It has proven to reduce this workload to 4-6X hours. Figure 2 displays a snapshot of the VAgyojaka tool.

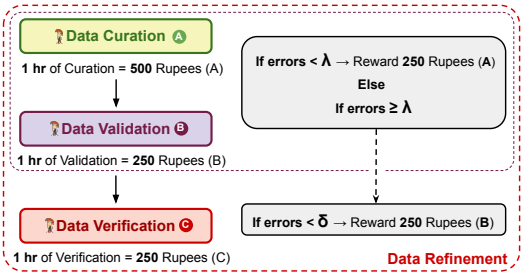

Figure 3: The reward system adopted to encourage data curators and validators. Alongside standard compensation, we used λ as 5% and δ as 2% of the total number of words for enforcing the reward system while curating the KrishiVaani dataset.

Data Curation (A): In this step, we collectively perform transcription correction, utterance alignment, and speaker diarization. Due to the absence of open-source guidelines for data curation in Indian languages, we create and use our own set of guidelines for this purpose. We have used fifteen annotators (experts in the Hindi language) for data curation.

Post-Processing of Transcript (10): This step involves normalizing the text by removing punctuation and tags. We also back-transliterate English words like "*Local*" as "*lokl*" using the transliteration model (IndicTXlit (Madhani et al., 2023)), which linguists further verify. We also check if any edits are made in the transcripts, depending on which, they can be sent back to the data curator. Eventually, we split the audio based on utterance alignment, which will be used for further validation and verification.

Data Validation (B): Data validation is a crucial step in ensuring the quality of the curated data. The same fifteen annotators carried out the validation, ensuring that no one validated their own curated data. The data validator corrects the transcript of the split audio. We also introduce a reward system so that annotators can work better as data curators and validators. As seen in Figure **??**, whenever any data validator makes edits less than a certain threshold (λ), the data curator will be rewarded, or else, the data validator will perform the subsequent

[4]http://mp3splt.sourceforge.net/mp3splt$_page/home.php$

post-correction. If this step achieves less than a certain threshold (δ), the data validator will be rewarded.

Data Verification (C): The Data verifier examines the ASR transcript for the split audio and determines whether to include it in the final dataset. These verifiers are Hindi language experts with extensive experience. The verifier also helps in complying with the reward system.

B Baseline Models Details

1. IndicWav2Vec (Javed et al., 2023), a multilingual ASR model based on wav2vec 2.0, pretrained on 17,314 hours of Indian language audio using self-supervised learning and fine-tuned with CTC, achieving reduced WER for low-resource languages like Hindi.

2. IndicWhisper (Bhogale et al., 2023b), a fine-tuned version of OpenAI's Whisper trained on 2,150 hours of Hindi audio using multilingual byte-level BPE tokenization, balancing weak supervision with competitive WER.

3. SALSA (Mittal et al., 2024), a hybrid ASR approach integrating Whisper's encoder-decoder with a decoder-only LLM (Llama2-7B) (Touvron et al., 2023) to improve recognition in low-resource settings through synchronous decoding. It is fine-tuned on 10 hrs of FLEURS (Conneau et al., 2023) Hindi dataset.

4. IndicConformer (Javed et al., 2024a), a 130M parameter conformer-based ASR Hindi model combining convolution and transformer layers to support Indian languages with a unified subword-character tokenization strategy.

5. SeamlessM4T (Barrault et al., 2023), a multilingual ASR and translation model covering 100 languages, leveraging 4.5 million hours of pretraining but with unexplored effectiveness in real-world low-resource speech settings.

C LM/LLM Models Details

1. mT5: mT5 (Xue, 2020) belongs to the family T5 models, which is an encoder-decoder based LM. It is a multilingual version of the T5 (Raffel et al., 2020) model. The mT5 model uses a subword-based tokenization strategy. It follows the SentencePiece tokenizer (Kudo, 2018), which is a variant of the BPE tokenizer. We selected the *mT5-small* variant as the correction model.

2. ByT5: ByT5 (Xue et al., 2022) model shares the same architecture as the mT5 model. It is a tokenizer-free variant of the mT5 model. Although mT5 has a standard tokenizer, ByT5 processes single-character tokens in UTF-8 encoded bytes. We select the *ByT5-small* variant as the correction model.

3. Llama: Llama-3-Nanda-10B-Chat (Touvron et al., 2023) (Nanda) is a Hindi-focused, instruction-tuned LLM with 10 billion parameters. Built on the Llama-3 model, it has been extensively trained on 65 billion Hindi tokens. Unlike general multilingual models, Nanda prioritizes Hindi, using a balanced 1:1 Hindi-English dataset during training to enhance both languages' capabilities and make it well-suited for fine-tuned text correction tasks.

4. ChatGPT: Powered by OpenAI, ChatGPT (Ma et al., 2023) is an advanced LLM that is used for multiple downstream tasks, including text correction. Unlike fine-tuning for the above LMs/LLMs, we use ChatGPT-4o mini for in-context learning and transcript correction using zero-shot, 1-shot, and few-shot learning. We use SBERT (Joshi et al., 2023) to create sentence embedding for in-context learning.

D KVWav2Vec Model

We acknowledge the potential bias in evaluating KVWav2Vec solely on the KrishiVaani dataset. The primary goal of our study is to investigate whether fine-tuning the existing IndicWav2Vec model with a relatively small domain-specific dataset (KrishiVaani) could effectively support rapid dataset curation. To demonstrate this, we selected the best-performing baseline model (IndicWav2Vec) and fine-tuned it with KrishiVaani data, thereby creating distinct test scenarios - Known, Unknown, and Out-Of-Domain (OOD). The overarching aim is to minimize human annotation effort and accelerate data creation.

KVWav2Vec primarily serves to evaluate whether a domain-specific fine-tuned model can expedite and enhance the data curation process effectively, particularly in conversational and unknown-speaker scenarios. While not claiming state-of-the-art performance, we demonstrate its utility in reducing human annotation effort significantly as shown in Table 1

We experimented on the IndicWav2Vec model fine-tuned with IndicVoice (IV) + KrishiVaani (KV) vs IndicVoice alone, which showed modest but con-

Model	KVWav2Vec	Wav2Vec-IV
KrishiVaani-Known	22.38	23.4
KrishiVaani-UnKnown	26.04	27.69
KrishiVaani-OOD	24.61	22.38

Table 5: Comparison of different WER and CER (%) scores of KVWav2Vec with IndicVoice (IV) + KrishiVaani (KV) and IndicVoice (IV) alone

sistent improvements in both WER and CER across Known and Unknown test sets.

E LLM-based Post-Correction

We explore both small LMs (ByT5, mT5) and larger LLMs (Llama, ChatGPT) for automatic ASR post-correction. Our experiments indicate that smaller models (e.g., mT5) performed better than larger LLMs like ChatGPT, guiding our choice to integrate mT5 into our automatic correction pipeline. Moreover, the central objective of our work is to minimize human annotation effort and accelerate high-quality data creation for conversational ASR in Hindi.

F Data Annotators

The annotation guideline is mentioned in the KrishiVaani Github[5].

We have developed our in-house team and volunteers who help us in data curation, data validation, and data verification tasks for the KrishiVaani Dataset. We have trained them for the task and made them familiar with the VAgyojaka tool, then we ask them to do the following tasks. We have paid 5$ per hour for the data curation task and 2.5$ for both data validation and data verification tasks. All the annotators were from India.

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]

VAgyojaka Tool details: We developed the data validation and data verification task in VAgyojaka tool using the QT 6 framework[7].

H Prompts

[5]https://anonymous.4open.science/r/KrishiVaani-2463/

[6]https://pytorch.org/

[7]https://www.qt.io/product/framework

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 ṭyūreṣṭa dvārā taitālīsa minaṭa kī derī kī gaīhai

Predicted Output: