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

Anonymous submission to Interspeech 2025

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 above-mentioned issues. 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.

Index Terms: low resource language, speech recognition, corpus

1. Introduction

The advancement of speech technology has transformed human interaction with Artificial Intelligence (AI) systems. This has created the need for various voice user interfaces (VUIs) [1]. These VUIs rely on ASR systems, which play a crucial role in facilitating seamless communication in low-resource languages. This is particularly significant in countries like India, where many languages pose challenges in written form [2]. An interesting example is the development of Agriculture Bot, a conversational AI powered by foundational models designed to assist farmers by providing agricultural solutions in their native languages. Such ASR advancements enhance accessibility but face challenges like linguistic diversity, accents, code-mixed speech, and noise. Developing scalable solutions and robust, domain-specific models is essential for the deployment of AI systems in real-world applications. Despite the advancements in ASR systems, progress in low-resource Indian languages remains constrained due to the scarcity of high-quality, domain-specific, real-world labeled speech data [3]. This limitation presents a significant challenge for developing robust ASR systems in widely spoken but underrepresented languages like Hindi [4]. 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 for building robust domain-specific Hindi ASR systems. Furthermore, the corpus encompasses diverse vocabulary, recording channels, 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 [5], 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 [6], with 142 hours of training data, includes test sets (*Kathbath Test-Unknown* and *Kathbath Test-Known*) but primarily consists of read speech from the IndicCorp [7] monolingual corpora, failing to capture conversational and code-switched speech. The Spring-inx dataset [8], a comparatively large resource with 351 hours of Hindi speech, ensures gender and dialectal diversity, but its limited representation of code-mixed and code-switched speech (only 10%) restricts its ability to model such linguistic phenomena. IndicVoice [4], while covering 65 hours of read, extempore, and conversational speech from 287 speakers, features conversational segments that are too short (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 build scalable and adaptable ASR models for Indian languages. To address these challenges and ensure high-quality transcriptions, we incorporate VAgyojaka [9] an open-source ASR post-correction tool, which we have extended to support data validation and data verification.

ASR systems are often used alongside language models (LMs) [10, 11] and large language models (LLMs) [12, 13, 14] 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 [15, 16], which requires a large training dataset, and in-context learning [17], which uses a small set of in-domain examples for correction with minimal manual effort. By employing the usage of LMs/LLMs as automatic ASR post-correctors, we reduce the annotation effort. Our contribution can 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, and subsequently introduce **KVWav2Vec**, our ASR model to assess its quality and effectiveness.
- We present a comparative analysis of various LMs and LLMs

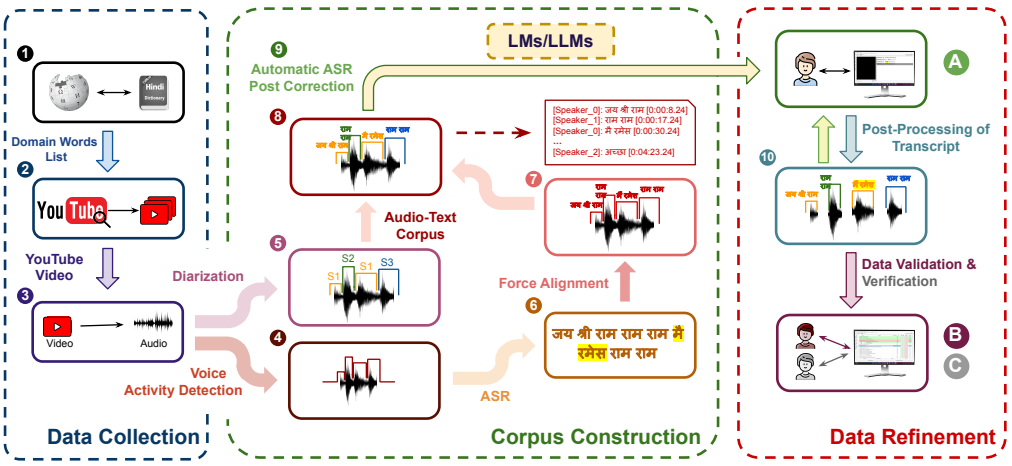

Figure 1: *Overview of the multi-stage pipeline used to develop the **KrishiVaani** Hindi speech corpus. The process begins with domain-specific Data Collection (left), Speech Corpus Construction (center), and Data Refinement (right) to ensure the generation of a high-quality, domain-specific Hindi speech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

for automatic post-correction in ASR, minimizing annotation effort and enhancing data curation.

- We also release the code, our proposed KVWav2Vec model, KrishiVaani dataset, and associated annotation guidelines to facilitate further research.

2. Pipeline for Corpus Construction

This section describes the generation of 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:

2.1. 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 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. 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.

2.2. 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[1]. 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 [18]. 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 [19], IndicWav2Vec [6], 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 the 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 [20]. 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 [21], mT5 [22]) and LLMs (Llama [23], 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.

2.3. Data Refinement

For the entire stage of data refinement, we use the VAgyojaka tool [9] 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.

[1] http://mp3splt.sourceforge.net/mp3splt$_{page/home.php}$

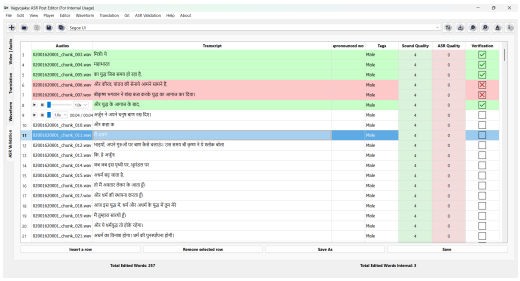

Figure 2: *Performing data validation and data verification through the open-source VAgyojaka tool*

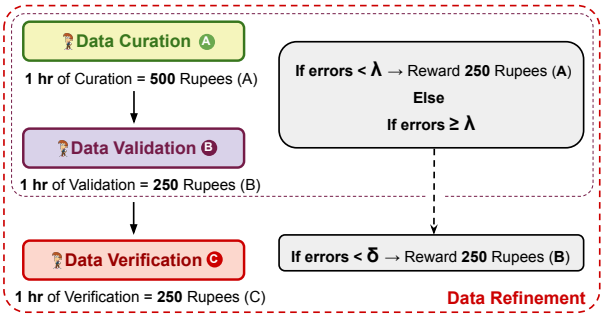

Figure 3: *The reward system adopted to encourage data curators and validators. Alongside standard compensation, we used λ as 5% and δ as 2% of the total number of words for enforcing the reward system while curating the KrishiVaani dataset.*

Figure 2 displays a snapshot of the VAgyojaka tool.

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 "*लोकल*" using the transliteration model (IndicTXlit [24]), 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 annotators can work better as data curators and validators. As seen in Figure 3, whenever any data validator makes edits lesser than a certain threshold (λ), the data curator will be rewarded, or else, the data validator will perform the subsequent 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.

3. Experimentation

In this section, we describe KrishiVaani dataset and KVWav2Vec, our proposed model, followed by the various baselines used in our experiments. Additionally, we outline the LM/LLM configurations 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 [6] model for the Hindi language. The training dataset combined 65 hours from IndicVoice [4] with 10 hours from the KrishiVaani training split data.

3.2. Baselines

We compare the performance of our proposed KVWav2Vec model with the following ASR baselines:

IndicWav2Vec [6], a multilingual ASR model based on wav2vec 2.0, pre-trained 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.

IndicWhisper [19], 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.

SALSA [25], a hybrid ASR approach integrating Whisper's encoder-decoder with a decoder-only LLM (Llama2-7B) [23] to improve recognition in low-resource settings through synchronous decoding. It is fine-tuned on 10 hrs of FLEURS [26] Hindi dataset.

IndicConformer [27], a 130M parameter conformer-based ASR Hindi model combining convolution and transformer layers to support Indian languages with a unified subword-character tokenization strategy.

SeamlessM4T [28], a multilingual ASR and translation model covering 100 languages, leveraging 4.5 million hours of pre-training but with unexplored effectiveness in real-world low-resource speech settings.

3.3. LM/LLM Configurations

This section highlights how we use the LM/LLMs for automatic ASR post-correction. For fine-tuning and in-context learning, we have used the IndicVoice hypotheses generated from IndicWav2Vec and mapped them to the IndicVoice ground truths.

mT5: mT5 [22] belongs to the family T5 models, which is an encoder-decoder based LM. It is a multilingual version of the T5 [29] model. The mT5 model uses a subword-based tokenization strategy. It follows the SentencePiece tokenizer [30], which is a

Table 1: *Models Comparison across different KrishiVaani datasets*

Model	IndicWav2Vec	IndicWhisper	IndicConformer	SeamlessM4T	SALSA	KVWav2Vec
KrishiVaani-Known (WER)	23.7	24.2	23.42	49.94	87.59	**22.4**
KrishiVaani-Known (CER)	**8.5**	10.65	10.3	31.23	67.35	8.97
KrishiVaani-UnKnown (WER)	28.57	30.08	26.84	41.5	95.50	**26.02**
KrishiVaani-UnKnown (CER)	12.69	16.62	12.92	25	75.20	**11.81**
KrishiVaani-OOD (WER)	**22.41**	49.78	28.56	42.73	79.09	24.60
KrishiVaani-OOD (CER)	**8.51**	35.96	17.34	27.73	61.24	9.51

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

Model	KVWav2Vec	ByT5	mT5	Llama
KrishiVaani-Known	22.38	24.33	**22.29**	26.37
KrishiVaani-UnKnown	26.42	26.77	**25.57**	37.75
KrishiVaani-OOD	24.60	25.12	**24.35**	30.08

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

Experiment	Shots	KrishiVaani-Known	KrishiVaani-UnKnown	KrishiVaani-OOD
KVWav2Vec	-	22.38	26.04	24.60
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
+ 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**

variant of the BPE tokenizer. We selected the *mT5-small* variant as the correction model.

ByT5: ByT5 [21] 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.

Llama: Llama-3-Nanda-10B-Chat [23] (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.

ChatGPT: Powered by OpenAI, ChatGPT [13] 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 [31] to create sentence embedding for in-context learning.

4. Results and Discussions

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. These results suggest 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 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 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 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.

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

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

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.

6. References

[1] A. M. Deshmukh and R. Chalmeta, "User experience and usability of voice user interfaces: A systematic literature review," *Information*, vol. 15, no. 9, p. 579, 2024.

[2] D. Ghosh, T. Dube, and A. Shivaprasad, "Script recognition—a review," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 32, no. 12, pp. 2142–2161, 2010.

[3] D. Adiga, R. Kumar, A. Krishna, P. Jyothi, G. Ramakrishnan, and P. Goyal, "Automatic speech recognition in sanskrit: A new speech corpus and modelling insights," *arXiv preprint arXiv:2106.05852*, 2021.

[4] T. Javed, J. A. Nawale, E. I. George, S. Joshi, K. S. Bhogale, D. Mehendale, I. V. Sethi, A. Ananthanarayanan, H. Faquih, P. Palit *et al.*, "Indicvoices: Towards building an inclusive multilingual speech dataset for indian languages," *arXiv preprint arXiv:2403.01926*, 2024.

[5] K. Bhogale, A. Raman, T. Javed, S. Doddapaneni, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "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)*. IEEE, 2023, pp. 1–5.

[6] T. Javed, K. Bhogale, A. Raman, P. Kumar, A. Kunchukuttan, and M. M. Khapra, "Indicsuperb: A speech processing universal performance benchmark for indian languages," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 11, 2023, pp. 12 942–12 950.

[7] D. Kakwani, A. Kunchukuttan, S. Golla, N. Gokul, A. Bhattacharyya, M. M. Khapra, and P. Kumar, "Indicnlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 4948–4961.

[8] A. Gangwar, S. Umesh, R. Sarab, A. K. Dubey, G. Divakaran, S. V. Gangashetty *et al.*, "Spring-inx: A multilingual indian language speech corpus by spring lab, iit madras," *arXiv preprint arXiv:2310.14654*, 2023.

[9] R. Kumar, D. Adiga, M. Kothyari, J. Dalal, G. Ramakrishnan, and P. Jyothi, "Vagyojaka: An annotating and post-editing tool for automatic speech recognition." in *INTERSPEECH*, 2022, pp. 857–858.

[10] R. Kumar, D. Adiga, R. Ranjan, A. Krishna, G. Ramakrishnan, P. Goyal, and P. Jyothi, "Linguistically informed post-processing for asr error correction in sanskrit." in *INTERSPEECH*, 2022, pp. 2293–2297.

[11] C. Wang, S. Dai, Y. Wang, F. Yang, M. Qiu, K. Chen, W. Zhou, and J. Huang, "Arobert: An asr robust pre-trained language model for spoken language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1207–1218, 2022.

[12] R. Kumar, S. Ghosh, and G. Ramakrishnan, "Beyond common words: Enhancing asr cross-lingual proper noun recognition using large language models," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 6821–6828.

[13] R. Ma, M. Qian, P. Manakul, M. Gales, and K. Knill, "Can generative large language models perform asr error correction?" *arXiv preprint arXiv:2307.04172*, 2023.

[14] Y. Li, J. Yu, M. Zhang, M. Ren, Y. Zhao, X. Zhao, S. Tao, J. Su, and H. Yang, "Using large language model for end-to-end chinese asr and ner," *arXiv preprint arXiv:2401.11382*, 2024.

[15] S. Li, C. Chen, C. Y. Kwok, C. Chu, E. S. Chng, and H. Kawai, "Investigating asr error correction with large language model and multilingual 1-best hypotheses," in *Proc. Interspeech*, 2024, pp. 1315–1319.

[16] Y. Hu, C. Chen, C.-H. H. Yang, R. Li, C. Zhang, P.-Y. Chen, and E. Chng, "Large language models are efficient learners of noise-robust speech recognition," *arXiv preprint arXiv:2401.10446*, 2024.

[17] S. Ghosh, M. S. Rasooli, M. Levit, P. Wang, J. Xue, D. Manocha, and J. Li, "Failing forward: Improving generative error correction for asr with synthetic data and retrieval augmentation," *arXiv preprint arXiv:2410.13198*, 2024.

[18] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, "Pyannote. audio: neural building blocks for speaker diarization," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7124–7128.

[19] K. S. Bhogale, S. Sundaresan, A. Raman, T. Javed, M. M. Khapra, and P. Kumar, "Vistaar: Diverse benchmarks and training sets for indian language asr," *arXiv preprint arXiv:2305.15386*, 2023.

[20] E. Variani, T. Bagby, K. Lahouel, E. McDermott, and M. Bacchiani, "Sampled connectionist temporal classification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4959–4963.

[21] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 291–306, 2022.

[22] L. Xue, "mt5: A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020.

[23] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[24] Y. Madhani, S. Parthan, P. Bedekar, G. Nc, R. Khapra, A. Kunchukuttan, P. Kumar, and M. M. Khapra, "Aksharantar: Open indic-language transliteration datasets and models for the next billion users," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 40–57.

[25] A. Mittal, D. Prabhu, S. Sarawagi, and P. Jyothi, "Salsa: Speedy asr-llm synchronous aggregation," *arXiv preprint arXiv:2408.16542*, 2024.

[26] A. Conneau, M. Ma, S. Khanuja, Y. Zhang, V. Axelrod, S. Dalmia, J. Riesa, C. Rivera, and A. Bapna, "Fleurs: Few-shot learning evaluation of universal representations of speech," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 798–805.

[27] T. Javed, J. Nawale, S. Joshi, E. George, K. Bhogale, D. Mehendale, and M. M. Khapra, "Lahaja: A robust multi-accent benchmark for evaluating hindi asr systems," *arXiv preprint arXiv:2408.11440*, 2024.

[28] L. Barrault, Y.-A. Chung, M. C. Meglioli, D. Dale, N. Dong, M. Duppenthaler, P.-A. Duquenne, B. Ellis, H. Elsahar, J. Haaheim *et al.*, "Seamless: Multilingual expressive and streaming speech translation," *arXiv preprint arXiv:2312.05187*, 2023.

[29] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

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

[31] A. Joshi, A. Kajale, J. Gadre, S. Deode, and R. Joshi, "L3cube-mahasbert and hindsbert: Sentence bert models and benchmarking bert sentence representations for hindi and marathi," in *Science and Information Conference*. Springer, 2023, pp. 1184–1199.