International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

# Low Resource Language Model for Indian Regional Language

**Soundarya C S[1], Harish T A[2], Dr.Shantala C P[3]**

[1]M.Tech Student, Department of Computer Science and Engineering, Channabasaveshwara Institute of Technology, Gubbi,Tumkur
[2]Asst. Professor, Department of Computer Science and Engineering, Channabasaveshwara Institute of Technology, Gubbi,Tumkur
[3]Professor & Head, Department of Computer Science and Engineering, Channabasaveshwara Institute of Technology, Gubbi,Tumkur.

**Abstract**
Although many languages exist in India, they all cannot be said to have adequate support for proper language processing. The present project aims at designing a program which would aid language processing from selected Indian languages that have limited availability in digital format. The program will facilitate basic text-processing procedures such as language identification, sentence and word segmentation, stop-word elimination, word frequency count, and POS tagging.. To implement its functionality, the toolkit would use Unicode-based algorithms and simple rules-based methods, thus being able to work with several languages at once, for example, Hindi, Tamil, Telugu, Kannada, Bengali, etc. Moreover, other features like transliteration and translation would be available in order to provide additional comfort for the user. The GUI would be implemented via the Tkinter library allowing people to enter some text and conduct certain language processing actions, while watching the results. The goal of this project is to develop a simple multilingual toolkit for text processing without using complex models or large datasets.

**Keywords:** Natural Language Processing (NLP), Indian Regional Languages, Multilingual Text Processing, Rule-Based Language Processing

## INTRODUCTION

NLP is the branch of Artificial Intelligence that helps to make sense of and perform computations using human language. NLP algorithms can be defined as software or methods developed to help machines comprehend and analyze language content, whether text-based or speech-based. The main focus of this project is to create a light version of a tool kit of NLP which will handle Indian regional languages. This is because the vast majority of tools designed in NLP work mainly for English or other major languages used worldwide. There is a considerable divide created between the two types of language when it comes to their ability to be processed digitally. This is why this project aims at dealing with fundamental tasks such as language detection, word counting, etc.

There are unique characteristics about Indian languages which make it difficult to use for NLP purposes. There are various complexities in Indian languages because of the changes in tense, gender, number, and many other forms. Compound words and inflectional variations also contribute to the difficulty in pro-

cessing. Another challenge is that there is great diversity of scripts as each language has its own script with its own Unicode characters. Therefore, the system will be based on Unicode processing to enable processing multiple languages without requiring heavy dependence on language models. On the architectural side, the design of the system will be made to be simple, modular, and expandable. This implies that each functionality is separately implemented in a way that makes it easy to replace modules without interfering with others. For instance, rules used can easily be replaced with machine learning algorithms at some point in the future. A graphical user interface will also be incorporated to facilitate ease of interaction. Users will not need any command-line instructions but rather will interact by clicking buttons and viewing text output or visualization. The implementation combines core Python programming with optional external libraries to ensure lightweight performance and flexibility.

## RELATED WORK

Modern advancements in natural language processing largely depend on transformer-based models and large language systems such as BERT, GPT, and LLaMA. They proved themselves in such tasks as text classification, translation, sentiment analysis, and summarization. Nevertheless, there was an uneven development in high-resource languages, with English being the most common language used, while low-resource languages (LRLs), such as numerous Indian regional languages, were left aside [1].

The main issue in the field related to low-resource languages (LRLs) is the lack of sufficient training data. Most models based on large language architectures require massive amounts of annotated corpora, which are difficult to obtain for many regional languages. As a result, model performance decreases, leading to bias and poor generalization. As a consequence, the performance of models decreases, leading to bias and difficulties with generalization. Moreover, another problem is connected with the specifics of Indian languages' structure, namely its highly inflectional character, agglutination, and flexible syntactic constructions, along with issues such as code-switching, existence of dialects, and lack of digitized text materials [2]. The ways out that have been proposed include, among others, cross-lingual transfer learning, parameter-efficient fine-tuning (PEFT), and data augmentation. However, despite achieving some progress, they still face limitations in the absence of sufficient data and comprehensive linguistics representation of the target languages.. Adversarial robustness has emerged as a crucial concern in NLP systems, especially for low-resource languages where models already suffer from limited generalization. Kansal et al. [3] demonstrated that transformer-based models such as IndicBERT are highly vulnerable to adversarial attacks generated using the TextFooler framework. These attacks introduce minor perturbations, such as synonym substitutions, which preserve semantic meaning but significantly degrade model performance—often resulting in accuracy drops exceeding 30%. Morphologically rich languages like Bengali are particularly vulnerable due to their complex grammatical structures and higher variability in word forms. This increases the number of possible perturbation points, making models more susceptible to adversarial manipulation. Further, Wallace et al. [4] introduced the concept of universal adversarial triggers, which are fixed token sequences capable of misleading models across multiple inputs. This highlights fundamental weaknesses in neural architectures and emphasizes the need for robust evaluation mechanisms. Gupta & Nair [5] , showed the efficiency of SVM in classification applications, especially because of the efficiency of handling high dimensional data. Yet, these methods depend greatly on manual feature extraction processes, thus making it difficult for them to discover complex semantic relationships in textual data. To improve model robustness, researchers are exploring techniques such as adversarial training, language-aware tokenization, and morphology-based

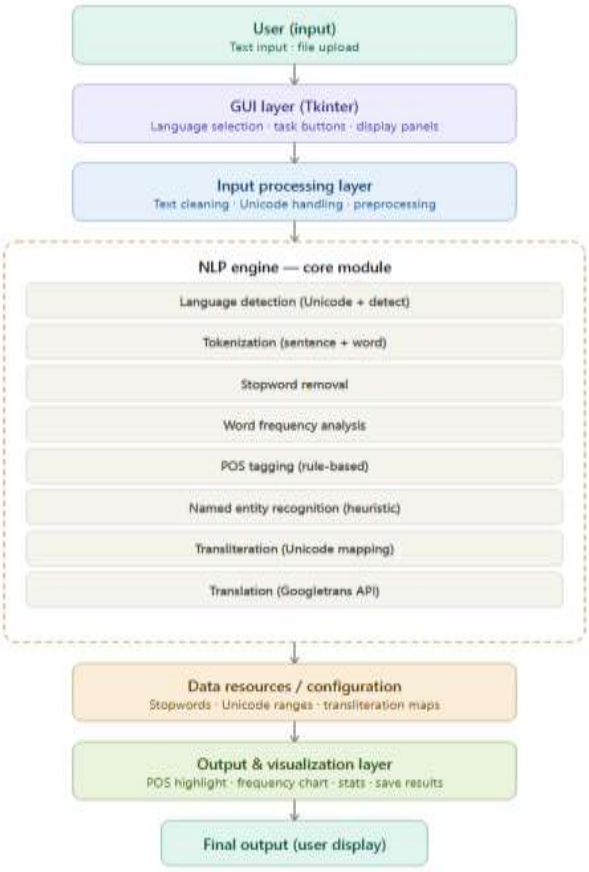
embeddings. These algorithms used techniques such as feature extraction using TF-IDF and bag-of-words. Hybrid models have been proposed to address these limitations by combining traditional machine learning with deep learning techniques such as CNN, LSTM, and GRU. Aqil et al. [6] showed that these models are more effective because they capture both local features and sequential patterns. However, they still face challenges such as complex preprocessing, spelling variations, and limited contextual understanding, especially in low-resource settings. Transformer-based models later addressed these issues by using attention mechanisms, as seen in models like mBERT and XLM-R. As mentioned by Hoque et al. [7], one approach that has been used in combining various models in order to enhance robustness and improve generalization is the one known as transformer stacking for detecting cyberbullying cases. Approaches such as preprocessing through addressing emojis and normalization of the language have also been emphasized in the proposed approach. However, the proposed approaches suffer from some limitations including the limited data availability, computational difficulty, and cross-language issues in LRLs. Tokenization is a fundamental component of NLP systems, particularly in multilingual environments. Common techniques such as Byte Pair Encoding (BPE) and WordPiece are primarily optimized for high-resource languages, leading to inefficient tokenization for Indic languages. Rana et al. [8] proposed the MUTANT tokenizer, which incorporates language-aware preprocessing and balanced vocabulary allocation.This approach reduces token fragmentation and improves computational efficiency by generating shorter token sequences. Efficient tokenization is crucial for transformer-based models, as longer sequences increase memory usage and computational complexity. Therefore, improving tokenization strategies is essential for scaling NLP systems in multilingual and low-resource settings. Transformer-based LLMs have shown significant improvements in several NLP tasks under low-resource conditions. In machine translation, Nair et al. [9] proposed a reinforcement learning-based framework using mT5, and incorporated optimization techniques such as Direct Preference Optimization (DPO). This method provides better translation quality on several evaluation metrics. In text summarization, multi-lingual transformer models generate more coherent and context-aware summaries as evident in SaralMarathi [10]. Furthermore, Sharma et al. [11] proposed IndicSuperTokenizer (IST) to improve the tokenization efficiency and reduce the computational overhead. Another common approach is cross-lingual transfer learning that is used to transfer knowledge from high-resource languages. These approaches, however, still suffer from challenges such as domain mismatch, high computational cost, and limited scalability. Multimodal learning has shown promise in addressing data scarcity for low-resource languages by leveraging multiple data sources like text, images and audio. The paper entitled “Language Flamingo: A Visual Language Model for Few-Shot Learning” [12] showed that by utilizing both visual and textual information, the performance in visual question answering and cross-modal reasoning tasks was greatly enhanced. Some approaches to multimodality were explored further in “Multimodality Research in NLP – a Survey , which covered topics like multimodal fusion, creation of synthetic data, and cross-modal transfer learning. Multimodal approaches offer better representation capabilities but suffer from issues like insufficient data, heavy computation costs, and lack of benchmark frameworks. Speech-based approaches in NLP play an essential role in increasing accessibility in multilingual and illiterate regions. The authors of [13] designed a highly accurate and fast multilingual wake-word detector based on deep learning techniques. Palivela et al. [15] demonstrated that multilingual Automatic Speech Recognition (ASR) systems outperform monolingual models in code-switching scenarios, which are common in Indian languages. Techniques such as transfer learning and self-supervised learning are used to improve performance. However, challenges such as limited

annotated speech datasets, phonetic diversity, and real-time processing constraints remain significant barriers.

## SYSTEM ARCHITECTURE

The system architecture of the proposed Indic NLP Toolkit is designed with a modular and layered approach to ensure both efficiency and ease of use. The architecture is divides into several part, each responsible for a specific function. At the top level, the user interface layer is developed using It provides a user-friendly interface where text can be entered and multiple NLP tasks can be executed easily.. The subsequent layer is the input processing layer wherein the typed text undergoes cleansing and normalization processes prior to its analysis. This ensures that the data is processed in a consistent form for precise analysis. The heart of the proposed system is the NLP Engine that comprises several modules performing specific tasks, including language detection, tokenization, stop word filtering, word frequency counting, part-of-speech tagging, named entity recognition[NER], transliteration, and translation.

**Figure 1 System Architecture of the Proposed Indian Regional Language NLP Toolkit.**




## IMPLEMENTATION

This research involves the development of Indic NLP Toolkit which helps in analyzing and processing Indian regional languages in low-resource settings. This implementation is mainly concerned with offering a variety of functions of natural language processing including language identification, tokenizing, part-of-speech tagging, transliterating, translating, and statistical processing using a graphical user interface. The implementation has been done using Python programming language and the approach is

modular in nature whereby every module performs some functions and works together with others to achieve desired results.

**5.1 Development Environment**

Python is chosen as the main programming language because it is easy to understand, flexible, and widely used for NLP tasks. Tkinter is used to develop the graphical user interface with buttons, menus, and text boxes for user interaction. NLTK, langdetect, and googletrans libraries are used for tokenization, language detection, and translation functionalities. Regular expressions, Counter, and unicodedata modules are used for text preprocessing, word frequency analysis, and multilingual Unicode handling.

**5.2 Language Support Module**

The language processing module that is part of the suggested approach has been created to account for the variability of Indian languages, especially under resource limitations. In this case, considering that the languages used in India vary in linguistic family and script, the scalability of the system becomes one of its critical requirements. Currently, ten Indian languages can be processed using this module, such as Hindi, Telugu, Tamil, Kannada, Bengali, Marathi, Gujarati, Malayalam, Punjabi, and Odia. Such a feature is provided by having specific metadata for each language.

**5.3 NLP Engine Implementation**

IndicNLPEngine controls all core Natural Language Processing operations in the system and acts as the main framework for text processing. The engine is designed in a modular manner where functions such as tokenization, stopword removal, transliteration, translation, POS tagging, and named entity recognition are implemented independently. The input text first undergoes preprocessing and advanced analysis, and the results are returned in structured formats like lists and dictionaries for easy visualization and maintenance. The modular and loosely coupled design also improves scalability, extensibility, and support for future enhancements.

**5.4 Language Detection**

Language detection is an important stage in NLP because it recognizes the language in which the input text is written prior to doing other tasks like tokenization, translation, and part-of-speech tagging.

**5.4.1. Unicode-Based Script Analysis**

Languages spoken in India have their respective writing styles, which are assigned different Unicode codes. Examples include Devanagari for languages such as Hindi and Marathi and languages like Tamil, Telugu, and Kannada that use separate writing styles. The software evaluates the input string character by character. Each character is then compared to pre-set Unicode code ranges representing each of the supported languages. Depending on this comparison, the number of characters in each writing style is determined, and the language with the highest number of characters identified.

**5.5 NLP Functionalities**

proposed toolkit integrates several core Natural Language Processing (NLP) functionalities to effectively process and analyze text written in Indian regional languages. These features are designed to handle multilingual input and extract meaningful linguistic information.

**5.5.1 Tokenization**

Tokenization is an important preprocessing step in NLP that converts raw text into meaningful units such as sentences and words. In this project, sentence tokenization is performed using punctuation de-

limiters including ".", "!", "?", and Indic-specific symbols like "।" and "॥". Word tokenization is carried out using regular expression-based splitting, followed by removal of punctuation and unnecessary spaces. The tokenizer generates structured outputs such as lists of sentences, words, and basic statistics like sentence and word count.

**5.5.2 Stopword Removal**

Stopword removal is an important preprocessing technique used to eliminate common words that carry little meaning in text analysis. Words such as "is", "the", "and" in English and "है", "और", "के" in Hindi are removed using predefined stopword lists. The input text is tokenized and compared with the selected language's stopword list, and matching words are discarded. This process improves tasks such as keyword extraction, word frequency analysis, and text summarization by reducing unnecessary noise.

**5.5.3 Word Frequency Analysis**

Word frequency analysis is used to identify the most commonly occurring words in a text. The process begins with tokenization and optional stopword removal, after which Python's Counter function is used to count the occurrences of each word. The results are arranged in descending order based on frequency. This analysis helps in understanding the main content and identifying important terms within the text.

**5.5.4 POS Tagging (Part-of-Speech Tagging)**

POS tagging assigns grammatical categories such as noun, verb, and adjective to words in a sentence. A rule-based approach is used in this project due to limited datasets for Indian languages. The system mainly relies on suffix patterns in Devanagari-based languages such as Hindi and Marathi. Each tokenized word is checked against predefined linguistic rules, and appropriate tags are assigned. Numerical values and punctuation symbols are also identified separately.

**5.5.5 Named Entity Recognition (NER)**

Named Entity Recognition is used to identify and classify entities such as person names, locations, and proper nouns from the input text. The system uses rule-based pattern matching, especially suffix-based identification for location names. The extracted entities are stored in a structured format along with their corresponding categories. This approach provides a simple and computationally efficient solution for information extraction in low-resource language environments.

**5.5.6 Transliteration**

The transliteration module converts text written in Indic scripts into Roman or English script while preserving pronunciation. Unlike translation, transliteration changes only the script and not the meaning of the text. This feature improves readability and helps users understand Indic language text more easily.

**5.5.7 Machine Translation**

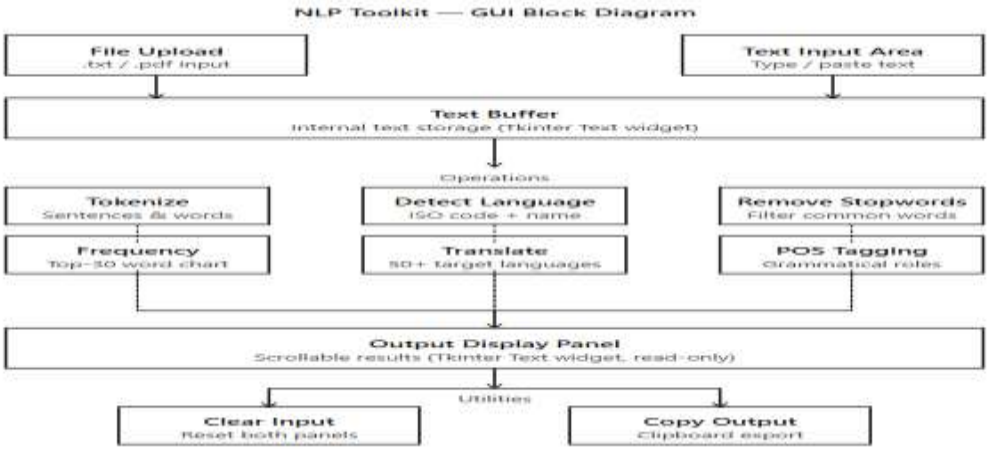
The machine translation module translates text from one language to another based on the user's selected source and target languages. Automatic language detection can also be used when the source language is unknown. Language names are converted into ISO codes for processing translations. The system includes error handling mechanisms to manage issues such as connection failures during translation.

**5.5.8 Text Statistics**

The text statistics module provides quantitative analysis of the input text. It calculates values such as total characters, words, sentences, unique words, average word length, lexical diversity, and average words per sentence. These statistics help in understanding the structure, complexity, and richness of the text content.

### 5.6 Graphical User Interface (GUI)

**Figure 2 GUI Architecture of the Proposed System.**




## RESULTS

**Figure 3: System Interface Showing Initial State.**

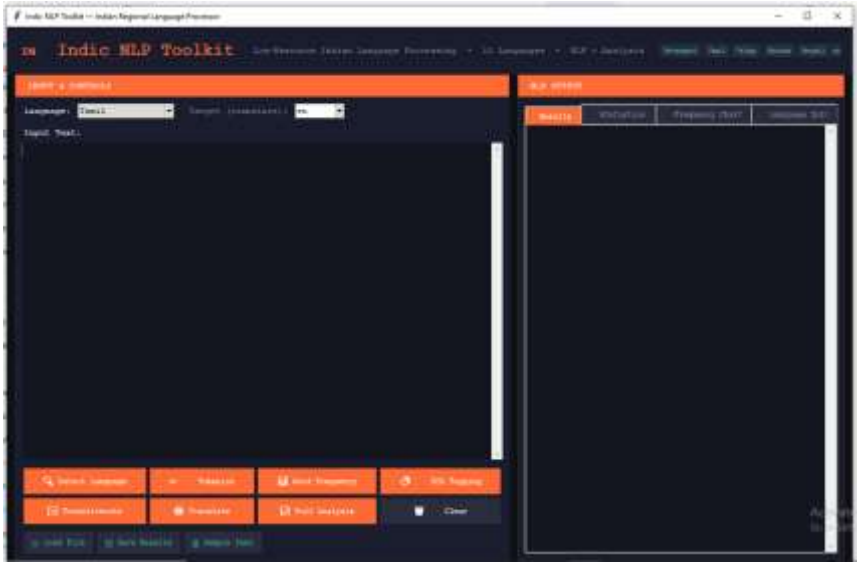



**Figure 4 Giving Input Text as a Tamil to Translate it into English.**

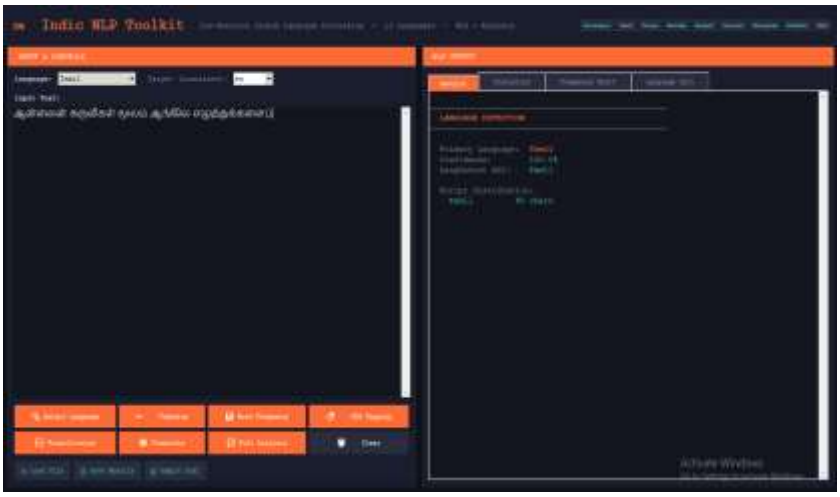


**Figure 5 Full NLP Analysis Output of the indic NLP Toolkit for Indian Regional Language Text.**


## CONCLUSION

The present project revolves around developing an Indic NLP Toolkit, which is intended to deal with Indian languages that lack computing power. The toolkit incorporates a variety of NLP functions like language identification, tokenization, counting word frequencies, part-of-speech tagging, transliteration, and translation in one single application. As evidenced by the outcomes, the toolkit successfully processes multilingual texts and produces some useful results like linguistic observations and translations. The modular structure of the toolkit makes it easier to use and extend., users can conduct either separate or various combined operations. Even though the current development employs mostly rule-based and lightweight approaches to language processing, the achieved outcomes are quite satisfactory to conduct simple NLP tasks. Therefore, this observation suggests that efficient language processing does not always require highly sophisticated models or large volumes of data. Overall, the project demonstrates the practical applicability of basic NLP techniques for Indian languages.

## REFERENCES

1. N. Unjum, S. Evert, K. Sarveswaran, and R. Weerasinghe, “LLMs for Low-Resource Languages: A Survey,” arXiv preprint, 2025.
2. E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, “On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?” in Proc. ACM Conf. Fairness, Accountability, Transparency (FAccT), 2021.

3. K. Kansal, A. Sharma, and R. Gupta, "Adversarial Attacks on Indian Language Models using TextFooler," in Proc. Int. Conf. Natural Language Processing, 2023.
4. E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal Adversarial Triggers for Attacking and Analyzing NLP," in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), 2019, pp. 2153–2162.
5. D. Gupta and A. R. Nair, "Text classification for Indian languages using Machine Learning Techniques," Int. J. Comput. Appl., vol. 176, no. 32, pp. 1–5, 2020.
6. M. Aqil, S. Ahmed, and F. Khan, "Hybrid Text Summarization Techniques for Urdu Language," Journal of Information Processing Systems, vol. 17, no. 4, pp. 789–803, 2021.
7. M. M. Hoque, M. S. Islam, and M. A. Rahman, "Transformer-Based Cyberbullying Detection for Low-Resource Languages," IEEE Access, vol. 10, pp. 1–12, 2022.
8. R. Rana, P. Singh, and A. Verma, "MUTANT: A Multilingual Tokenization Framework for Low-Resource Languages," in Proc. ACL Workshop on Multilingual NLP, 2023.
9. A. Nair, S. Krishnan, and P. Menon, "Reinforcement Learning-Based Machine Translation for Low-Resource Languages," in Proc. Int. Conf. Artificial Intelligence and Data Science, 2024.
10. S. Kulkarni, R. Patil, and M. Deshmukh, "Abstractive Text Summarization for Marathi using Transformer Models," in Proc. Int. Conf. Computational Linguistics, 2023.
11. A. Sharma, R. Gupta, and V. Kumar, "IndicSuperTokenizer: A Tokenization Framework for Indic Languages," arXiv preprint, 2024.
12. J.-B. Alayrac et al., "Flamingo: A Visual Language Model for Few-Shot Learning," in Advances in Neural Information Processing Systems (NeurIPS), 2022.
13. A. Lupascu, D. Ionescu, and M. Popescu, "A Survey on Large Multimodal Models for Low-Resource Languages," IEEE Access, vol. 11, pp. 1–15, 2023.
14. P. Jaybhaye, S. Patil, and R. Kulkarni, "Multilingual Wake Word Detection using Deep Learning," in Proc. IEEE Int. Conf. Signal Processing and Communication, 2022.
15. K. Palivela, R. Reddy, and S. Kumar, "Automatic Speech Recognition for Code-Switched Indian Languages," in Proc. INTERSPEECH, 2023.