International Journal of Advanced and Applied Sciences

Journal homepage: http://www.ijaas.in

Analyzing Dialectal Variations in Odia: Computational Approaches to Native Language Identification

Biswaranjan Bhukta [1*], Dr. Manoj Kumar Jena [2], Dr. Pabitrananda Patnaik [3]

[1] *Research Scholar, Department of Computer Science & Application, GIET University, Gunupur, India*
[2] *Professor, Department of Computer Science & Application, GIET University, Gunupur, India*
[3] *Scientist F/Sr Technical Director, Department of IT, National Informatics Centre, Government of India, Odisha, India*

ARTICLE INFO

Article history:
Received: 05-08-2025
Received in revised form: 19-09-2025
Accepted: 03-10-2025

Keywords:
Dialectal Variations, Odia Language, Native Language Identification, Computational Approaches, Machine Learning, Natural Language Processing (NLP), Language Identification, Speech Recognition, Regional Dialects, Linguistic Analysis.

ABSTRACT

This study explores computational approaches to analyzing dialectal variations in the Odia language, with a focus on native language identification. Odia, a prominent language spoken in India, exhibits significant regional dialectal diversity, which can pose challenges in automatic language processing tasks. This research investigates various computational techniques for identifying native Odia speakers by analyzing dialectal features, such as phonetic, lexical, and syntactic variations, across different regions. We employ machine learning models, including supervised and unsupervised learning, to classify and distinguish between different dialects of Odia. By leveraging natural language processing (NLP) tools and a large corpus of dialectal data, the study aims to enhance the accuracy of language identification systems, which are critical for applications in speech recognition, language modeling, and automated translation. The findings highlight the potential of computational methods to capture the subtle nuances of dialectal differences and improve the performance of language processing systems for less-resourced languages like Odia.

Introduction

God created human beings from the elements of the earth and endowed them with a higher sense of self-esteem than the rest of creation. Humans are made to have a relationship with God, to live in harmony with one another, and to care for the rest of creation. Among all living beings, humans are remarkable because they can think for themselves as well as for others. They possess the intelligence to generate new ideas and the ability to apply those ideas in their daily lives, making life easier and more comfortable.

Among all forms of communication between human beings, the most natural and primary

means is speech. There are several ways a person can interact with others such as eye contact, facial expressions, voice variations, and gestures but the most widely used and effective method for exchanging information, emotions, and thoughts is speech. Over the past few decades, Speech-to-Text (STT) systems have been extensively used in various applications across industry and academia. In the field of education, and particularly for deaf or mute students, STT or speech recognition systems serve as a highly effective means of communication [1].

The most challenging task in speech processing is enabling a machine to recognize speech in a specific language. Speech recognition is the process of converting spoken waveforms into continuous written words or text using computational techniques. In recent years, speaker recognition and verification have also received significant attention. There are several reasons for this growing interest. For instance, speech (i) offers a convenient and natural form of input, (ii) carries a substantial amount of speaker-specific information, and (iii) is relatively inexpensive to collect and analyze.

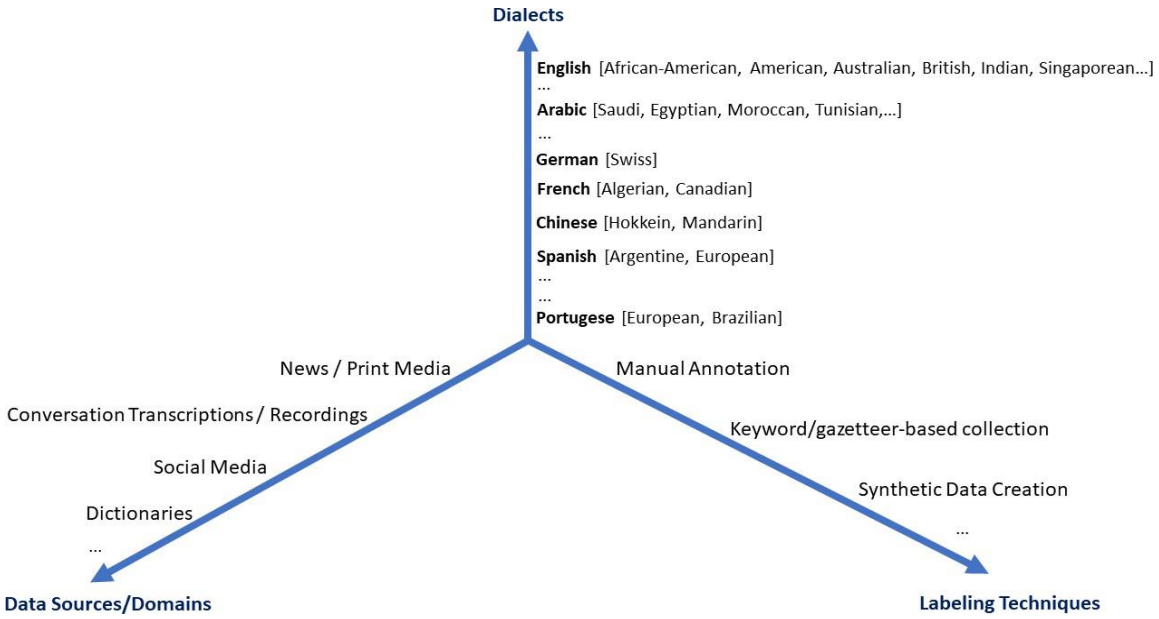

Figure 1: Overview of Dataset Creation Approaches

Despite these advantages, significant challenges related to system robustness remain, and these must be resolved before speaker recognition technology can be reliably deployed in real-world environments. Recent progress in STT research has been driven by the development of efficient machine-based systems capable of interpreting human spoken communication. Speech processing is the most natural form of human interaction and is considered one of the most dynamic and impactful areas of research in signal processing. Generally, speech is natural, simple, fast, hands-free, and requires no technical expertise.

The primary goal of STT conversion is to develop techniques and systems that can take speech as input and produce an accurate transcript. People feel more comfortable communicating directly with machines rather than relying on traditional interfaces such as keyboards, mice, and other input devices [2]. These basic interfaces often require a certain level of skill for effective use. For example, operating a mouse demands good hand eye coordination, making it difficult for visually impaired individuals to use a computer.

Furthermore, current computer interfaces require the user to possess a certain level of prior knowledge. This assumes that the operator has adequate proficiency in English as well as typing skills. In this context, speech interfaces help address these concerns. Today, speech technologies are widely available for a limited but significant range of tasks. These technologies enable machines to respond accurately and consistently to human voices, providing convenient and valuable services whenever needed. Communicating with a computer through speech is generally faster than using input devices such as keyboards. As a result, users tend to prefer systems that allow quicker machine interaction. After all, communication among human beings is predominantly governed by spoken language.

Therefore, it is natural for people to expect voice-based interfaces when interacting with machines. This can be achieved through an effective Speech-to-Text (STT) system that enables a machine to interpret voice commands and dictation and convert them into text [3]. STT is the process of transforming an audio signal captured by a microphone into its corresponding transcription. STT conversion is an application of speech recognition. The major

challenges in implementing an STT system arise from variations in human speaking styles and the presence of environmental noise. Thus, the goal of an STT system is to convert speech into an accurate text message that is independent of the device, speaker, or surrounding conditions. Such systems also enhance accessibility by providing data-entry alternatives for individuals with physical disabilities. For example, deaf individuals cannot hear but can understand information through text. Additionally, storing audio files for a series of lectures requires far more memory than storing their corresponding text transcriptions.

A person can benefit greatly from using voice commands to interact with living room systems, vehicle systems, and other smart environments. Speech-based interaction is also highly valuable in the medical field. For example, office staff and doctors do not need to rely on keyboards to enter patient information. The software can learn to recognize a doctor's unique speech patterns. Dictation, voice recognition, and transcription can work together within a workflow management system to boost productivity, enhance employee collaboration, and save significant time each day. Speech processing also involves the study of the acoustic characteristics of speech, including both normal and abnormal speech signals. This field examines the physical properties of spoken language.

These analyses include waveform analysis, FFT or LPC analysis, voice onset time (VOT) measurements, formant frequency measurements, and others. The captured speech is an analog signal, which cannot be directly fed into an STT conversion system. Therefore, the signal must first be converted into a digital waveform before processing can occur. Sampling is performed to divide the signal into frames, based on the assumption that speech remains relatively constant within each sampling interval. After sampling the input speech waveform, acoustic features are extracted from each of the resulting digital frames for further analysis [5].

When considering the task of speech recognition, a raw audio signal captured by a microphone or recording software is too complex to use in its original form. It must be transformed into a more meaningful representation. Initially, the received audio is divided into a sequence of frames at uniform time intervals typically around 10 milliseconds. These frames must be evaluated so that perceptually significant information can be extracted. This process is

guided by the sensitivity of speech signals and the need to eliminate redundant data. The acoustic feature extraction step produces a set of feature parameters from the speech signal, each corresponding to meaningful aspects of the waveform. These parameters are organized into feature vectors. The primary goal of the parameter extraction stage is to retain essential information while discarding irrelevant details.

The acoustic signal is divided into frames, typically ranging from 10 to 25 milliseconds, with overlapping between consecutive frames. Each frame is then multiplied by a window function, and various feature extraction techniques are applied to generate the feature vectors. Common methods include MFCC, LPC, PLP, wavelet-based features, and RASTA-PLP. This stage also involves modeling the acoustic characteristics of phonemes recognized by the system. Such models are built through a training process using large volumes of labeled audio data. They are generally specific to a particular language and may also be adapted to certain accents.

The greatest challenge is that phonemes are context-dependent, meaning they tend to sound different depending on the phonemes that come before and after them. Each context-dependent phoneme is modeled using a Hidden Markov Model (HMM) [6]. The states of the HMM describe how the sound of a phoneme evolves over time. While the number of states can vary depending on the model design, most HMM-based systems typically use five states.

Literature review

Abdin et al. (2024) [1] A highly capable language model locally on your phone This technical report introduces Phi-3, a compact yet highly capable language model optimized for on-device performance, especially mobile phones. The study emphasizes performance-efficiency trade-offs, presenting Phi-3 as a breakthrough in making large language models accessible without cloud dependency.

Acharya et al. (2025) [2] Detection of language, hate speech, and targets using fasttext and BERT, The authors present a system for multilingual hate speech detection in Devanagari-script languages using a combination of FastText and BERT models. It was developed as part of the CHiPSAL 2025 shared task, demonstrating

strong performance across multiple classification tasks.

Aralikatte et al. (2021) [3] This paper introduces Itihasa, a comprehensive parallel corpus for Sanskrit-to-English translation, filling a major resource gap in low-resource machine translation. The dataset enables training and evaluation of neural translation systems for the ancient language.

Barbieri et al. (2020) [4] Tweeteval: Unified benchmark and comparative evaluation for tweet classification The TweetEval benchmark offers standardized datasets and evaluation metrics for various tweet-level NLP tasks such as sentiment analysis and hate speech detection. It facilitates consistent benchmarking across models and has become a popular resource in social media NLP research.

Basile et al. (2019) [5] SemEval-2019 Task 5: Multilingual detection of hate speech against immigrants and women in Twitter This SemEval shared task focuses on hate speech detection in English and Spanish tweets, especially targeting immigrants and women. The dataset and evaluation framework have been widely used to benchmark hate speech classification models in multilingual settings.

Chakraborty et al. (2025) [6] One_by_zero@nlu of devanagari script languages 2025: Target identification for hate speech leveraging transformer-based approach This study presents a transformer-based approach to detect hate speech targets in Devanagari-script languages. Participating in the CHiPSAL 2025 shared task, the system leverages attention-based architectures to achieve robust target identification.

Conneau et al. (2020) [7] Unsupervised cross-lingual representation learning at scaleThe authors propose a large-scale unsupervised method for learning multilingual sentence representations. This work underpins models like XLM-R and shows that cross-lingual embeddings can be learned without parallel corpora, significantly advancing multilingual NLP.

Devlin et al. (2019) [8] BERT: Pre-training of deep bidirectional transformers for language understanding This foundational paper introduces BERT, a pre-trained transformer model that achieves state-of-the-art performance on a variety of NLP tasks. BERT's bidirectional training method marks a paradigm shift in how language models understand context.

Mistral AI & NVIDIA (2024) [9] Mistral-Nemo-Instruct-2407This online release documents a collaborative model from Mistral AI and NVIDIA, aimed at instruction-following language generation. While informal, the model page provides important insights into architecture, fine-tuning methods, and inference capabilities for real-world deployment.

Rahman Aodhora et al. (2025) [10] Cuet_hateshield@nlu of devanagari script languages 2025: Transformer-based hate speech detection in devanagari script languages This system leverages transformer architectures (like BERT variants) for hate speech classification in Devanagari-based languages. It was developed as part of the CHiPSAL 2025 shared task and emphasizes linguistic diversity in hate speech datasets.

Doddapaneni et al. (2022) [11] Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages This arXiv preprint introduces large-scale efforts to build monolingual corpora and language models for 20+ Indic languages. The initiative focuses on resource creation, benchmark tasks, and model development to support inclusive NLP research for underrepresented languages.

Doddapaneni et al. (2023) [12] Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages An expanded version of the 2022 preprint, published at ACL 2023, provides deeper evaluations and baseline models.

Gupta & Arora (2022) [13] Stemming techniques on English language and Devanagari script: This review article explores and compares various stemming techniques used for English and Devanagari-script languages. It evaluates rule-based, statistical, and hybrid approaches, identifying key challenges in morphological normalization for Hindi and related languages.

Gupta, Singhal & Wasi (2025) [14] Iitr-ciol@nlu of devanagari script languages 2025: Multilingual hate speech detection and target identification in devanagari-scripted languages Submitted to CHiPSAL 2025, this paper presents a multilingual system for detecting hate speech and identifying its targets using transformer-based models across Devanagari-scripted languages, contributing to automated content moderation in multilingual contexts.

Gupta et al. (2022) [15] The authors present a scalable system for detecting abusive content across multiple Indic languages. Using large datasets and advanced deep learning techniques, the work sets new benchmarks in multilingual abusive language detection, as published in NeurIPS.

Guragain et al. (2025) [16] Nlpineers@nlu of devanagari script languages 2025: Hate speech detection using ensembling of BERT-based models This CHiPSAL 2025 entry uses an ensemble of BERT-based models for hate speech detection in Devanagari languages. The approach boosts robustness and accuracy, showcasing the effectiveness of model ensembling in multilingual NLP tasks.

Hong, Lee & Thorne (2024) [17] Reference-free monolithic preference optimization with odds ratio. This preprint proposes a novel method for preference optimization in language models without relying on explicit reference texts. The method improves alignment with human preferences using odds ratios as a ranking metric, advancing RLHF techniques.

Hossan et al. (2025) [18] Cuet_big_o@nlu of devanagari script languages 2025: Identifying script language and detecting hate speech using deep learning and transformer model Developed for CHiPSAL 2025, this system combines deep learning and transformer-based methods for language identification and hate speech detection in Devanagari scripts. The model addresses multilingual challenges in South Asian online content moderation.

Hu et al. (2021) [19] LoRA: Low-rank adaptation of large language modelsThe LoRA technique introduces efficient fine-tuning of large language models using low-rank updates. This lightweight method significantly reduces memory and computation costs, making model adaptation more accessible and scalable.

Ibrahim (2025) [20] Cufe@nlu of devanagari script languages 2025: Language identification using fast text This CHiPSAL 2025 contribution employs Fast Text for language identification across Devanagari-scripted languages. The approach demonstrates strong baseline performance with fast and efficient inference, ideal for real-time multilingual applications.

The performance of an STT system is heavily influenced by fluctuations in speech signals, with both the testing and training environments playing crucial roles in

accurate speech interpretation. Over the past few decades, extensive research in speech recognition has been conducted worldwide, leading to major advancements in both recognition and understanding technologies. Much of this technological progress has focused on enhancing the overall robustness and accuracy of recognition systems. The potential for STT development in Indian languages is especially promising. Although India's literacy rate exceeds 65%, fewer than 6% of the population uses English for communication. With the internet becoming a universal necessity, it is essential that access to information be available in formats understandable to the vast majority of the population. Without regional language support, nearly 95% of Indians are unable to fully benefit from the digital revolution. Providing data and interfaces in local languages could enable India to harness the full potential of STT technology and compete more effectively with technologically advanced nations. Currently, however, there is no standardized input system for many Indian languages. From an Indian language perspective with more than 2,371 dialects across diverse speech forms STT conversion holds significant potential and practical relevance. Current research places particular emphasis on the Odia language. Today, speech-based search has become one of the primary and simplest methods for accessing information on the internet using smartphones. The central challenge, however, lies in enabling computers to accurately recognize the voices of numerous speakers and convert them into corresponding text. STT technology refers to a system's capability to identify individual words or groups of words in any language and transform them into textual output.

Designing a mechanism that replicates human abilities especially the capacity to speak and respond has long captivated engineers and researchers. Voice technologies have undergone remarkable evolution, progressing from early speech machines using resonance tubes, to G. Bell's recording devices, Dictaphone tools, and the first speech synthesizer, VODER, eventually leading to modern intelligent assistants such as Apple's Siri, Microsoft's Cortana, and Amazon's Alexa. Thanks to advancements in artificial intelligence, speech-to-text conversion has rapidly gained popularity and practical application.

The fundamental concept behind STT systems is to convert recorded audio into a sequence of words, functioning as an alternative to typing through traditional

input devices like keyboards. STT technologies are widely used in various applications, from assisting individuals with physical disabilities and transcribing conversations, to language learning, hands-free navigation, and voice-activated file retrieval. Their expanding utility underscores the growing importance of speech-driven interfaces in the digital age.

STT systems enable the interaction of the user with the machine by enabling hands-free demands. In the last part of the year 2009, feed forward networks that are non-recurrent for acoustic modelling were developed.

Methodology

An STT system is used to convert spoken input into its corresponding text. Beyond transcription, automatic speech recognition (ASR) is also employed for user authentication through voice and for executing tasks based on spoken commands. ASR represents one of the most dynamic applications of speech processing, playing a vital role in human–machine interaction. It is particularly essential in voice-controlled operations, automatic debit and credit card activation, and various security and investigative services. However, major challenges in implementing an effective STT system arise from variations in human speaking styles and the presence of environmental noise. Therefore, the primary goal of an STT system is to convert speech into accurate text that remains independent of the device, speaker, or surrounding conditions. Additionally, STT technology enhances accessibility by providing alternative data-entry methods for individuals with physical disabilities. For instance, deaf individuals may not hear audio but can understand information through text. Moreover, text files require significantly less storage space than audio recordings of the same content, making STT useful for efficiently archiving lectures and other spoken materials. Voice commands significantly enhance user interaction with systems in living rooms, vehicles, and other smart environments. This technology is also highly beneficial in the medical field. For instance, doctors and their office staff can record patient information without relying on a keyboard, as STT software learns to recognize and adapt to a physician's unique speech patterns. Using STT systems, callers' voice messages can be automatically converted into text. Dictation, voice recognition, and transcription tools can seamlessly integrate with workflow

management systems to boost productivity, improve collaboration, and save substantial time. Today, ASR-based applications such as Google Assistant, Apple's Siri, and Microsoft's Cortana are embedded in smartphones, laptops, and tablets, making daily tasks easier and more efficient. One common application of STT technology is automatic spoken digit recognition, which is widely used in designing voice dialer systems. STT systems are particularly valuable for individuals with disabilities, such as visually impaired persons, and for elderly users, allowing them to make phone calls without physically dialing numbers. Numerous research studies have been conducted to develop digit recognition systems for different languages [11].

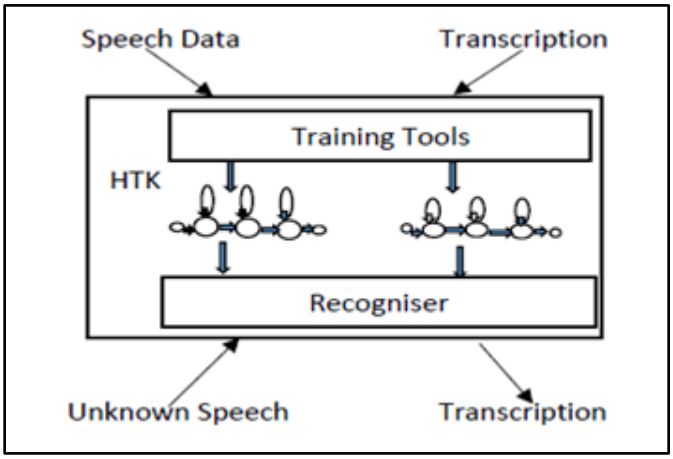

Figure 2: Overview of HTK Toolkit

Similarly, isolated word recognition and spoken digit recognition are widely used in applications such as automated data entry, OTP input for various services, banking system automation, and voice-based auto-dialing systems. Many researchers have developed isolated word recognition systems for regional languages. For example, M. Dua et al. developed an ASR system for the Punjabi language using the HTK toolkit based on Hidden Markov Models (HMM). The model was trained on 115 discrete Punjabi words recorded from eight different speakers and tested in real-world conditions using voice samples from six additional speakers. In Assamese, a technique for isolated word recognition has been developed that supports both speaker-dependent and speaker-independent modes. The system extracts distinctive features from

spoken Assamese words and uses a corpus of 100 commonly used words, achieving high accuracy. Karim et al. focused on Bengali vowel recognition from spoken alphabets, employing the Linear Predictive Coding (LPC) method to extract fundamental sound units, or phonemes, from the recorded letters. Isolated Oriya word recognition was carried out by S. Mohanty et al., who developed an HMM-based speech recognition model. In their study, 1,800 isolated Oriya words were collected from 30 speakers for training, and testing was conducted with 5 additional speakers, achieving a word recognition accuracy of 76.23%. A system for both speaker identification ("who has said it") and speech recognition ("what has been said") was also developed, employing SVM for speaker identification and HMM for voice recognition. A. Mohamed et al. proposed a framework for independent, small-word, continuous speech recognition. Firoze et al. developed a hybrid approach combining fuzzy logic with artificial neural networks (ANN) for Bengali voice recognition, effectively addressing language ambiguity issues in Bengali. Additionally, a digit speech recognizer for Hindi was developed, capable of functioning in both noisy and noiseless environments. The acoustic models for this system were trained using recordings of 10 Hindi digits from 8 different speakers. The recognition performance was evaluated using the HTK toolkit, assessing effectiveness at both the word and phoneme levels. Kurian et al. developed a continuous, speaker-independent digit recognition system for the Malayalam language, employing Perceptual Linear Prediction (PLP) cepstral coefficients for speech feature extraction and continuous-density HMMs for recognition. Similarly, Malayalam digit recognition and its applications have been explored using Mel-Frequency Cepstral Coefficients (MFCC) as features with HMM for recognition [12]. A word-level model for isolated digit recognition in Marathi using HTK has also been proposed.

For Odia, a corpus for isolated digit recognition was created using recordings from ten speakers, comprising five male and five female participants. Each speaker recorded ten Odia digits, from "SUNA" to "NAA," ten times each, resulting in a total of 1,000 recorded signals (10 speakers × 10 digits × 10 repetitions). Short pauses were maintained before and after each digit to ensure accurate segmentation of the speech signals.

The ***HGraf*** of HTK tool which shows the graphical interface was used for recording and labelling the signal. The “Rec” is used to start there cording and the “Stop” button is used for stopping recoded signal. Each signal is logged with the sampling frequency of 16000Hz and with each sample of 16 bits. Each recording signal contains some silence followed by the actual digit spoken followed by a short silence. After the recording was over each signal was being labelled with “sil”, " digit” and “sil” respectively by choosing the appropriate region of the recorded signal.

In our work on Odia digit recognition, Mel Frequency Cepstral Coefficients (MFCCs) are employed for acoustic feature extraction. Among the various techniques available, MFCCs are widely preferred because they closely approximate human auditory perception. For each signal frame, the first 12 MFCC coefficients along with the "null" coefficient (c0) are extracted. In addition, 13 delta coefficients representing the first-order derivatives of [c0, c1,…, c12], and 13 acceleration coefficients representing their second-order derivatives, are computed. This results in a total of 39 features for every frame of the recorded Odia digit waveforms. The acoustic feature vectors are generated from the original speech signals using the **H Copy** tool from the HTK toolkit.

Table1: Kernels and corresponding mapping functions of SVM

Kernel	**Kernel function**
Linear	$K_l(x,z)=x^T z_i$ i
Polynomial	$K_l(x,z)=(x^T z_i+\gamma)^k$k: degree of polynomial i
RBF	$K_l(x,z)=\exp(-\gamma\|\|x_i-z_i\|\|)^2$
Sigmoid	$K_l(x,z)=\tanh(\gamma x^T z_i+1)$ i

In our work, 11 Hidden Markov Models (HMMs):10 Odia digits and a “Silence” are to be modelled. In the first step, a priori topology for each HMM has to be

considered which includes the number of states, observation probability matrix and the transition probability matrix. We considered the identical topology for individual 11 HMMS. The basic prototype model of HMM actually consists of 6 states. Out of them, there are four dynamic states {S2, S3, S4, S5}: the initial and final states are non-emitting. The observation probability is denoted with ***bi*** is a diagonal matrix which is considered as a Gaussian distribution. The transition probabilities between states of ajand ajare mentioned as ***aij***. An HMM model is described with an equivalent text description file.

The **DIGIT** non-terminal is replaced by any Odia digit during recognition. Long silence or no-silence segments are represented using braces { }. If no sound corresponding to a digit is detected, the system identifies it as silence. For each grammar symbol **EKA, DUI, …, NAA, SUNA, START_SIL**, and **END_SIL** a corresponding HMM must be defined. These definitions are stored in a task dictionary file. In this file, the entries on the left represent the names of the grammar non-terminals, while the entries on the right indicate the associated HMM models used for recognition.

Table 2: Performance of isolated digit recognition rate using different kernel mapping

Kernel Mapping function	Average Recognition Rate (%)	Support Vector Points
Polynomial	81.75	920
Sigmoid	85.77	840
RBF	89.5	720

Prior to performance evaluation, two master label files were created. The first file, named **"ref. mlf"**, contains the "correct" transcriptions for the entire test corpus, which were generated through manual labeling. A Python script was used to produce this reference file. The second master label file, **"rec. mlf"**, is generated by the recognizer and contains the predicted

labels. Performance evaluation is carried out by comparing these two files.

The **H Results** tool from HTK is used to measure recognition performance by comparing **ref. mlf** and **rec. mlf**. The evaluation considers the number of words in the test corpus (NNN), along with the counts of deletions (DDD), substitutions (SSS), and insertions (III) required to match the recognized output to the reference transcription. These metrics are used to calculate the overall accuracy of the system

Properly recognized test data is represented as **#H = 8**, indicating eight correctly recognized samples. Similarly, **#S = 2** denotes the number of errors due to substitutions, and **#N = 10** represents the total number of test samples evaluated at a given time. The remaining nine samples were tested sequentially using our Odia digit recognition system. The average recognition rate across all samples was found to be **80%**. For comparison, isolated Hindi digit recognition using only two speakers achieved **85%**, while isolated Malayalam digit recognition reached **80%**.

Confusion Matrix: The confusion matrix provides a comprehensive overview of the model's performance. It is an N×NN \times NN×N matrix, where NNN is the number of target classes, used to evaluate the accuracy of a classification model. Each element of the matrix compares the actual target values with those predicted by the model, offering a detailed summary of correct predictions and the types of errors made. This allows for an in-depth assessment of the proposed classification model's effectiveness and highlights areas where misclassifications occur.

For our proposed **IODR-HMM model**, a total of **1,000 signal files** were recorded, with **100 files corresponding to each Odia digit**. For training purposes, **80% of the signals for each digit** were used, while the remaining **20% (i.e., 200 signals)** were reserved for testing.

Result analysis

High-performance **speech-to-text (STT)** and **text-to-speech (TTS)** systems are crucial for creating effective human–machine interfaces. Speech recognition has become an integral part of daily life, appearing in smartphones, gaming consoles, and smart watches. Devices such as the Amazon Echo Dot, priced under $53, enable users to order pizza, play favorite songs, check weather reports, or even book flight

tickets simply through voice commands. In today's world, speech-enabled products like **"Ok Google," Apple's Siri, Amazon's Alexa,** and **Microsoft's Cortana** leverage voice command technology to perform tasks such as searching for information, making them increasingly popular among end-users. Most Android phones now offer fully hands-free services via Google Assistant. In particular, **keyword spotting** is a powerful technique that facilitates hands-free operation, allowing users to access information without typing, which is especially beneficial for mobile users. Speech command recognition, commonly used on tablets and smartphones, is particularly important for interacting with devices while driving or controlling smart home components in living spaces.

Among various techniques, **statistical Hidden Markov Models (HMMs)** have been widely used for recognizing voiced keywords. In this approach, an HMM is created for each keyword to be recognized, and a decoding algorithm is applied to find the maximum likelihood match of the input to the trained models. However, this method

Machine learning involves developing general algorithms that can extract meaningful patterns or insights from a dataset without the need to write problem-specific code. Instead of programming explicit rules, data is fed into a generic algorithm, which then constructs its logic based on the underlying patterns in the dataset. Machine learning has proven to be a powerful approach for **classification problems**.

requires careful initialization and efficient training for each word, which can be computationally intensive. More recent research has introduced advanced approaches, such as **large-margin methods** and **recurrent neural networks (RNNs)**, to improve performance and reduce computational overhead.

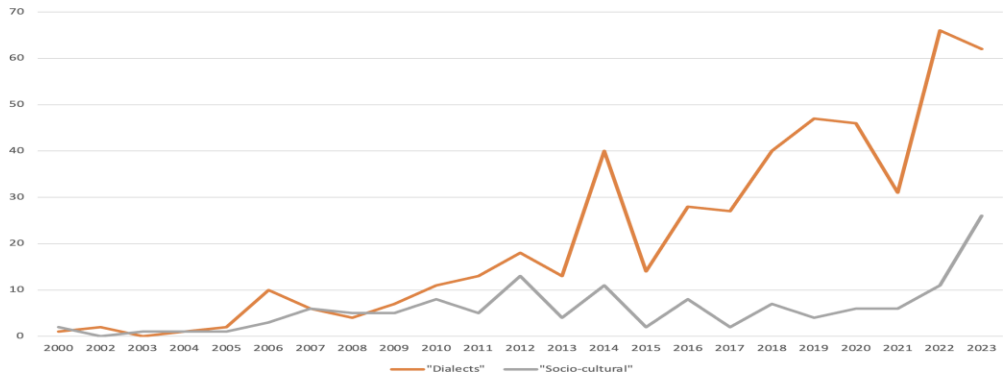

Figure 3: Number of relevant 'papers-per-year' for keywords 'dialects' and 'socio-cultural' in the ACL Anthology.

The **Deep Neural Network (DNN)** technique applied by Google for keyword recognition has demonstrated better performance compared to traditional statistical HMM systems. DNNs are also simpler to implement and require comparatively lower computational resources. When considering variants with both smaller and larger vocabulary sizes, **Convolutional Neural Networks (CNNs)** outperform DNNs as well as fully connected single-layer neural network models. The superior performance of CNNs in keyword recognition can be attributed to several factors: CNNs require fewer parameters than DNNs, reducing both memory usage and computational cost, and their ability to capture local patterns in the spectrogram representation of audio signals correlations in frequency and time allows for more effective feature extraction. Consequently, CNNs provide both enhanced performance and a more compact model size compared to DNNs, making them a contemporary and preferred approach for keyword recognition tasks.

Table 3: Average sentence accuracy rate

Approach	**Average sentence Accuracy rate**
Isolated	70.7%

Unigram	80.5%
Bi-gram	82.8%
Tri-gram	86.6%

As voice command recognition is commonly applied on smartphones, tablets, and other portable devices, it is essential that the system operates with **low latency, small memory footprint, and minimal computational requirements** [15]. Motivated by this, the present work aims to develop a **context-specific keyword recognition system** capable of accurately identifying predefined keywords and enabling machines to respond effectively to user commands. The keywords selected for this study correspond to **ten important human body parts**, spoken in the **Odia language**.

The dataset is modeled similarly to the one used by Google's Tensor Flow and AIY teams for keyword spotting, comprising **over 10,000 .wav audio files** of different speakers pronouncing these ten keywords. Each audio file is one second long and contains a single keyword. The proposed system classifies each one-second Odia voice signal according to its label. Both a **fully connected DNN model** and a **proposed CNN model** are employed to compute the probability that the input signal corresponds to each label, ultimately selecting the label with the highest likelihood.

The audio recordings were collected from a diverse group of speakers, primarily students and faculty aged **18 to 60 years**, with each speaker producing individual keywords rather than full sentences. Each speaker was asked to pronounce the ten keywords, typically repeating each keyword **five times**, resulting in a total of **10,335 audio files**. Recordings were captured using software such as **Audacity** or mobile recording apps, and then converted to **16-bit little-endian PCM WAV files** with a **16 kHz sampling rate**. Preprocessing included noise reduction and the removal of continuous silent segments using Audacity. Audio files were organized into folders based on keyword content, creating a dataset suitable for training machine learning models.

For feature extraction, a **reference pitch of 1000 Hz** with an audible threshold of **40 dB** was set. The audio signals were processed

using a **filter bank**, and **Mel Frequency Cepstral Coefficients (MFCCs)** were computed. The MFCCs employ a **discrete cosine transform (DCT)** to capture the real logarithm of short-term energy reflected on the Mel frequency scale, providing robust acoustic features for keyword recognition.

Table 4: Comparison of recognition rates (%) with our continuous voiced Odia digit recognition system

Reference	**Feature Extraction**	**Feature classification Technique**	**Recognition Rate (%)**
R. Thanga rajan et.al	MFCC	HMM	Speaker Dependent:88.82 Speaker Independent: 92.06
M.Z Bhotto et.al	MFCC	VQ	70-85
S. Mohammad et.al	MFCC	VQ	88.88
S.M. Ahadi et.al	MFCC (noisy) MFCC (noiseless)	HMM (noisy) HMM (noiseless)	28 -78 86
Proposed system (CVODR)	MFCC	HMM	Isolated:70.7

			Unigram:80.5 Bigram:82.5 Trigram:86.6

The **MFCC (Mel Frequency Cepstral Coefficient)** technique has been widely applied across various signal analysis tasks and is considered highly effective compared to other feature extraction methods. In this work, MFCCs are used to extract speech parameters from each audio file. Based on human auditory perception, the Mel-frequency scale transforms the frequency dimension to provide a more perceptually relevant representation of the voice signal.

To prepare the data for training a **Convolutional Neural Network (CNN)**, the audio signals are first converted into spectrograms, which are then saved as corresponding **Num Py arrays**. The files are subsequently summarized into a CSV file that maps each spectrogram to its respective class. Using the **Fourier Transform**, a voice signal is decomposed into its constituent frequency components, and the spectrogram captures the intensity of each frequency over time. All audio signals are sampled at **16,000 samples per second**, effectively capturing frequencies up to **8,000 Hz**. For example, the keyword "□□□□□" (head) is represented in this manner.

MFCCs are computed as additional transformations on the spectrogram, capturing critical aspects of the human voice. For each input frame of a signal, MFCC calculates the **cepstral coefficients**, their first-order derivatives (**delta coefficients**), and second-order derivatives (**delta-delta coefficients**), along with variations in the power spectrum. This approach provides a robust representation widely used in standard speech recognition systems.

The resulting spectrograms are represented as **NumPy arrays with a shape of 28×28 (time × frequency)**, where each element encodes the strength of a particular frequency at a given time. Treating these arrays as images allows the application of

CNN models, which are well-suited for image-like data, to effectively perform keyword recognition on the audio signals.

Methods for Recognition of Voiced Odia Keywords

In this work, two deep learning methods are implemented to recognize voiced Odia keywords. These are the convolutional neural network (CNN) model and deep neural network (DNN) model.

Table 5: Big ram counts for a mobile number consisting of 10 digits

□□	□□□	□□□□	□□	□□	□□□□□	□□□□□	□□	□□□	□□□
6	10	10	8	6	6	6	20	16	10
0	0	6	2	0	4	4	4	16	0
6	6	12	6	6	12	12	2	4	6
30	12	12	8	30	8	8	10	14	12
6	10	10	8	6	6	6	20	16	10
12	0	4	6	12	12	12	10	2	0
12	0	4	6	12	12	12	10	2	0
8	6	10	2	8	12	12	12	4	6
16	10	6	18	16	6	6	14	8	10
0	0	6	2	0	4	4	4	16	0

Convolutional Neural Network Model

Convolutional Neural Networks (CNNs) are among the most popular deep learning architectures and are extensively used in **speech-to-text (STT)** systems. The key components of CNNs include **weight sharing, convolutional filters, and pooling layers**. These components collectively contribute to the remarkable performance improvements observed in STT applications.

A general block diagram of a CNN model begins with **preprocessing, feature extraction, and spectrogram representation** of each audio signal,

followed by the generation of **NumPy arrays**. The data is then passed through a sequence of convolutional layers, each consisting of three main components: **convolution, pooling, and nonlinearity**. The convolutional layers extract local patterns in the data, while pooling reduces the dimensionality of the feature maps, retaining the most important information and discarding less significant data. Nonlinear activation functions are applied to introduce nonlinearity, allowing the network to model complex relationships.

After the convolutional layers, the output is passed through a **fully connected layer**, often including dropout for regularization, and finally through a **soft max layer** that maps the network outputs to the desired number of classes. Modern CNN architectures, combined with improved training techniques, achieve high performance comparable to human-level recognition. In particular, **pooling** is a crucial operation in CNNs, reducing the size of the parameter maps while preserving essential features. Max pooling, one of the most commonly used techniques, has been shown to improve performance in **automatic speech recognition (ASR)** tasks by emphasizing salient features and suppressing irrelevant information.

Conclusion

In conclusion, analyzing dialectal variations in Odia through computational approaches to native language identification reveals both the richness and complexity of the language. The study highlights the challenges posed by regional dialectal differences at the lexical, phonetic, and syntactic levels, which make accurate dialect identification a difficult task. However, machine learning models, when applied with sufficient and diverse training data, show promising results in distinguishing these dialects. While the overlap between dialects and the lack of comprehensive datasets remain significant hurdles, the use of computational techniques such as feature extraction, decision trees, and neural networks offers a path forward. The research emphasizes the importance of considering contextual and sociolinguistic factors, and suggests that continued advancements in machine learning, along with the development of richer datasets, can improve accuracy in dialect identification. Ultimately, these computational approaches hold great potential for enhancing language technologies and preserving linguistic

diversity in Odia, benefiting applications like speech recognition and machine translation.

Reference

1. Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A. A., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., et al. (2024). Phi-3 technical report: A highly capable language model locally on your phone. arXiv preprint arXiv:2404.14219.

2. Acharya, D., Dawadi, S., Saud, S., &Regmi, S. (2025). Paramananda@nlu of devanagari script languages 2025: Detection of language, hate speech, and targets using fasttext and BERT. In Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL), Abu Dhabi. International Committee on Computational Linguistics (ICCL).

3. Mistral AI & NVIDIA. (2024). Title of webpage. https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407. Accessed: 2024.

4. Rahman Aodhora, S., Ahsan, S., & Hoque, M. M. (2025). Cuet_hateshield@nlu of devanagari script languages 2025: Transformer-based hate speech detection in devanagari script languages. In Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL), Abu Dhabi. International Committee on Computational Linguistics (ICCL).

5. Aralikatte, R., De Lhoneux, M., Kunchukuttan, A., & Søgaard, A. (2021). Itihasa: A large-scale corpus for Sanskrit to English translation. In Proceedings of the 8th Workshop on Asian Translation (WAT2021), 191–197.

6. Barbieri, F., Camacho-Collados, J., Neves, L., & Espinosa-Anke, L. (2020). Tweeteval: Unified benchmark and comparative evaluation for tweet classification. arXiv preprint arXiv:2010.12421.

7. Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Pardo, F. M. R., Rosso, P., & Sanguinetti, M. (2019). Semeval-2019 task 5:

Multilingual detection of hate speech against immigrants and women in Twitter. In Proceedings of the 13th international workshop on semantic evaluation, 54–63.

8. Chakraborty, D., Hossain, J., &Hoque, M. M. (2025). One_by_zero@nlu of devanagari script languages 2025: Target identification for hate speech leveraging transformer-based approach. In Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL), Abu Dhabi. International Committee on Computational Linguistics (ICCL).

9. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., &Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 8440–8451, Online.

10. Devlin, J., Chang, M.-W., Lee, K., &Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 4171–4186, Minneapolis, Minnesota.

11. Doddapaneni, S., Aralikatte, R., Ramesh, G., Goyal, S., Khapra, M. M., Kunchukuttan, A., & Kumar, P. (2022). Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. arXiv preprint arXiv:2212.05409.

12. Doddapaneni, S., Aralikatte, R., Ramesh, G., Goyal, S., Khapra, M. M., Kunchukuttan, A., & Kumar, P. (2023). Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 12402–12426, Toronto, Canada.

Association for Computational Linguistics.

13. Gupta, S., &Arora, B. (2022). Stemming techniques on English language and Devanagari script: A review. Recent Innovations in Computing: Proceedings of ICRIC 2021, Volume 1, 541–550.

14. Gupta, S., Singhal, S., &Wasi, A. T. (2025). Iitr-ciol@nlu of devanagari script languages 2025: Multilingual hate speech detection and target identification in devanagari-scripted languages. In Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL), Abu Dhabi. International Committee on Computational Linguistics (ICCL).

15. Gupta, V., Roychowdhury, S., Das, M., Banerjee, S., Saha, P., Mathew, B., Mukherjee, A., et al. (2022). Multilingual abusive comment detection at scale for Indic languages. Advances in Neural Information Processing Systems, 35, 26176–26191.

16. Guragain, A., Poudel, N., Piryani, R., &Khanal, B. (2025). Nlpineers@ nlu of devanagari script languages 2025: Hate speech detection using ensembling of BERT-based models. In Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL), Abu Dhabi. International Committee on Computational Linguistics (ICCL).

17. Hong, J., Lee, N., & Thorne, J. (2024). Reference-free monolithic preference optimization with odds ratio. arXiv e-prints, arXiv:2403.

18. Hossan, M. R., Sakib, N., Miah, M. A., Hossain, J., &Hoque, M. M. (2025). Cuet_big_o@nlu of devanagari script languages 2025: Identifying script language and detecting hate speech using deep learning and transformer model. In Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL), Abu Dhabi. International Committee on Computational Linguistics (ICCL).

19. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). Lora: Low-rank adaptation of large

language models. arXiv preprint arXiv:2106.09685.

20. Ibrahim, M. (2025). Cufe@nlu of devanagari script languages 2025: Language identification using fasttext. In Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL), Abu Dhabi. International Committee on Computational Linguistics (ICCL).