04-09-2025

Statewise top 3 languages?

Rule-based classifier between bodo and nepali. (using morphology analysis)

Top 5 language, geographic location and improve the lid performance.

Python lib lang id.

Infer all the samples with langid model.

English or non-english

Test on all data samples.

Dogri - punjabi

Bodo

classify bodo and nepali language

Fallback Ideas

## 1 Language Detection with Location Fallback

Process:

* Primary Detection: Call IndicLID API to get top 5 language predictions
* Ground Truth Check: Compare with expected language
* Fallback Strategy (when mismatch occurs):  
  + Filter by scheduled languages for user's state/location
  + Apply word overlap scoring using common word dictionaries
  + Select best candidate based on combined confidence and word evidence

Key Features:

* Geographic filtering using Constitutional language schedules
* Dictionary-based linguistic validation
* Multi-tier confidence scoring system
* Comprehensive tracking of fallback usage

## 2️⃣ Transliteration with Confidence Selection

Dual Method Approach:

* IndicXlit API: Professional transliteration service
* Rule-based: Enhanced transliteration using linguistic rules
* Confidence Calculation: Character overlap, length ratios, word matching
* Best Selection: Choose method with highest confidence score

## 3️⃣ Quality Evaluation

Metrics Used:

* BLEU Score: Standard translation quality metric
* Word Error Rate (WER): Word-level accuracy
* Character Error Rate (CER): Character-level precision

Evaluation Stages:

* Compare transliteration output with native script reference
* Evaluate final English translation against reference translation

## 4️⃣ Translation Pipeline

Process:

* Take best transliterated output from Step 2
* Use IndicTrans2 API for translation to English
* Evaluate translation quality using same metrics
* Track translation accuracy across different languages

## 5️⃣ Comprehensive Reporting

Excel Output Contains:

* Original data (Language, Roman Script, Native Script, English Translation)
* Detection results with confidence scores
* Fallback information and method used
* Transliteration outputs from both methods
* Quality evaluation scores (BLEU, WER, CER)
* Translation results and evaluation
* Statistical summaries by language and location

## 🔧 Key Fallback Strategies

## Location-Aware Filtering

* Uses your SCHEDULED\_LANGUAGES\_BY\_STATE mapping
* Filters API predictions to geographically valid languages
* Reduces false positives from unrelated language families

## Dictionary-Based Validation

* Your COMMON\_WORDS\_DICT provides linguistic evidence
* Calculates word overlap scores for candidate languages
* Heavily weights actual word evidence over model confidence

## Multi-Stage Decision Making

text

API Prediction → Ground Truth Check → Location Filtering → Word Validation → Final Decision

## Confidence-Based Selection

* Combines multiple signals: API confidence + word matches + location validity
* Uses weighted scoring: word\_matches × 30 + api\_confidence × 10
* Prioritizes linguistic evidence over statistical confidence

## 📊 Pipeline Strengths

## Robustness

* Multiple fallback layers prevent complete failures
* Location context reduces geographic mismatches
* Dual transliteration methods improve accuracy

## Transparency

* Detailed logging of decision process
* Tracks which fallback methods were used
* Provides confidence scores for all outputs

## Evaluation

* Comprehensive metrics (BLEU, WER, CER) at each stage
* Compares against both native and English references
* Language-wise and location-wise accuracy breakdowns

## Scalability

* Batch processing of entire Excel datasets
* Progress saving for large datasets
* Error handling for partial failures

## 🎯 Use Cases

This integrated pipeline is ideal for:

* Multilingual NLP Applications: Handling diverse Indian language inputs
* Translation Quality Assessment: Systematic evaluation of translation pipelines
* Geographic Language Processing: Location-aware language detection
* Research and Development: Comparative analysis of different methods

## 📈 Expected Improvements

By combining your location fallback strategy with the full pipeline, you can expect:

* Higher Detection Accuracy: Especially for geographically contextualized text
* Reduced False Positives: Location filtering eliminates impossible language combinations
* Better Transliteration Quality: Confidence-based selection between multiple methods
* Comprehensive Analysis: End-to-end evaluation from input to final translation

This creates a production-ready system that handles the complexities of Indian language processing while providing detailed metrics for continuous improvement and optimization.

1. <https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/102011964/bd89efee-f225-4a1d-abfb-7f563b422097/paste.txt>

Threshold to both xlit and rule-based.

Casual Roman to standardized roman. - (using mapping aksharamukha.) then rule-based transliteration or xlit to native .

* Combines multiple signals: API confidence + word matches + location validity

Classifier for languages - dogri taking default punjabi

Top 5 loop through dictionary, checking which detection matches with the language in the dictionary. Then if you have two languages then go through the common words dictionary.

Which classifier? Rule-based classifier

**Rule-based Classifier**

Unique suffixes, pronouns, interrogatives,copulas,particles,conjuctions, prepositions

Aksharamukha supported scripts

**North Indian (Indo-Aryan) scripts**

* Devanagari (Hindi, Marathi, Sanskrit, Nepali, Konkani, etc.)
* Bengali–Assamese (Bengali, Assamese, Bishnupriya Manipuri)
* Gujarati
* Gurmukhi (Punjabi)
* Oriya/Odia

**South Indian (Dravidian) scripts**

* Tamil
* Telugu
* Kannada
* Malayalam

**Other Indic scripts**

* Sinhala

**ITRANS (Indian languages TRANSliteration)**

Indic scripts (Devanagari and other Indian scripts) into Roman (Latin) script using only plain English keyboard characters.

Uses both uppercase and lowercase letters, and sometimes character sequences, to represent Indian language sounds.

For example, ‘aa’ for long ‘a’, ‘sh’ for ‘श’, ‘th’ for ‘थ’.

* kh → ख (aspirated k)
* ch → च (palatal ch)
* th → थ (aspirated t)
* bh → भ (aspirated b)
* sh → श (palatal s)
* zh → ळ (retroflex l in some languages)

| aa | A | आ | Long 'a' |
| --- | --- | --- | --- |
| ee | I | ई | Long 'i' |
| oo | U | ऊ | Long 'u' |
| ou | au | औ | Diphthong 'au' |
| ai | ai | ऐ | Diphthong 'ai' |
| ei | ai | ऐ | Also maps to 'ai' |

1. **INDICLID for Language Identification (Rule-based Classifiers + Location Fallback)**

The misclassifications are found in some languages.

**Languages**

Bodo - Nepali, Bodo - non-bodo

Punjabi/Dogri - Non-Punjabi/Dogri

English - Non English

Hindi- Non-Hindi

Kashmiri and Sindhi

Manipuri

Assamese

**2. Increasing threshold**

TP above 0.99

Not assamese FP 0.87

Assamese

**Combination of two or more languages**

Based on the confidence score, we can pass those language with the highest confidence score

1. **Xlit and Rule-based Transliteration**

Problems in selecting the best translation output using confidence score only.

1. Input: two strings (original and transliterated).  
   Output: a float between 0.0 and 1.0 (confidence score).
2. If either string is empty, or if they are exactly the same → return 0.0.

(This assumes identical input/output means "no meaningful transliteration happened".)

1. Convert both to lowercase, remove spaces, and take unique characters.

Example: "Namaste" → {n, a, m, s, t, e}.

1. If original has no characters, score = 0.
2. Measures how many characters match between original and transliterated.

Range: 0.0 (no overlap) → 1.0 (perfect overlap).

1. Compares string lengths.

If lengths are similar → ratio close to 1.

If one is much shorter/longer → closer to 0.

1. Splits by spaces and compares number of words.
2. Ensures transliteration hasn’t merged/split words too much.  
   1.0 if same number of words, lower if mismatched.

* Weighted average of three factors:  
  + 40% → character overlap
  + 30% → length similarity
  + 30% → word count similarity

This weighting prioritizes character similarity but still accounts for structure.

Trywithout rule-based transliteration?

Done.

Try?

Aksharamukha to convert raw roman script to standardised roman script

**Aksharamukha** is used as a **pre-processor**:

* Convert **casual Roman input → standardized Roman**.
* Example: “namastey” → “namaste”.
* After that, feed into **IndicXlit** for final script transliteration.

Implemented

1. **Raw Roman → Aksharamukha → Standard Roman → Detection → Dual Transliteration → Dual Translation → Evaluation**
2. **Raw Roman → Detection → Aksharamukha → Standard Roman →Dual Transliteration → Dual Translation → Evaluation**

Check whether there are any problems with the implementation of Aksharamukha?

TryAksharamukha to convert Roman text into Native Indic scripts?

Try Combine xlit with aksharamukha?

Daily Updates:Language Detection………………………………………………………………………..kllllll

H nbvfsxzA

India’s agricultural landscape is linguistically diverse, and farmers naturally prefer to ask questions in their own language and script. To make ACE inclusive and accessible, the system incorporates a **multi-stage language pipeline** that detects, translates, and delivers responses seamlessly across more than 20 Indian languages.  
**Step 1: Language Detection**The first step is identifying the language and script of the incoming query. This is handled by *IndicLID*, a specialized language identification tool that supports both native-script and romanized forms of Indian languages. It can distinguish between 22 constitutionally recognized languages, even in cases where romanized text creates ambiguity (e.g., “kheti” could belong to Hindi, Marathi, or Punjabi). The system also uses a weighted linguistic feature scoring mechanism for language detection. Each feature earns points proportional to its length and discriminative power: particles (2–3×), suffixes (1–4×), pronouns (5×), interrogatives (5–8×), copulas (5–6×), conjunctions (3×), and postpositions (4–5×). A threshold-based framework classifies results: ≥30 = high confidence, 15–29 = medium, 10–14 = low, and <10 = negative. This ensures morphologically rich, unique features drive accurate detection, while the thresholds provide clear confidence levels for multilingual NLP pipelines.  
**Step 2: Transliteration (if needed)**If a farmer types in Roman script (e.g., “urad ke liye dawa”), the system transliterates the text into the appropriate native script using *IndicXlit* or a rule-based transliteration engine. This ensures the text is properly interpreted before translation.  
**Step 3: Translation to English**Once the script is standardized, the text is translated into English using *IndicTrans2*, a transformer-based multilingual model. English serves as the working language for the language models and knowledge base, ensuring consistency across all queries.  
**Step 4: Processing the Query**The translated query is passed through the ACE engine (Buckets A, B, C, or cache/scaffolding) to generate an appropriate answer.  
**Step 5: Translation Back to Farmer’s Language**The English answer is translated back into the farmer’s original language using *IndicTrans2*. If the farmer originally typed in Roman script, the response is also transliterated back using *IndicXlit* or rule-based methods to match their preferred input style.  
**Step 6: Delivery in Preferred Mode**The final response is delivered in text and, where needed, converted to speech through tools like IndicTTS, making the system usable even for farmers with limited literacy.  
**Error Handling and Safeguards**To reduce misclassification, the pipeline uses confidence thresholds, fallback rules, and lexicon checks. For sensitive cases where the language cannot be determined with high certainty, the system either requests clarification or defaults to the most likely option based on location data. As an additional safeguard, an LLM-based fallback is employed to handle ambiguous queries and provide context-aware responses. By combining **language detection, transliteration, translation, and text-to-speech**, ACE ensures that farmers can interact in whichever language or script feels most natural to them. This inclusivity is central to its role as a Digital Public Good for all of India’s farmers.