Language Modeling for Romanized Telugu WhatsApp Text

Deekshith Rathod, Vishnu Hemanth | TDL Project · April 2026

Agenda

What is Romanized Telugu?

- Telugu (83M speakers) is a Dravidian language written in its own script.
- In WhatsApp chats, users write Telugu phonetically in Latin script.
- The result freely mixes Telugu, English, and Hindi at the word level.
- Tens of millions use this register — nearly absent from NLP corpora.
- Standard autocomplete & LMs perform poorly on it.

Example Messages

"nenu ikkadiki vastunnanu bro, okka nimisham agu"

→ I'm coming there bro, wait a moment

"have some shame anna"

→ have some shame, brother (English + Telugu address term)

Problem Definition

Task: Given a prefix from a Romanized/Code-Mixed Telugu WhatsApp message, predict the most likely next word.

Why is this hard?

Challenge	Example
Tokenization mismatch	cheppukovadam → 6 GPT-2 tokens
Code-mixing	Single sentence switches 3 languages
Data scarcity	No benchmark; corpus built from scratch
Metric distortion	Top-1 on first consonant ≠ correct word

Research Questions

1. Does fine-tuning GPT-2 on this corpus improve next-word prediction?
2. Does a domain-trained tokenizer further improve performance?

Dataset — 298 WhatsApp Archives

325,478	296,903	373 MB	80/10/10
Raw Messages	After Filtering (91.2% kept)	Corpus Size	Train/Val/Test Split

Cleaning Pipeline

Split Details

Split	Sentences	Purpose
Train	237,522	Model training
Validation	29,690	Hyperparameter tuning
Test	29,691	Final evaluation (100 examples)

Proposed Solution — 4-Model Comparison

Isolating contributions of multilingual pretraining vs. fine-tuning vs. tokenizer design

GPT-2 Pretrained	mGPT Pretrained	GPT-2 Fine-tuned (orig-tok)	GPT-2 Fine-tuned (custom-tok)
Tokenizer	**Tokenizer**	**Tokenizer**	**Tokenizer**
GPT-2 50k BPE	mGPT 60-lang	GPT-2 50k BPE	Custom 12k BPE
Training	**Training**	**Training**	**Training**
None	None	LoRA on corpus	LoRA on corpus
Tests	**Tests**	**Tests**	**Tests**
English baseline	Multilingual baseline	Does fine-tuning help?	Does custom tokenizer help?

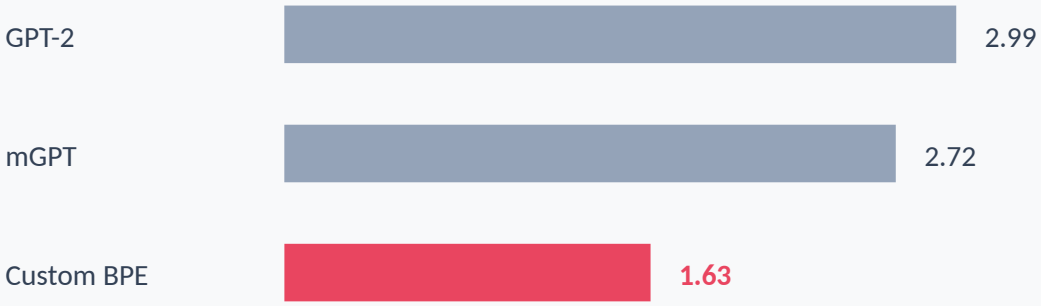
Tokenizer — The Fertility Problem

GPT-2 was built for English. Telugu words get shredded into useless sub-pieces.

Token Count Comparison

Word / Phrase	GPT-2 Tokens	Custom BPE Tokens
cheppukovadam	6	2
meeru ela unnaru	6	3
vivarinchandi	6	1–2

Fertility (Avg Tokens / Word)



45% lower fertility than GPT-2

Custom BPE Details

Vocabulary size

12,000 tokens

Algorithm

Byte-Level BPE

Min frequency

2

Training data

296,903 sentences

Fine-Tuning with LoRA

Why LoRA?

Full fine-tuning of 124M parameters is expensive. LoRA freezes the model and injects small trainable rank-decomposition matrices into attention layers — enabling efficient domain adaptation.

LoRA Config

Rank r: 8

Target layers: c_attn (attention)

Trainable params: 294,912 (0.24% of GPT-2)

Training: 50k sentences · 4 epochs · T4 GPU · ~20 min

Training Curve (orig-tok)

Epoch	Val Loss	Val PPL
1	4.72	112.6
2	4.56	95.8
4	4.46 ✓	86.2 ✓

Custom Tokenizer Problem

Randomly initialized embeddings never learned coherent representations. Val PPL stayed at ~3,200 across all epochs — mode collapse to corpus unigram.

Fix: Initialize from subword-averaged GPT-2 embeddings before LoRA training.

Results — The Full Picture

All models evaluated on the same 100 held-out next-word prediction examples

Model	Test PPL	Top-1	Top-3	Top-5
GPT-2 pretrained	362.9	2%	5%	6%
mGPT pretrained ★	**213.3**	**15%**	**22%**	**25%**
GPT-2 fine-tuned (orig-tok)	92.3	4%	8%	13%
GPT-2 fine-tuned (custom-tok)	3,624 †	1%	1%	1%

Key Finding

mGPT with zero fine-tuning beats everything we trained — by 7.5× on top-1 accuracy. Multilingual pretraining > domain adaptation for code-mixed South Asian text.

† Not directly comparable — different vocabulary size

What Does Fine-Tuning Actually Do?

Perplexity ✓

PPL drops dramatically:
362.9 → 92.3 (4× improvement)

The model clearly learned the subword distribution of Romanized Telugu.

Accuracy ✗

Top-1 barely moves: 2% → 4%

Fine-tuned top predictions are dominated by Telugu consonant stubs:

Prediction	Count / 100
v	16
k	11
mar	4

The model predicts stubs (v, k, mar…) and gets credit only when the gold word starts with that stub. It's measuring stub prediction, not word prediction.

The Code-Switching Wall

All models fail when the prompt is English but the correct answer is Telugu:

Prefix: *"have some shame"* → Correct next word: **anna** *(Telugu: "brother")*

Model	Prediction	Correct?
GPT-2 pretrained	in	✗
mGPT pretrained	on	✗
GPT-2 fine-tuned	about	✗
GPT-2 custom-tok	mariyu	✗

Why this matters

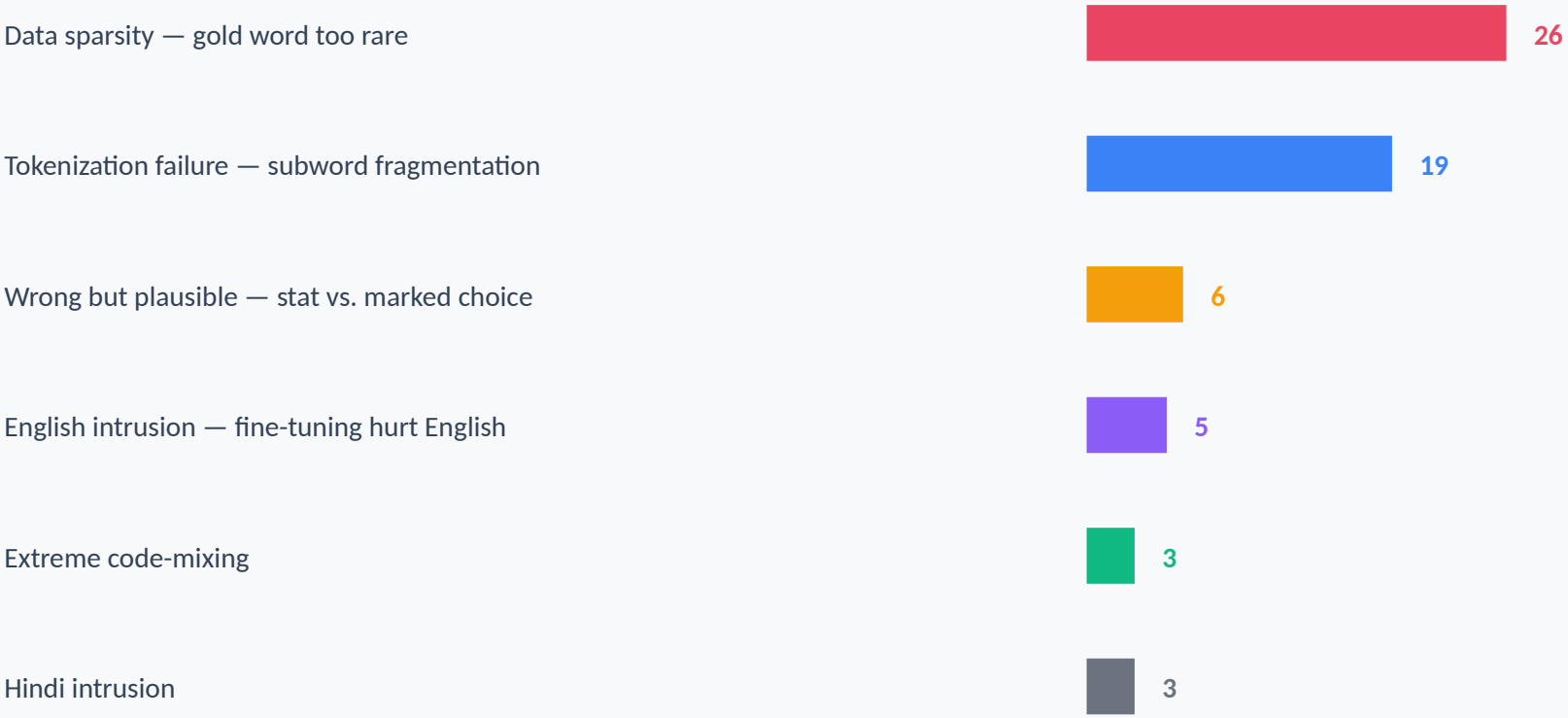
Every model sees 'have some shame' as English and continues in English.

The Telugu social address marker anna is invisible to any model without explicit code-switch training data.

This pattern (English reprimand + Telugu address) is structurally common — and structurally unlearnable from our setup.

Error Analysis

From 30 examples where ≥ 3 of 4 models fail top-1 prediction:



Data sparsity + tokenization account for 45 of 58 recorded failure instances — structural issues, not fixable by more training on the same setup.

Key Takeaways

1 **Multilingual pretraining > domain fine-tuning**

mGPT (no fine-tuning): 15% top-1 vs GPT-2 fine-tuned: 4% top-1

2 **Fine-tuning helps perplexity, not accuracy**

GPT-2's tokenizer is structurally wrong for this language

3 **LoRA alone cannot fix a vocabulary transplant**

Embedding retraining is required alongside LoRA for custom tokenizer fine-tuning

4 **Top-k accuracy is a misleading metric here**

Whole-word match needed — first-subtoken hits inflate scores

5 **Code-switching is an unsolved structural problem**

No model can cross a language boundary without bilingual pretraining

Future Work

P1	Fine-tune mGPT (not GPT-2) on our 296k corpus	High — right initialization
P2	Fix embedding init — subword-average GPT-2 embeddings into custom vocab slots	High — unblocks custom tokenizer
P3	Change metric — whole-word top-k instead of first-subtoken	Medium — fairer evaluation
P4	Stratified eval set — separate English-only, Telugu-only, code-mixed prefixes	Medium — clearer diagnostics
P5	Larger corpus — more WhatsApp archives to reduce data sparsity failures	High but slow

Thank You

Repository: github.com/Deekshithrathod/TDL-Project

Summary

- → 296,903 WhatsApp sentences collected and cleaned
- → Custom 12k BPE tokenizer — 45% lower fertility than GPT-2
- → LoRA fine-tuning: 4× perplexity reduction in 20 min on a free GPU
- → Main finding: Use mGPT, not GPT-2, as the starting point for code-mixed South Asian NLP

Questions?