Malayalam GPT: Building a LLM from Scratch

Tokenizer, Architecture, and 8GB Corpus Training

Joel K George Aquib Mohammed Shahma Fathima Ashik MA

Department of Computer Engineering Business Systems
Model Engineering College, Thrikkakara

April 2025

Overview

Introduction

Introduction

- Malayalam is one of the least represented languages in NLP despite its rich linguistic structure and large speaker base.
- Existing multilingual models are not optimized for Malayalam-specific scripts and grammar.
- Our project aims to build a large language model (LLM) from scratch tailored for Malayalam, covering the full pipeline — from tokenization to deployment.
- The goal is to improve accessibility, content generation, and digital inclusivity for Malayalam speakers.

Literature Review

Literature Review

Year	Methodology	Disadvantage
2020	GPT-2 based multilingual models like mBERT and XLM-R used for low-resource languages including Malayalam.	Generic tokenization causes poor performance on complex Malayalam morphology.
2021	Fine-tuning large pre-trained models (e.g., GPT-J, mT5) with Malayalam datasets.	Limited dataset size leads to underfitting; model struggles with grammar and script-specific patterns.
2022	Byte-level tokenization with multilingual corpora including Indic languages.	Inefficient for Malayalam due to long sequence lengths and suboptimal context understanding.

Literature Review

Year	Methodology	Disadvantage
2023	Use of transliterated data and Romanized Malayalam for NLP tasks.	Loses semantic fidelity, cannot represent native script effectively.
2023	Character-level models and syllable-based embeddings designed for morphologically rich Indian languages.	Training is computationally expensive and lacks large-scale datasets.
2024	MalayaLLM and IndicGPT introduced Malayalam support through fine-tuning.	Still dependent on multilingual base models, lacks Malayalam-first architecture and tokenizer.

Project Proposal

Problem Statement and Proposed Solution

Problem Statement

- Malayalam is underrepresented in NLP.
- Existing multilingual models are not optimized for script and grammar.
- There is a need for a native, from-scratch LLM.

Proposed Solution

- Develop a transformer-based Malayalam LLM from scratch.
- Design a custom tokenizer to handle chillu characters, diacritics, and ligatures.
- Train on a large, curated 8GB Malayalam corpus covering diverse domains.
- Deploy using lightweight, scalable inference infrastructure.

Key Features

- **Custom Tokenizer for Malayalam:** Handles chillu characters, ligatures, and vowel signs for accurate, efficient tokenization.
- **Transformer-based Model Architecture:** Decoder-only GPT model with positional encoding, multi-head attention, and layer normalization.
- **Open-Domain Corpus Coverage:** Trained on an 8GB Malayalam corpus including news, Wikipedia, books, and forums.

Approach

- **Modular Training Pipeline:** Built using PyTorch and DeepSpeed for efficient data loading, optimization, and checkpointing.
- **Inference Interface:** Provides a user-friendly web UI for prompt-based Malayalam text generation across various tasks.
- **Feedback Loop for Fine-tuning:** Enables users or evaluators to flag outputs for quality control and further domain adaptation (e.g., education, healthcare).

Benefits and Use-Cases

Benefits

- **Native Language Support:** Produces natural and grammatically accurate Malayalam output.
- **Complex Script Handling:** Tokenizer supports chillu characters, diacritics, and conjuncts.
- **Fast Local Inference:** No need for cloud models like GPT-3 or Bard.
- **Customizability:** Easily fine-tuned for specific dialects and domains.
- **Educational/Research Use:** Serves as a foundation for regional AI and NLP research.

Use-Cases

- **Malayalam Chatbots:** For tourism, customer support, and local services.
- **Content Generation:** Automates writing of blogs, news, and educational materials.
- **Language Learning Tools:** Helps students improve grammar and vocabulary.
- **Summarization & Translation:** Context-preserving document processing.
- **Public Services:** Native-language government and civic interfaces.

System Architecture

System Architecture

1. Tokenizer Module

- Custom Malayalam tokenizer.
- Handles chillu characters, conjuncts, and diacritics.
- Uses syllable-aware or subword-level encoding.
- Reduces sequence length while preserving meaning.

2. Preprocessing & Corpus Builder

- Aggregates 8GB data from web, books, news, and Wikipedia.
- Cleans, normalizes, deduplicates, and segments text.
- Balances domains to avoid training bias.

3. Transformer Model Engine

- GPT-style decoder-only architecture.
- Multi-head self-attention and feedforward layers.

System Architecture

4. Training Pipeline

- Efficient batching, FP16 mixed precision.
- Gradient accumulation and checkpointing.
- DeepSpeed/ZeRO optimization for 1-GPU setups.

5. Inference & API Layer

- REST/CLI interface for generation, Q&A, summarization, translation.
- Accepts Malayalam prompts; returns native script output.
- Supports ONNX/quantized deployment for mobile and web.

6. Evaluation & Fine-tuning Module

- Metrics: BLEU, ROUGE, and perplexity.
- Human judgment for fluency and coherence.
- Domain-specific tuning (education, health, government).

Architecture Diagram

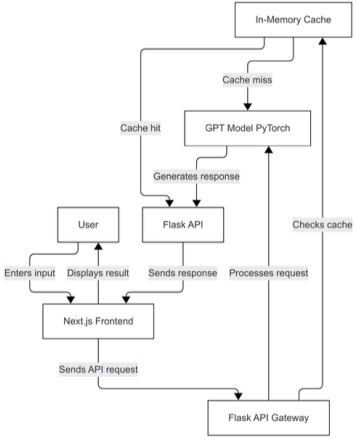

Figure 4.2: Malayalam GPT Training Pipeline

Figure: Malayalam GPT System Architecture

Design Layers

- Tokenizer Layer $\rightarrow$ Character-level, chillu-aware
- Model Layer $\rightarrow$ Transformer blocks, attention
- API Layer $\rightarrow$ Flask endpoints
- Frontend $\rightarrow$ HTML/JS, styled interface

Software Requirements

- Python, PyTorch, Flask, JavaScript
- GPU-enabled training environment (min 8GB VRAM)
- Corpus storage (8GB), ONNX Runtime

Software Requirement Specification

Functional Requirements

- **User Prompt Input and Response Generation:** Accepts Malayalam input and returns coherent, context-aware responses.
- **Malayalam Tokenization and Text Processing:** Handles chillu characters, ligatures, and native grammar with precision.
- **Training Pipeline Execution:** Supports data loading, cleaning, training, and validation at scale.
- **Model Inference and Text Generation:** Enables real-time or near-real-time generation for tasks like Q&A, summarization, etc.
- **Fine-tuning and Continuous Learning:** Future-ready architecture for domain-specific improvements.

Non-Functional Requirements

- **Accuracy:** Generates grammatically and semantically correct Malayalam content.
- **Performance:** Fast inference and optimized training with support for parallelism.
- **Privacy and Security:** Protects user data and prompt content when deployed online.
- **User Experience:** Malayalam-friendly interface with native script support and smooth UX.

Software Design Document

DFD Level 0 Diagram

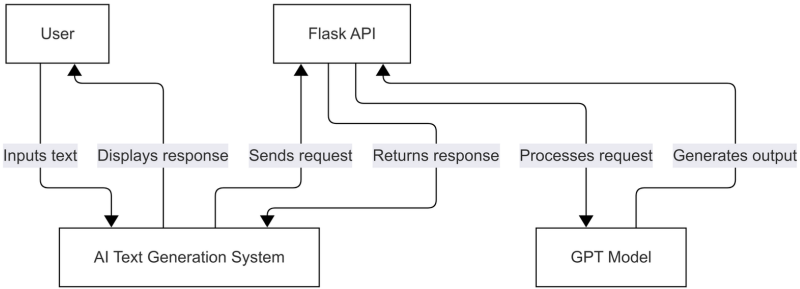

Figure 5.1: DFD Level 0

Figure: Context-Level Data Flow Diagram (Level 0)

DFD Level 1 Diagram

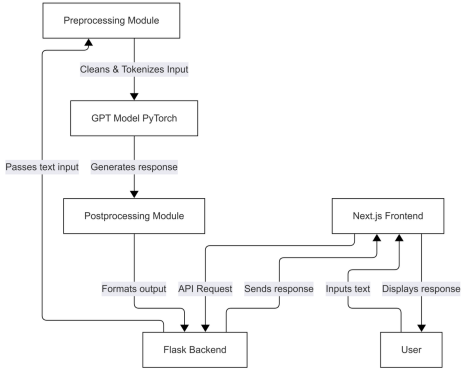

Figure 5.2: DFD Level 1

Figure: Detailed Process Flow Diagram (Level 1)

Screenshots

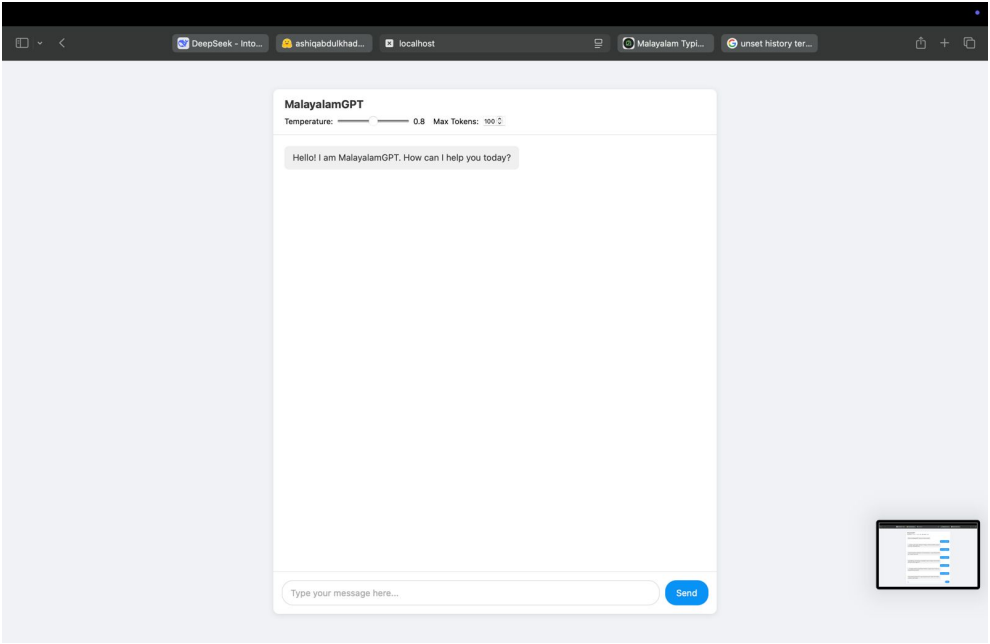

Figure: Malayalam GPT – Homepage

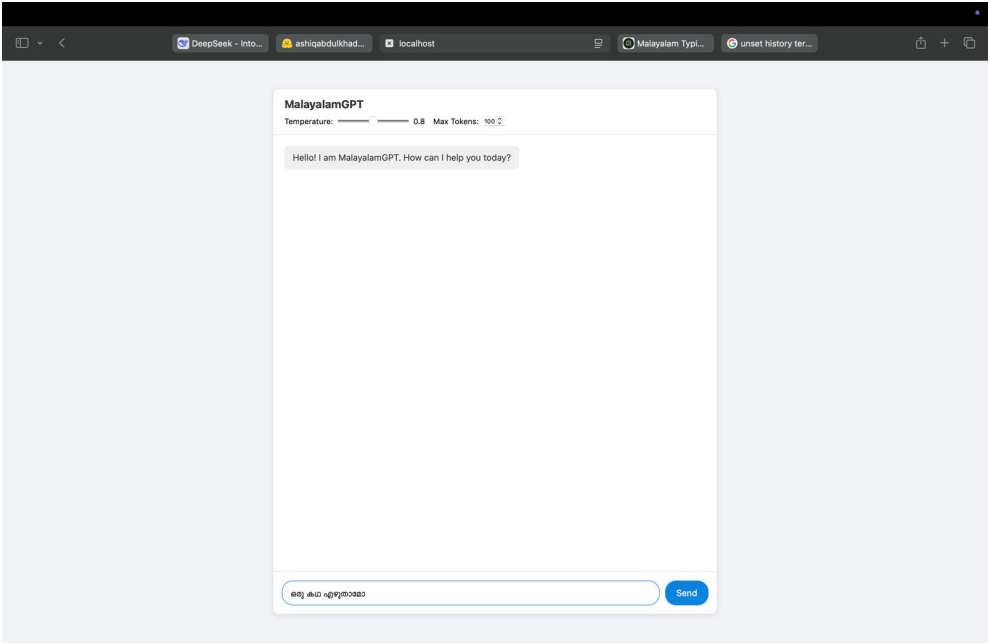

Figure: User Input Prompt for Text Generation

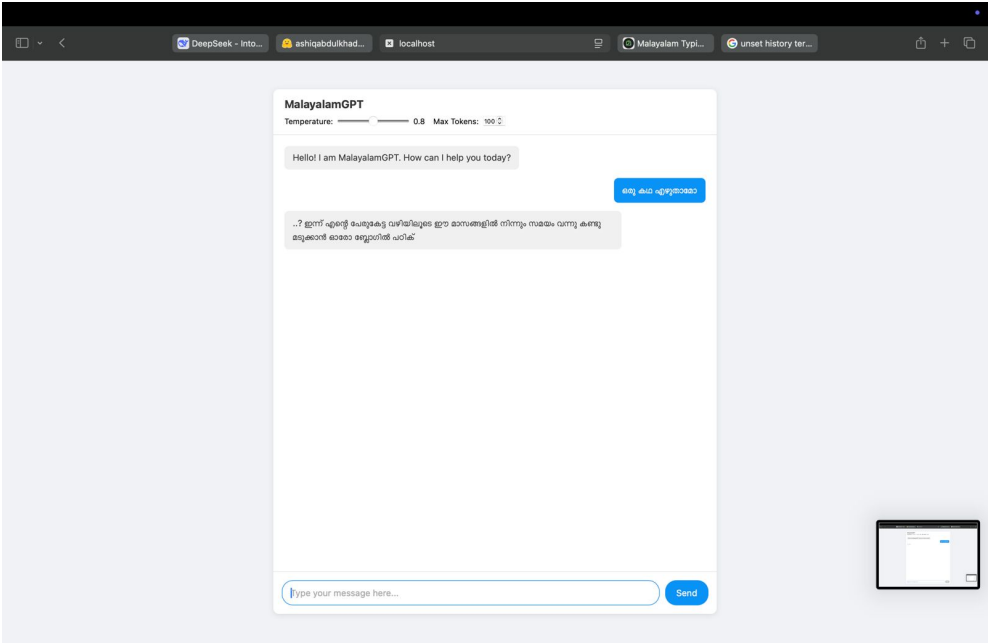

Figure: Generated Malayalam Text Output

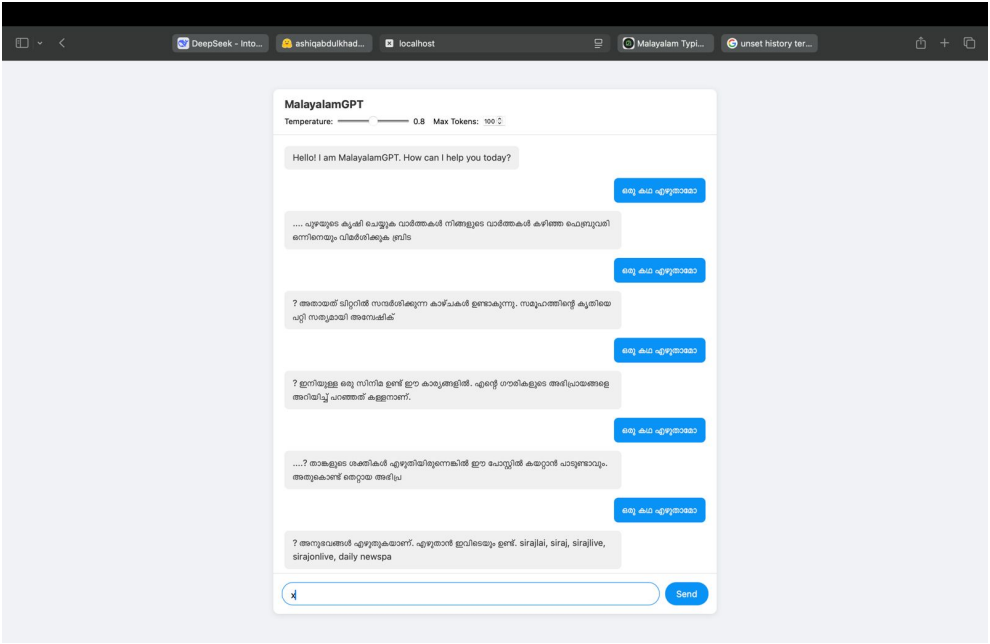

Figure: Admin Panel for Managing Logs and Sessions

Conclusion

Summary

- Built and deployed the first native Malayalam GPT model.
- Shows promising performance on grammar and fluency.
- Suitable for low-resource NLP development.

Future Scope

- **Multilingual Expansion:** Extend the model to support English–Malayalam translation and code-switching use-cases.
- **Speech Integration:** Add speech-to-text and text-to-speech capabilities for voice-based interactions.
- **Scalable Architecture:** Upgrade to larger transformer variants (e.g., 3B+ parameters) for enhanced performance.
- **Cross-Domain Adaptation:** Fine-tune models for specific sectors like healthcare, education, and legal systems.
- **Mobile Deployment:** Optimize inference with quantized/ONNX models for use in mobile and edge devices.

Bibliography

1. Radford et al., *Improving Language Understanding by Generative Pre-Training*
2. Brown et al., *Language Models are Few-Shot Learners*
3. *Generative Pre-trained Transformer: A Comprehensive Review...*
4. *A Morpheme Based Language Model for Malayalam Spoken Short Query Processing*
5. *Natural Language Inference for Malayalam Language Using Language Agnostic Sentence Representation*
6. *Neural Machine Translation System for English to Indian Language Translation Using MTIL Parallel Corpus*
7. Manohar et al., *Syllable Subword Tokens for Open Vocabulary Speech Recognition in Malayalam*
8. *MalayaLLM: A Continually LoRA PreTrained and FineTuned 7B Llama-2 Indic Model for Malayalam Language*

Thank You!