JSPM's
Rajarshi Shahu College of Engineering, Tathawade, Pune
(An Empower Autonomous Institute under Savitribai Phule Pune University)

Department of Computer Science and Business Systems
Artificial Intelligence Mini Project
AY-2025-26

Title of Project: **Katha-GPT: A Marathi Story Generation Small Language Model**

Student Name:

Vedant Mahesh Neve – RBT23CB024

Faculty Name:

Prof. Kavita Patil Ma'am

Introduction

Objective of the Project:

The primary objective of this project is to develop and train a generative small language model (~50M Parameters) capable of creating coherent and contextually relevant short stories in the Marathi language. The project aims to build a custom Generative Pre-trained Transformer (GPT) model from scratch using the PyTorch framework.

Scope:

The scope of this project encompasses the entire lifecycle of a language model development process, including:

- Data acquisition and preprocessing of a large Marathi text corpus.
- Design and implementation of a transformer-based neural network architecture.
- Training the model on a specialized dataset of Marathi stories and evaluating its performance using metrics like loss and perplexity.
- Developing a text generation module with various decoding strategies (e.g., top-k, top-p sampling) to produce creative and coherent text.

Mapping With SDG Goal:

This project maps to **Sustainable Development Goal 4 (SDG 4): Quality Education**. Specifically, it aligns with Target 4.7, which aims to ensure all learners acquire knowledge and skills needed to promote sustainable development, including the appreciation of cultural diversity and culture's contribution to sustainable development. By creating a tool that generates stories in Marathi, this project aids in the preservation and promotion of local language and culture, providing educational and creative resources in a regional language.

Key Features:

- **Custom GPT Model:** A transformer-based model built from the ground up.
- **Marathi Story Generation:** Capable of generating short, creative stories in Marathi based on a user-provided prompt.
- **Coherence Enhancement:** Implements techniques such as repetition penalty and n-gram blocking to improve the quality and readability of the generated text.
- **Flexible Training:** The system allows for training a new model from scratch or resuming training from a saved checkpoint.
- **Interactive Interface:** Provides an interactive command-line interface for users to generate text with customizable parameters like temperature and maximum length.

Algorithm Used

Chosen Algorithm:

The core of this project is a **Generative Pre-trained Transformer (GPT)**, a neural network architecture based entirely on the self-attention mechanism. This model eschews recurrence and convolutions, allowing it to model long-range dependencies in text effectively.

Reason for Selection:

The Transformer architecture is the state-of-the-art for a wide range of natural language processing tasks, especially language modeling and text generation. Its ability to process sequences in parallel and capture global dependencies between words makes it superior to older architectures like RNNs for generating fluent and coherent text.

Advantages:

- **Efficient Performance:** The architecture, especially with optimizations like Flash Attention, is highly parallelizable, leading to significantly faster training times on GPUs compared to recurrent networks.
- **Superior Quality:** Self-attention mechanisms allow the model to draw global dependencies between input and output, resulting in higher-quality and more coherent text generation.
- **Scalability:** The model is highly scalable and can be trained effectively on very large datasets to improve its generative capabilities.

Module

List of Modules (with implementation details):

- **Data Preparation Module:** This module is responsible for loading the TinyStories-Regional/marathi-generated_4o-mini_2M dataset, tokenizing the text, and saving the processed data into efficient binary .bin files for training and validation. Two tokenization methods were considered:
  - **Method 1: Pre-trained Tokenizer (Implemented):** This method uses the existing, pre-trained l3cube-pune/marathi-gpt tokenizer from Hugging Face. This approach is straightforward and leverages a tokenizer that is already optimized for Marathi.
  - **Method 2: Custom Subword Tokenization (Alternative Approach):** This method involves training a new **Byte-Pair Encoding (BPE)** tokenizer from scratch on our specific Marathi story corpus.
    - **Reasoning:** Subword tokenization is highly effective for morphologically rich languages like Marathi. It handles out-of-vocabulary (OOV) words gracefully by breaking them into known sub-tokens, rather than mapping them to a single "unknown" token. This allows the model to better understand novel or rare words. Training a custom tokenizer ensures that the vocabulary and merge rules are perfectly tailored to the statistical properties of the story dataset, which can lead to more efficient text representation and potentially better model performance.
    - **Implementation:** To implement this, the raw text from the dataset would be used to train a new BPE tokenizer (e.g., using the tokenizers library). This would generate custom vocab.json and merges.txt files, which would then be used to tokenize the dataset for training.
- **GPTConfig Dataclass:** A configuration class that holds all the hyperparameters for the model, such as vocabulary size, block size (context window), number of layers, attention heads, and embedding dimensions.

- **GPTModel (PyTorch nn.Module):** This is the main model class. Its implementation includes:
  - Token and positional embedding layers.
  - A series of TransformerBlock modules.
  - A final layer normalization and a linear head to produce logits over the vocabulary.
  - An enhanced generate_enhanced method for text generation that incorporates top-k, top-p, temperature sampling, repetition penalty, and n-gram blocking.
- **TransformerBlock (PyTorch nn.Module):** A single block of the transformer network, which contains a MultiHeadAttention layer and a FeedForward network, with residual connections and layer normalization.
- **Trainer Class:** This class encapsulates the entire training and evaluation logic. Its implementation includes:
  - An AdamW optimizer and a Cosine Annealing learning rate scheduler.
  - A training loop with gradient accumulation to simulate a larger batch size.
  - An estimate_loss function to evaluate the model on the training and validation sets periodically.
  - Functions for saving and loading model checkpoints.
- **TextGenerator Class:** A utility class that uses the trained model and tokenizer to generate text. It provides a high-level API for standard sampling-based generation and an interactive command-line interface for users.

Methodology & Results

- **Methodology:**
  1. **Problem Analysis:** The project identified the need for a creative text generation model specifically for the Marathi language to support cultural and educational applications.
  2. **Algorithm Design & Coding:** A GPT-style transformer model was designed and implemented using PyTorch. This included creating modular classes for the overall architecture, individual transformer blocks, and attention mechanisms. A comprehensive Trainer class was also developed to handle the training process.
  3. **Data Preparation and Training:** The TinyStories-Regional/marathi-generated_4o-mini_2M dataset was processed and tokenized using the pre-trained tokenizer (Method 1). The model was then trained for 20,000 iterations with a batch size of 12 and 4 gradient accumulation steps.
  4. **Testing & Validation:** The model's performance was continuously monitored using a validation set. The final trained model was tested by providing it with various Marathi prompts to assess the coherence and creativity of the generated stories.
- **Results & Conclusion:**
  - The model was trained successfully, achieving a final validation loss of **2.6442**, which corresponds to a perplexity of approximately **14.07** with a size of about **50 Million** Parameters **3,426 times smaller** than GPT-3 which has 175 Billion Parameters.
  - The best validation loss achieved during training was **2.6005**, with a corresponding perplexity of **13.47**.
  - The generated text examples demonstrate that the model is capable of producing coherent and contextually relevant short stories in Marathi that continue from a given prompt.
  - **Conclusion:** The project successfully demonstrates the feasibility of building a custom GPT model from scratch for a regional language like Marathi and achieving good generative performance.
- **Future Scope:**
  - **GUI Integration:** Develop a user-friendly graphical interface for easier interaction with the model.
  - **Database Connectivity:** Integrate a database to store and manage prompts and generated stories.
  - **Model Scaling:** Experiment with a larger model architecture (more layers, larger embedding size) and train on a more extensive and diverse Marathi corpus to further improve generation quality.
  - **Fine-Tuning:** Fine-tune the base model for specific tasks, such as generating stories in a particular genre or writing poetry.

References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008). [Paper: https://arxiv.org/pdf/1706.03762]
2. Hugging Face Datasets. (n.d.). *TinyStories-Regional/marathi-generated_4o-mini_2M*. Retrieved from huggingface.co/datasets/TinyStories-Regional/marathi-generated_4o-mini_2M . [Paper: https://arxiv.org/pdf/2504.07989]
3. Hugging Face Transformers. (n.d.). *l3cube-pune/marathi-gpt*. Retrieved from huggingface.co/l3cube-pune/marathi-gpt [Paper: https://arxiv.org/pdf/2202.01159]
4. United Nations. (n.d.). *Goal 4: Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all*. Sustainable Development Goals. Retrieved from sdgs.un.org/goals/goal4
5. PyTorch. (n.d.). *Open source machine learning framework*. Retrieved from pytorch.org