Generatively Pretrained Transformer

Fall-2023 **Sony Kiran**

- Executive Summary

  The code implements a simplified version of the Transformer model, which is a sophisticated neural network design noted for its performance in natural language processing applications. The code is divided into parts that handle data preparation, model components, and training techniques. In this review, we will break down major functions, offer an architecture model, and cover the training and testing procedures.

- Data Preparation

  The code begins by attempting to download a dataset entitled Finalfile.txt, which looks to include the text data used for training. The text input is read and converted into numerical values using a character-level encoding technique.

- Model Architecture

  The Transformer model serves as the foundation for the core architecture, which consists of self-attention mechanisms and feedforward layers grouped into blocks. Head, MultiHeadAttention, FeedForward, and Block are some of the model's components. BigramLanguageModel is the core language model class, which includes embeddings, self-attention blocks, layer normalization, and a predictive linear head.

- Training Loop

  Using the AdamW optimizer, the training loop iterates over batches of data, computes loss, and updates model parameters. To evaluate the model's progress, evaluations are done at predetermined intervals on both the training and validation sets.

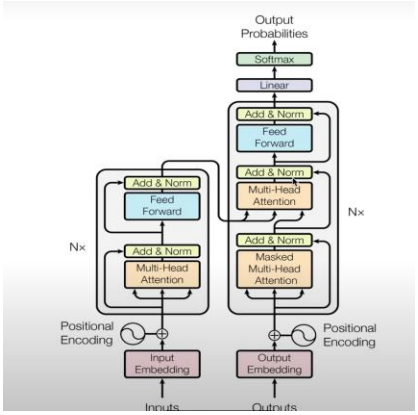
- Data Generation

  The code provides a function that uses a basic autoregressive sampling method to produce fresh text sequences from the training model.

- Components Breakdown

  Head: One head of self-attention is represented by key, query, and value linear transformations.
  MultiHeadAttention: Combines several self-attention heads in parallel.
  FeedForward: A basic linear layer is implemented, followed by a non-linearity.
  Block: A Transformer block with self-attention and feedforward components.
  components.

- Architecture Diagram



- Training Procedure

  Data Loading:

  The text data is imported and divided into training and validation sets.

  Model Initialization:

  The embedding layers, self-attention blocks, and a linear head are used to initialize the BigramLanguageModel.

Loss Computation:

The model computes the cross-entropy loss between predicted and target sequences during training.

Optimizer:

The AdamW optimizer is employed for parameter updates with a specified learning rate.

Training Loop:

Iterative training is performed, and losses are evaluated at intervals.

Model parameters are updated using backpropagation.

- Testing and Generation

Model Evaluation:

The estimate_loss function assesses the model on training and validation sets, offering insight into performance.

Text Generation:

The trained model is used to generate new text sequences.

Autoregressive sampling is employed to predict each token based on previous context.

Results Analysis:

The generated text may be examined for coherence and inventiveness, providing qualitative insights into the learnt patterns of the model.