LMA Assignment 1(BLT)

Due: 20th September 2025

Introduction

In this assignment, you will implement a simplified version of the Byte-Latent Transformer (BLT) that relies on entropy-based tokenization of data into patches to make computation more efficient for the language model and evaluate it on a simple reversal task. You will also compare it to the traditional tokenization method by training a simple model using character tokenization.

Tokenization

- Use printable **ASCII characters (32–126) + [PATCH]** as your vocabulary.
- For patching, use a sliding window mechanism with a fixed window size (e.g., W = 10). The window slides byte-by-byte, and patch boundaries are decided dynamically: (i.e can use https://kheafield.com/code/kenlm/)
    - If the Shannon entropy of the window's byte histogram exceeds a threshold (e.g., 2.0), **or**
    - If the current patch size exceeds 15, then start a new patch.
- Generate patch embeddings using hash-based n-gram embeddings with **n = 1, 2, 3**.
    - Use 4096 hash buckets for each n-gram size.
    - For each patch, extract all n-grams (up to 3-grams) and look up their embeddings from randomly initialized embedding tables (updated during training).
    - **Sum the embeddings of all n-grams in the patch** to form the final patch embedding.

Encoder–Decoder and Latent Global Transformer

- Use a **1-block transformer** for encoder and decoder and 2 blocks for global transformer.
- Each block consists of multi-head self-attention (2–4 heads), a feedforward layer, LayerNorm, and residual connections.
- Keep the vector dimension size fixed at **64** for simplicity.
- Train for **1000 epochs** on the task dataset. Choice of hyperparameters (optimizer, learning rate, etc.) are left up to you.

Task and Dataset

Your BLT implementation will be evaluated on the following task:

- **Task:** Given a string, predict the reverse of the string (it will only contain printable ASCII characters).
- **Example:** Input: `LMA is fun!` → Output: `!nuf si AML`.
- The datasets for training and evaluation will be posted on Moodle.

Baseline for comparison:

Train a separate model using **character tokenization** and a 2-block transformer on the same task.

Evaluation

Prepare a report with the following metrics for both BLT and the traditional model:

1. Accuracy on the reversal task
2. Hyperparameters used(including window size and entropy threshold).
3. Average sequence length (**#tokens vs. #patches**) on the test set

Also discuss:

- Is BLT tokenization more computationally efficient? Why or why not?

Submission Format

Submit the following on your GitHub classroom repo:

- Scripts used for training, predictions and calculation of metrics
- Report.pdf with the metrics and answers for the evaluation section along with Google drive links to saved models.
- Your predictions for the tasks as predictions_BLT.csv and predictions_normal.csv respectively for BLT and traditional models respectively. The csvs should have same headers as train and test csvs.