Homework Report

The theme of this project is *"Train a language model with next-token prediction objective on Shakespeare text"*.

For this project, I worked on implementing and training multiple transformer-based models with different configurations and tokenization schemes. The provided dataset contains approximately 4.58 million characters, and I processed it by first extracting all unique characters to create a vocabulary. This allowed me to map characters to integer indices and vice-versa. The dataset was split into training amd validation subsets (90-10 split). Although I have tried 80-20 and 85-15, we get better results with 90-10.

For the model architectures, I tested two sizes: small and large, each with two tokenization schemes: character-level and subword-level. Both models used a context window of 128 tokens. The small models used a hidden dimension of 488, 6 transformer layers, 7 attention heads, and a dropout rate of 0.1. For the subword tokenization, I applied Byte Pair Encoding (BPE) to generate a vocabulary of 10,000 subwords. Due to computal limitations, I chose to go with the smaller model.

In terms of training, I used a batch size of 16 and trained each model for 5,000 iterations with an evaluation interval every 100 steps. The AdamW optimizer was employed with a learning rate of $3*10^{-4}$. I monitored the training and validation loss, and I calculated perplexity as the main metric for model performance. Perplexity is particularly relevant because it reflects the model's ability to predict the next token in a sequence, adjusted for vocabulary size.

I also generated graphs for training and validation perplexity across the different configurations. These graphs allowed me to observe the impact of tokenization and model size on performance. In general, subword-level models performed better in terms of perplexity, which I believe is due to their ability to capture more meaningful patterns compared to character-level tokenization. The larger models also consistently outperformed the smaller ones, as expected, given their increased capacity to model complex dependencies.

For qualitative evaluation, I sampled text from the hold-out set and compared it to the model's predictions. I used top-k sampling with a k value of 10 for generating text. This approach provided more diverse outputs while avoiding repetitive patterns. I observed that the subword-level models generated text with better syntax and coherence, while the character-level models struggled with longer dependencies. Therefore, I am somewhat satisfied with the results, knowing that there is lots of room for improvement.

Bibliography

Let's build GPT: from scratch, in code, spelled out. - Andrej Karpathy