Pretrained Transformer with GPT-trained Tokenizers on Shakespeare Text Generation

Peter Liu
Department of Statistics
Stanford University

March 7, 2025

Abstract

In Part1 of the homework we've trained a vanilla transformer on a fraction of Shakespeare play with character-level tokenizations, which produced Shakespeare-like gibberish text. In this project, I endeavor to improve the model by replacing positional embedding lookup with rotary positional embedding (RoPE), adding dropout layer in feedforward layer, and pretraining the model on a larger dataset first. The generated texts from the two models were fed to the pretrained GPT-2 model for perplexity-score calculation to determine which model's result is superior. In the end, the improved model not only has better quality text generated, but also took slightly less time to train upon (excluding pretraining step).

1 Background

Since the advent of self-attention mechanism, transformer has largely become the backbone of many dominant LLMs today like GPT-4o or Llama-3B as its unique design allows the model to understand the interaction between words in a context-dependent ways. Furthermore, self-attention allows the computations of the relationship between all tokens simultaneously, thus allowing more efficient training than traditional RNNs.

In Part I of the homework, we've tokenized a Shakespeare text on the character-level and using a 9:1 train-validation split ratio and deployed a vanilla transformer which is assembled using the basic self-attention mechanism. Overall it has 6 attention layers with each layer containing a 6-head attention matrix with head size K, where the output of each head is concatenated and applied through a projection layer to the original token embedding size $D = 384$. Afterwards, a Feedforward layer is introduced for each layer of attention output with RELU to introduce non-linearity. In each sublayer, layer normalization is also applied before and after Feedforward layer to introduce stability. The learnable parameters (e.g. key, query matrices, embedding lookup matrix etc.) are trained using cross-entropy loss between predicted character token and the actual token. We used context size $T = 256$ throughout.

Finally, we used the trained transformer to produce Shakespeare-like texts with some original text as the starting context. The result is mostly gibberish, but it does have some Shakespearean flair to it. The model improvements methods are discussed in the next section.

2 Improvement Methods

2.1 Tokenizers

The original Shakespeare text was tokenized into individual characters, which results in a 65-char 'vocabulary' size. Such tokenization has the advantage of keeping the embedding look up matrix minimal (65*384), however it is intrinsically less reliable as character itself has little meaning than subwords or words itself. A convenient replacement method is to use a more robust, pretrained tokenizer. Neverntheless tokenizers such as GPT-2 has more than 50,000 unique tokens, which drastically increases the size of embedding lookup matrix and the training time as well.

The compromise I decided to go with is to use Byte-Pair Encoding (BPE), a method based on subword modelling which breaks down the original data into character first and then starts to merge the most common subwords until a stopping criteria is met. The nice thing about it is that we can set a max vocabulary size limit so that we don't end up with a forbiddingly large token embedding matrix. In the end I've set the size to 10,000 as a compromise between robustness and memory efficiency.

2.2 Architecture Improvements

In terms of architecture, I've kept the general architecture of transformer and made only minor changes that aim to make the baseline a more efficient and robust model.

2.2.1 Rotational Positional Embedding (RoPE)

The major update is the use of RoPE for each token, which eliminates the need of keeping a separate positional embedding look up for each token, saving us some GPU memory. Specifically, RoPE applies a rotation to the key and query vectors along the head context length dimension, where tokens at different positions are rotated by different amount. This allows the subsequent attention score computation to still be dependent on positions. Mathematically, given a token embedding x of D-dimension, the rotated key/query output is given by $R^D_{\Theta,m} W_{q/k} x$ where:

$$R^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{D/2} & -\sin m\theta_{D/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{D/2} & \cos m\theta_{D/2} \end{pmatrix}$$

2.2.2 Dropout Layer

Another minor change is the adding of Dropout Layer to attention output before and after the feedforward layer in each of the 6 transformer layer, which serves as a straightforward method to

prevent overfitting in deep neural network. Each dropout layer is set with the drop-out probability of 0.2

2.3 Pretraining & Finetuning

The pretraining-finetuning paradigm is mainly why LLMs has found many applications nowadays, as people take the open-sourced LLMs that are trained on huge amount of corpus with vast amount of compute and then finetune on their own datasets, which generally produce great results and saves humanity a great deal of time. To follow this paradigm, I've collected various English poems/essays/plays from *Gutenburg* by various authors from the Shakespearean time like Christopher Marlowe and Edmund Spencer, which results in a pretraining text roughly three times the size of our Shakespeare text. After pretraining, the updated model weights are saved, where they are subsequently used as starting parameters during finetuning with the Shakespeare text.

I've used 3000 iterations for pretraining and 5000 iterations for finetuning. During each iteration, a batch size of 32 training and target tokens of length 256 are randomly selected from the training set. All the model training and finetuning was carried out locally on 2021 MacBook Pro with M1-Pro GPU chip, which took around 40min for pretraining and 1hr15min for finetuning (around 1sec per iteration)

3 Evaluation

The evaluation of our original vanilla transformer and the upgraded, pretrained transformer is worth some discussion. During training, the cross entropy loss between the model predictions for the next-token and the targets are used to update learnable parameters, and the loss curve during training and final validation loss serve as a benchmark for each model group. However, such calculation is token vocabulary dependent, and since our two versions use different tokenization scheme, a direct comparison between the model loss is not feasible. Therefore, we need a numerical criterion that can directly compare the quality of the two models' generated output.

The evaluation method I decided to use is to feed the generated output to a pretrained GPT-2 and have it compute the text perplexity score from its 'own judgment'. In the end, a total of five different seed texts excerpted from Shakespeare plays were used as starting context for the two models to generate an additional 256 tokens, which were then fed to GPT-2 to calculate the perplexity score of each generated output.

4 Results

The table below demonstrates the GPT-2 perplexity scores for five generated texts under vanilla and upgraded versions of transformers, showing our upgraded version of transformer performs better on Shakespearean text generation from all five seed texts. In addition, a direct comparison of the generated texts from two model versions from the same seed text (left vanilla, right upgraded), where the perplexity score difference is the largest. Note that only a portion of upgraded transformer text is shown as it is significantly longer than vanilla transformer's generation. Upon a visual examination, the vanilla text has a lot of made-up words like *Enten* and *bunwiteing*, while the

upgraded text clearly has a high vocab accuracy, while being more coherent as well (althought still not very logical)

Table 1: Perplexity Scores on Generated Texts

	Text1	Text2	Text3	Text4	Text5
Vanilla	381	223	473	316	677
Upgraded	**302**	**167**	**199**	**257**	**251**

I'll look to like, if looking liking move:
her T's, and thus makes she her great P's.
It is in contempt of
(— generated text below —)
Were is If slander steem posers, seand the king
Enten child, in there must fearfer cause
Off the emberals it, stir wars'le of the vertal down
The lutters all Bolingbate they counserss
Of the bunwiteing the world march's king:
That words, to my venry scorro

I'll look to like, if looking liking move:
her T's, and thus makes she her great P's.
It is in contempt of
(— generated text below —)
Ford to-morrow, if she switch him sleep,
Dorace, breast, let him ask your language.
For you, poor babe, I advise you:
I tender you with your cousin'ld rederer.
Those are you yet he gives not well a man,
Nor thou substital'd against a man.
'Tis not a cunning thing; no shame shall live,
...

5 Discussion

It's also noteworthy to examine other aspects of the models besides the perplexity of the generated text. Size-wise, vanilla transformer is more compact: taking 84MB storage compared to 113MB of upgraded transformer. This is expected as the upgraded version of transformer has a much larger memory requirement for token embeddings. However, it eliminates the to keep a separate embedding for positions. Another concern is the training time: together with pretraining and finetuning, it took around 2hrs of GPU computation to train our upgraded transformer compared to under 40min for vanilla transformer. However, such sacrifices in storage and time are deemed worthwhile given the noticeable improvement in the quality of generated text.

Certain aspects of the methodology used in this project is also worth discussion. Firstly is the pretraining text choice: the texts I've collected are somewhat arbitrary, and it also very limited in size and scope. Given more time and compute resources, I could potentially gather a much larger and more diverse corpus for pretraining. Secondly is the evaluation criterion, as one can doubt the equivalence between a lower GPT-2 perplexity score and 'text-being-more-Shakespearean-like'(which is our end goal); additionally, there are other quantitative methods out there like BERTScore that I haven't tried.

Finally, during research I came across the concept of 'fast attention' which is supposed to make training faster and more efficient. It would be interesting for the future to implement this as well and compare the training time and generated output quality to our upgraded transformer results.

References

[1] Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2023). RoFormer: Enhanced Transformer with Rotary Position Embedding. Retrieved from `https://arxiv.org/abs/2104.09864`.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention Is All You Need. Retrieved from `https://arxiv.org/abs/1706.03762`.