On the Use of the Variance of Logits in Tokenizing and Training

Adarsh Senthil

November 5, 2023

Abstract

A tokenizer and loss function for Large Language Models based on the surmise that the meaningfulness of a sentence is related to the vanishing of the variance of logits. Code repository link: https://github.com/Adsen14/LLM-Logit-Variance

Large Language Models generate text by generating a logit vector at each timestep and sampling a component from it. Each component of the logit vector can be interpreted as the likelihood that the component is the correct token for the next timestep. As text is being generated, the previous tokens in the sentence act as constraints for the next token. The constraints change the amount of equally likely tokens. For LLMs, we use the variance of the top-k elements of the logit vector to infer the degree to which every token in the top-k elements is equally likely. If the next timestep is highly constrained, only a couple of tokens are likely to be the next token, and the variance would be far from zero. If the next timestep is lightly constrained, almost every token is likely to be the next, and the variance would be close to zero. A partial sentence is likely to be meaningful if its final token is lightly constrained, and meaningless/ambiguous if its final token is highly constrained. Measuring the meaningfulness of a sentence in this way may help in:

- Splitting sentences into meaningful pieces

- Marking tokens that end a meaningful sentence, which may help the LLM to generate meaningful text by ensuring that the variance of the logits vanish at the marker tokens

For the sentence piece tokenizer, we take the prompt, the top-k value (k), and the absolute tolerance (r) as inputs. We create an empty array to store the pieces, and an empty array to store the current piece. For each timestep, we append the current output token to the current piece, and apply the softmax function to the evaluated logits. Let the variance of the logits of the top-k tokens at the i^{th} timestep be $\sigma^2(x_i[:k])$. If $|\sigma^2(x_i[:k])| \leq r$ (i.e. $\sigma^2(x_i[:k]) \in B(0;r)$), we append the current piece to the pieces array, and clear the current piece. When the model outputs the EOS token, we return the pieces array P as the result.

For the loss function, we create a marker tokens array for the training dataset. It contains only the indices of the marker tokens. We can manually label the marker tokens, or we can specify a fixed set of marker tokens that are used in common language, such as "." and ",". Let the marker tokens array be denoted as M. We make the following modification to the cross-entropy loss function to accomodate the marker tokens,

$$l(y_t, \hat{y}_t) = -\sum_{j=1}^{|V|} y_{t,j} \log \hat{y_{t,j}},\ t \notin M$$

$$l(y_t, \hat{y}_t) = -\sum_{j=1}^{|V|} y_{t,j} \log \hat{y_{t,j}} + \frac{1}{|V|}(\sum_{j=1}^{|V|}(\frac{1}{|V|} + \hat{y_{t,j}})^2),\ t \in M$$

We surmise that this loss function can help in speeding up training, since the LLM has an additional constraint of bringing the variance of its logits down to zero when reaching a marker token, which may help the LLM generate more meaningful text for the same training time.

References

1. Aston Zhang, Zachary C. Lipton, Mu Li, Alexander J. Smola. *Dive into Deep Learning.*

2. Li Deng, Yang Liu. *Deep Learning in Natural Language Processing.* Springer Nature Singapore Pte Ltd., 2018.