Ben KABONGO B.
ben.kabongo_buzangu@ens-paris-saclay.fr

Lab session # 3
ALTEGRAD 2023

10/29/23

1 Question 1

1.1 Square mask

In our implementation, the square mask is used to ensure that during the self-attention operation, each token can only consider tokens previous to or equal to itself, but not future tokens.
This strategy ensures that the attention mechanism does not take future tokens into account when predicting based on past tokens, by assigning a value of $-\infty$ to future positions before performing the weighting phase in the self-attention calculation. This deliberately prevents the model from taking into account information from items after the current position, ensuring that only past information is used for prediction. [2]

1.2 Positional encoding

In the absence of recurrence and convolution, it is necessary to find a way of allowing the model to take into account the order of the sequence, in particular by injecting information about the position of the tokens. In this way, the positional encoding of tokens is coupled with their embeddings to provide this information on the position of tokens in the sequence. [2]

2 Question 2

Our framework comprises two main stages: pre-training and fine-tuning. The pre-training phase is language modeling; and for the fine-tuning phase, we do sentiment classification. [1]

2.1 Transfer learning

We employ a style transfer approach, wherein our model undergoes unsupervised pre-training on unlabeled text. We leverage the knowledge acquired during this pre-training phase by reusing the learned parameters when initializing the model for various fine-tuning tasks, including classification. This strategy eliminates the need to retrain a similar model from scratch for fine-tuning.
Since the objectives of pre-training and fine-tuning the model may not align perfectly, it is essential to preserve the parameters of the base model. Consequently, for the fine-tuning of our model, we opt to replace only the model head with a classification head, while initializing it with the parameters derived from the base model.

2.2 Language modeling and classification tasks

The main difference between language modeling and classification tasks are :

2.2.1 Objective

- **Language Modeling**: the objective is to predict the probability distribution of the next token in a sequence given the previous context.
- **Classification**: the objective is to assign a category or label to an input text.

2.2.2 Output

- **Language Modeling**: the output of a language modeling task is a probability distribution over the entire vocabulary of tokens.
- **Classification**: the output of a classification task is a single class or label from a predefined set of categories or classes.

2.2.3 Training Data

- **Language Modeling**: Language models are trained on large corpora of text, where the training data consists of text sequences without specific class labels.
- **Classification**: Classification models are trained on labeled data, where each input sequence is associated with a specific class or category.

3 Question 3

$n_{tokens} = 100$	vocabulary size
$n_{hid} = 200$	hidden dimension of the transformer layers
$n_{layers} = 4$	number of transformer layers in the encoder
$n_{classes} = 2$	number of classes for the classification task
$n_{embedding} = n_{tokens} \times n_{hid} = 20000$	number of embedding parameters
$n_{attention} = 4 \times n_{hid} \times (n_{hid} + 1) = 160800$	number of attention parameters
$n_{FC} = 2 \times n_{hid} \times (n_{hid} + 1) = 80400$	number of fully connected parameters for attention heads
$n_{layer\_norm} = 4 \times n_{hid} = 800$	number of parameters in the layer norm

The total number of parameters in an encoder layer is given by :

$$n_{encoder\_layer} = n_{attention} + n_{FC} + n_{layer\_norm} = 242000$$

The number of parameters of the base model is given by :

$$n_{base} = 4 \times n_{encoder\_layer} + n_{embedding} = 988000$$

The number of trainable parameters is given by :

- **Classification** : $n_{classif} = n_{base} + n_{hid} \times n_{classes} + n_{classes} = 988402$
- **Language modeling** : $n_{classif} = n_{base} + n_{hid} \times n_{tokens} + n_{tokens} = 1008100$

Upon examining the inner workings of PyTorch's implementation of the TransformerEncoderLayer, it becomes evident that in the absence of specified dimensions for K and V, the parameters for the input section of the attention mechanism adopt the form of ($3n_{hid}$, n_{hid}) for Q, K, and V (accompanied by an additional $3n_{hid}$ parameters for bias). Similarly, the output section of the attention mechanism takes on the shape of (n_{hid}, n_{hid}) with n_{hid} parameters for bias.
In the context of classification, we incorporate a linear layer with dimensions (n_{hid}, $n_{classes}$) along with a bias term.
In the context of language modeling, we incorporate a linear layer with dimensions (n_{hid}, n_{tokens}) along with a bias term.

4 Question 4

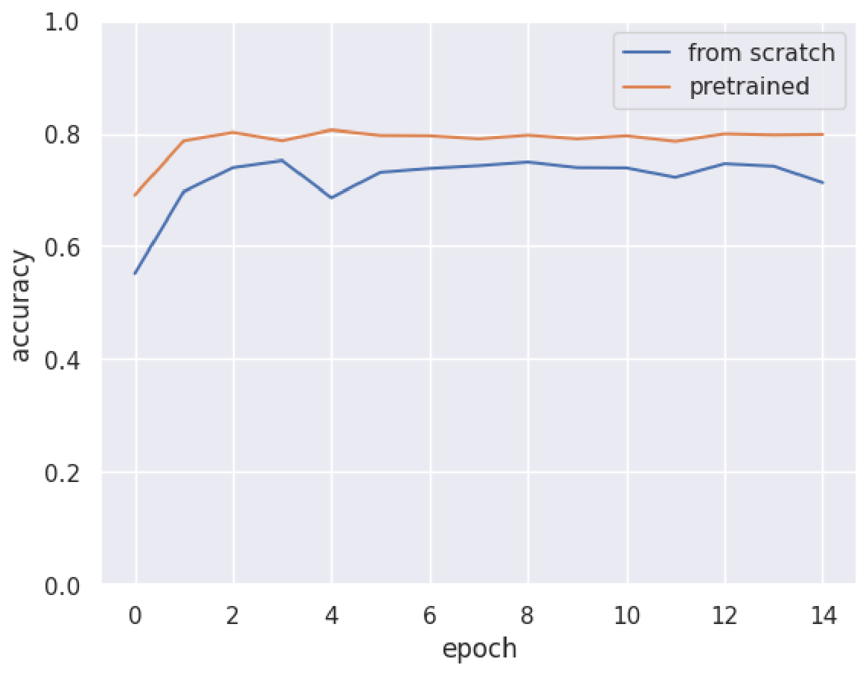

Figure 1: Enter Caption

Basic performance

- The from scratch model starts with a higher initial loss, indicating that it has no prior knowledge.
- The pretrained model starts with a lower initial loss, as it benefits from prior knowledge gained from pre-training.

Improvement over time

- The from scratch model shows a progressive improvement in loss and perplexity over time.
- The pretrained model also shows progressive improvement in loss and perplexity over time, but it starts from a pre-trained base, enabling it to reach lower perplexity values more quickly.

Classification ability

- Both models show increases in accuracy over epochs, indicating that they adapt well to the classification task.
- The pretrained model tends to achieve higher Accuracy values more quickly, thanks to its prior knowledge.

The from scratch model starts with no prior knowledge at all, and needs more epochs to improve. The pretrained model starts from a pre-trained base, enabling it to achieve better performance more quickly. Both models eventually show classification capability, but the pre-trained model is more efficient in terms of using prior knowledge.

5 Question 5

In our work, we model a unidirectional language. The probability of a token is given in relation to the tokens preceding it. When calculating the attention for a given token, the tokens that follow it are masked. A limitation of this model is that it only takes context into account in one direction. Taking context into account in both directions can be more informative; this is what is done in the paper [1].
In the paper [1], they present a bidirectional modeling of language. And for the masked language modeling task presented, when calculating attention, some tokens are masked and replaced by the special token [MASK]. This modeling allows context to be taken into account in both directions.

References

[1] Jacob Devlin and et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.