Julien Delavande
julien.delavande@ens-paris-saclays.fr

Question 1

The square mask, also known as the causal mask or subsequent mask, is used in the Transformer decoder to prevent the model from attending to future tokens when predicting the next token in a sequence. Specifically, it masks out (assigns a value of negative infinity to) the attention scores for positions j where $j > i$, ensuring that the prediction at position i only depends on the positions $\leq i$. This is essential for autoregressive language modeling, where the goal is to predict the next word based solely on the previous words.
Mathematically, the mask M is defined as:

$$M_{i,j} = \begin{cases} 0 & \text{if } j \leq i, \\ -\infty & \text{if } j > i. \end{cases}$$

Applying this mask before the softmax operation in the attention mechanism ensures that the model cannot "peek" at future tokens during training.
The positional encoding addresses the Transformer's lack of inherent positional information due to its non-recurrent, non-convolutional architecture. Since the Transformer processes tokens in parallel and does not have a built-in notion of sequence order, positional encodings inject information about the position of each token in the sequence. This is typically done by adding sinusoidal functions to the input embeddings:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right),$$
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right),$$

where pos is the token position, i is the dimension, and d_{model} is the model's dimensionality. This allows the model to distinguish between tokens at different positions and to capture the sequential nature of language.

Question 2

We need to replace the classification head because the output requirements for language modeling and classification tasks are fundamentally different. In language modeling, the model's objective is to predict the next token in the sequence, which requires the final layer to output a probability distribution over the entire vocabulary. Therefore, the final linear layer maps from the model's hidden dimension d to the vocabulary size V.

In contrast, the classification task involves assigning a single label (e.g., positive or negative sentiment) to an entire sequence. The classification head must map from the hidden dimension d to the number of classes C, which is typically much smaller than the vocabulary size. Consequently, we replace the language modeling head with a new classification head that is randomly initialized.

The main difference between the two tasks is:

- **Language Modeling:** Predicts the next token in a sequence, requiring the model to output a distribution over all possible tokens in the vocabulary at each position.
- **Sequence Classification:** Assigns a label to the entire input sequence, requiring the model to output a distribution over the set of class labels based on the sequence's overall representation.

Question 3

To calculate the number of trainable parameters in the model for both tasks, we consider the following components:

- **Embedding Layer:**
  - Token Embeddings: $V \times d$
    V beeing the size of the vocabulary and d the size of the hidden representation of a token.

- **Positional Embeddings:**
  - Positional Embeddings: We didn't use trainable parameters in our implementation.
- **Transformer Layers (per layer):**
  - Multi-Head Attention Sub-layer:
    d_K being the hidden representation of a token inside the projection space for Query and Key space, d_V being the hidden representation of a token inside the projection space for the Value space. In the implementation we use $d_K = d_V = \frac{d}{h}$.
    - Query, Key, Value weights: $2 \times d \times d_K + d \times d_V$
    - Number of heads: h
    - Output projection: $hd_V \times d$
    - Total: $h \times (2 \times d \times d_K + d \times d_V) + hd_V \times d = 4d^2$
  - Feed-Forward Network Sub-layer:
    d_{ff} is the hidden layer size in between the two feed forward layers. In the implementation we used $d_{\text{ff}} = d$.
    - First linear layer: $d \times d_{\text{ff}} + d_{\text{ff}}$
    - Second linear layer: $d_{\text{ff}} \times d + d$
    - Total: $2d \times d_{\text{ff}} = 2d^2 + 2d$
  - Total per Transformer layer: $6d^2 + 2d$
- **Final Linear Layer:**
  - Language Modeling: $d \times V$
  - Classification: $d \times C$

Assuming the following values:

- Vocabulary size (V): 5000
- Embedding dimension (d): 200
- Number of Transformer layers (L): 6
- Number of classes (C): 2

Calculations:

- **Embedding Layer:**
$$\text{Token Embeddings} = V \times d = 5000 \times 200 = 1{,}000{,}000$$
- **Transformer Layers:**
  - Per Layer:
$$\text{Attention Parameters} = 4d^2 = 4 \times 200^2 = 160{,}000$$
$$\text{Feed-Forward Parameters} = 2d^2 + 2d = 2 \times 200^2 + 2 \times 200 = 80{,}000$$
$$\text{Total per Layer} = 160{,}000 + 80{,}000 = 240{,}000$$
  - Total for All Layers:
$$\text{Total Transformer Parameters} = L \times \text{Total per Layer} = 6 \times 240{,}000 = 1{,}440{,}000$$
- **Final Linear Layer:**
  - Language Modeling:
$$\text{Final Layer} = d \times V = 200 \times 5000 = 1{,}000{,}000$$
  - Classification:
$$\text{Final Layer} = d \times C = 200 \times 2 = 400$$

- **Total Trainable Parameters:**
  - Language Modeling Task:

$$\begin{aligned}\text{Total Parameters} &= \text{Embeddings} + \text{Transformer Layers} + \text{Final Layer} \\ &= 1{,}000{,}000 + 1{,}440{,}000 + 1{,}000{,}000 \\ &= 3{,}440{,}000\end{aligned}$$

  - Classification Task:

$$\begin{aligned}\text{Total Parameters} &= 1{,}000{,}000 + 1{,}440{,}000 + 400 \\ &= 2{,}440{,}400\end{aligned}$$

Summary:

- Language Modeling Task: $3{,}440{,}000$ trainable parameters.
- Classification Task: $2{,}440{,}400$ trainable parameters.

Explanation:

The language modeling task has more parameters due to the larger final linear layer mapping to the entire vocabulary. The classification task's final layer is much smaller because it only needs to map to the number of classes. The rest of the model remains the same in both cases.

Question 4

The results show that the model utilizing transfer learning significantly outperforms the model trained from scratch in terms of both convergence speed and final accuracy on the sentiment analysis task.

- **Transfer Learning Model:**
  - Faster Convergence: The pre-trained model reaches higher accuracy levels in fewer epochs.
  - Higher Final Accuracy: It achieves better performance, indicating that the pre-trained weights capture valuable linguistic features that generalize well to the new task.
  - Efficient Training: Since only the classification head is randomly initialized and trained from scratch, the model benefits from the representations learned during pre-training.
- **Model Trained from Scratch:**
  - Slower Convergence: The model takes more epochs to reach comparable accuracy levels.
  - Lower Final Accuracy: It may not achieve the same level of performance as the transfer learning model due to the lack of pre-learned language representations.
  - Increased Training Effort: Learning both language representations and task-specific patterns simultaneously from limited data is challenging.

Interpretation:

The experiment demonstrates the effectiveness of transfer learning in NLP. By leveraging a pre-trained language model, we can significantly improve performance on downstream tasks with limited labeled data. The pre-trained model provides a strong foundation of linguistic knowledge, enabling it to quickly adapt to new tasks with minimal additional training. In contrast, training from scratch is less efficient and may not capture the complex patterns present in language data.

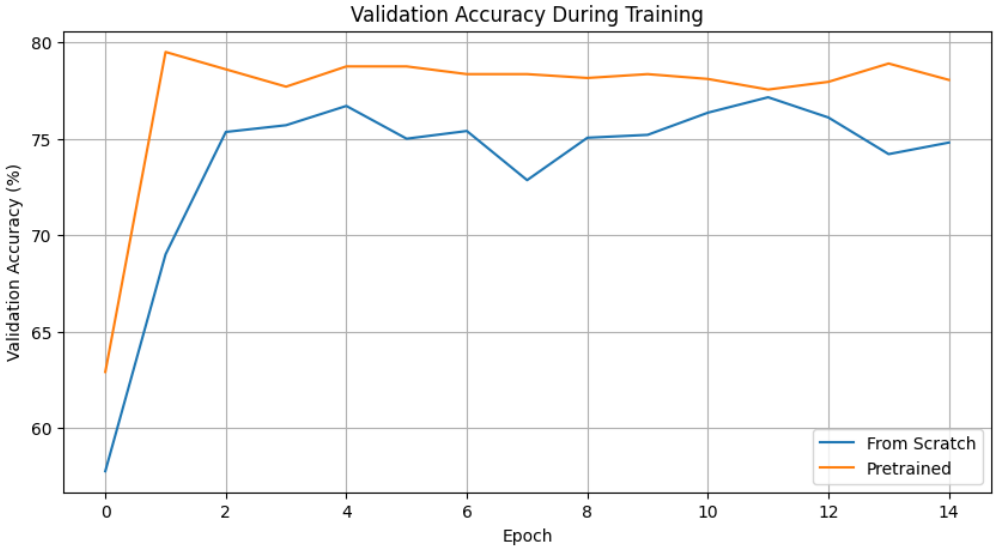

Figure 1: Evolution of the validation accuracy during training for both pre-trained and from scratch models.

Question 5

One of the limitations of the language modeling objective used in this notebook is that it is unidirectional; the model predicts the next token based solely on the preceding tokens (left-to-right context). This approach prevents the model from capturing dependencies on future tokens during training, which can be crucial for understanding the full context of a sentence.

In contrast, the masked language model (MLM) objective introduced in BERT [1] allows the model to utilize bidirectional context. By randomly masking tokens in the input sequence and training the model to predict these masked tokens using both left and right context, the MLM objective enables the model to learn richer and more nuanced representations of language. This bidirectional understanding is particularly beneficial for tasks that require comprehension of the entire input sequence, such as sentiment analysis or question answering.

Therefore, the limitation is that the standard language modeling objective does not allow the model to leverage information from future tokens during training, potentially leading to less effective representations for downstream tasks that benefit from bidirectional context.

References

[1] Devlin, Jacob, et al. ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” *arXiv preprint arXiv:1810.04805* (2018).