Kyllian Asselin de Beauville
kyllian.asselin-de-beauville@polytechnique.edu

1 Question 1

In our transformer-based implementation for language modeling and classification tasks, the role of the square mask is to maintain the autoregressive nature of predictions, where we want to predict the next word based on the previous words, and not the other way around. Specifically, the mask ensures that during the attention computation, a given token does not attend to subsequent tokens in the sequence. By doing this, we avoid the possibility of the model using future information to predict the current token, thereby preventing it from "cheating" during training.
To achieve this, we construct the mask as a triangular matrix where the upper right portion is set to $-\infty$. When the scaled dot-product attention (Figure 1) is applied to these values, they become zero due to the softmax, effectively masking the future tokens. This approach is similar to the one described for the decoder in *Attention is All You Need* [3], where illegal connections in the attention mechanism are masked out by setting them to $-\infty$, ensuring no leftward information flow in the decoder to preserve the autoregressive property.

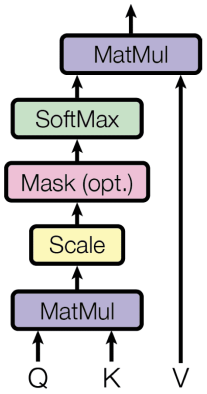

Figure 1: Scaled Dot-Product Attention.

Transformers are inherently permutation invariant, meaning they don't have a sense of the order of tokens, unlike RNNs, LSTMs, or GRUs, which process sequences token-by-token and inherently understand sequence order. Positional encodings are there to give transformers the ability to understand the order of tokens in a sequence.
The approach used to generate these encodings is based on sinusoidal functions. For any given position pos and dimension i, the positional encoding values are determined by:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{1}$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{2}$$

Where d represents the dimension of the embeddings. These sinusoidal functions guarantee a unique encoding for every position. After these encodings are generated, they are added directly to the token embeddings, giving the model the ability to understand the relative positions of tokens in a sequence and to make full use of it.

2 Question 2

The need to replace the classification head comes from the difference in objectives and expected outputs for language modeling compared to classification:

- The primary goal of language modeling is to predict the next token in a sequence given its context. It models the probability distribution over the entire vocabulary for each token in a sequence. As a result, the output layer, or the "head", of a language model consists of a linear layer with a size equal to the size of the vocabulary, followed by a softmax activation if we want to get the probabilities for each word in the vocabulary.
- In classification, the objective is to assign a class to an input sequence. For instance, in sentiment analysis, given a book review, the model determines if it is a positive or negative review. Here, the output layer is designed to represent each potential class. This involves a linear layer whose size equals the number of classes, followed by a softmax activation in the case of multi-class classification.

Pre-trained transformer models, such as GPT-3 [1], are usually trained on large-scale language modeling tasks. These models capture a broad understanding of language but are not fine-tuned to specific tasks. When adapting these models for tasks like classification, we replace the original language modeling head with a classification head to ensure that the model's final layers are optimized for producing class probabilities rather than word probabilities. Replacing the head allows for fine-tuning on a smaller, task-specific dataset, by learning the nuances and specifics of the task while having a good general language understanding thanks to the pre-training phase.

In summary, while the underlying transformer architecture can be shared across various tasks, the head of the model, which determines its final output, needs to align with the specific objectives and output format of the task it is designed to perform.

3 Question 3

Let's first define the parameters of the model for which we are computing the number of trainable parameters: $n_{\text{tokens}} = 100$ (size of the vocabulary), $n_{\text{hid}} = 200$ (hidden dimension), $n_{\text{layers}} = 4$ (number of transformer encoder layers), $n_{\text{head}} = 2$ (attention heads). For the classification, we will also need: $n_{\text{classes}} = 2$ (number of classes). **In the following operations, $|X|$ will denote the number of parameters of X.**

3.1 Base model

For the embedding layer, we have an embedding matrix E of size $n_{\text{tokens}} \times n_{\text{hid}}$:

$$|E| = n_{\text{tokens}} \times n_{\text{hid}} = 20000 \tag{3}$$

The positional encodings are fixed and do not introduce trainable parameters.

The transformer block has weights for the query, key, and value projections. Each of these is of size $n_{\text{hid}} \times n_{\text{hid}}$. Additionally, there are output weights of size $n_{\text{hid}} \times n_{\text{hid}}$.

$$|W_{Q,K,V}| = 3 \times (n_{\text{hid}} \times n_{\text{hid}} + n_{\text{hid}}) = 120600 \tag{4}$$
$$|W_O| = n_{\text{hid}} \times n_{\text{hid}} + n_{\text{hid}} = 40200 \tag{5}$$
$$|W_{\text{attention}}| = |W_{Q,K,V}| + |W_O| = 160800 \tag{6}$$

It is followed by a layer norm, a feed-forward layer, and another layer norm. Here are the associated parameters:

$$|W_{\text{feed-forward}}| = n_{\text{head}} \times (n_{\text{hid}} \times n_{\text{hid}} + n_{\text{hid}}) = 80400 \tag{7}$$
$$|W_{\text{layer norm}}| = 2 \times n_{\text{head}} \times n_{\text{hid}} = 800 \tag{8}$$

The full number of parameters P_{base} of the base model can be computed as follows:

$$P_{\text{base}} = |E| + n_{\text{layers}} \times (|W_{\text{attention}}| + |W_{\text{feed-forward}}| + |W_{\text{layer norm}}|) \tag{9}$$
$$= 20000 + 4 \times (160800 + 80400 + 800) \tag{10}$$
$$= 988000 \tag{11}$$

3.2 Language modeling

For the language modeling, the head is a linear layer with n_{hid} input features and n_{tokens} output features. The number of parameters of this layer is:

$$P_{\text{language modeling head}} = n_{\text{hid}} \times n_{\text{tokens}} + n_{\text{tokens}} = 20100 \tag{12}$$

The total number of trainable parameters for the language modeling task is:

$$\begin{aligned} P_{\text{language modeling}} &= P_{\text{base}} + P_{\text{language modeling head}} && (13)\\ &= 988000 + 20100 && (14)\\ &= 1008100 && (15) \end{aligned}$$

3.3 Classification

For the classification, the head is a linear layer with n_{hid} input features and n_{classes} output features.

$$P_{\text{classification head}} = n_{\text{hid}} \times n_{\text{classes}} + n_{\text{classes}} = 402 \tag{16}$$

And the total number of trainable parameters for the classification task is:

$$\begin{aligned} P_{\text{classification}} &= P_{\text{base}} + P_{\text{classification head}} && (17)\\ &= 988000 + 402 && (18)\\ &= 988402 && (19) \end{aligned}$$

4 Question 4

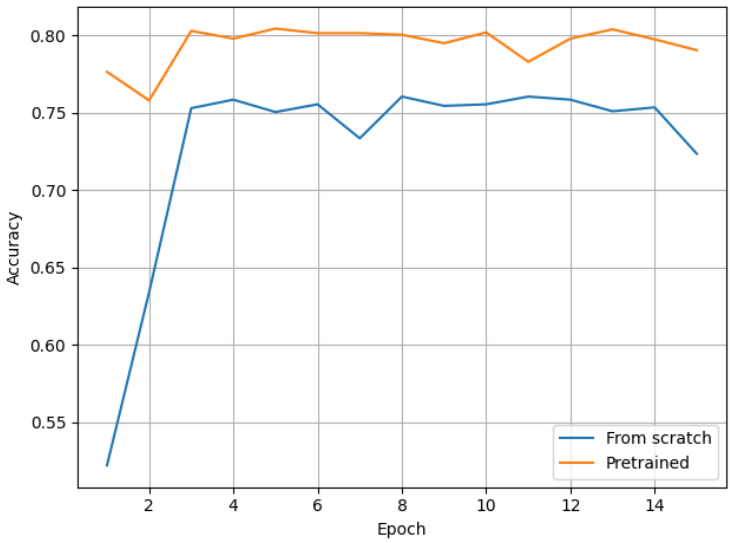

Figure 2: Accuracy of pre-trained and “from scratch” models over epochs.

The model with pre-trained weights starts with a significantly higher accuracy than the model trained from scratch ($\approx 77\%$ vs. 52%). This suggests that the pre-trained model benefits from a good initialization, taking advantage from its prior knowledge and natural language understanding. It’s indicative that the pre-trained model already possesses substantial relevant knowledge even before fine-tuning.

The model trained from scratch, while starting at a lower accuracy, shows a rapid improvement in the initial epochs. This could be attributed to it learning basic language structures and patterns. On the other hand, the pre-trained model, after an initial rise, plateaus relatively quickly. This behavior is consistent with the

understanding that the pre-trained model requires fewer epochs to adapt to the specific task, thanks to the prior knowledge it brings.

Towards the later epochs, both models seem to stabilize in terms of accuracy. However, the pre-trained model still outperforms the model trained from scratch ($\approx 79\%$ vs. 73%). This underlines the power of transfer learning. Within the span of 15 epochs, the from-scratch model struggles to match the performance of the pre-trained counterpart. It would require more epochs for the from-scratch model to potentially compete with the pre-trained one. Given the costs associated with extended training, leveraging pre-trained models often presents a more efficient option.

5 Question 5

The traditional language modeling objective, as used in the notebook, predicts the next word in a sequence given the previous words. It works in a unidirectional way, by predicting the next token given the previous ones. This limits the context it can utilize for each prediction to only the previous tokens.

On the other hand, the masked language model objective introduced in BERT [2] predicts randomly masked-out words in a sentence by leveraging both the left and right context. By predicting masked tokens, the masked language model objective trains the model to understand the context from both directions, leading to a deeper and more bidirectional understanding of text. This bidirectional context helps the model to better capture semantic relationships between words and phrases in a sentence.

Thus, one of the limitations of the traditional language modeling objective, compared to the masked language model objective, is its unidirectional nature, which restricts its capacity to fully grasp the bidirectional context present in language.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.