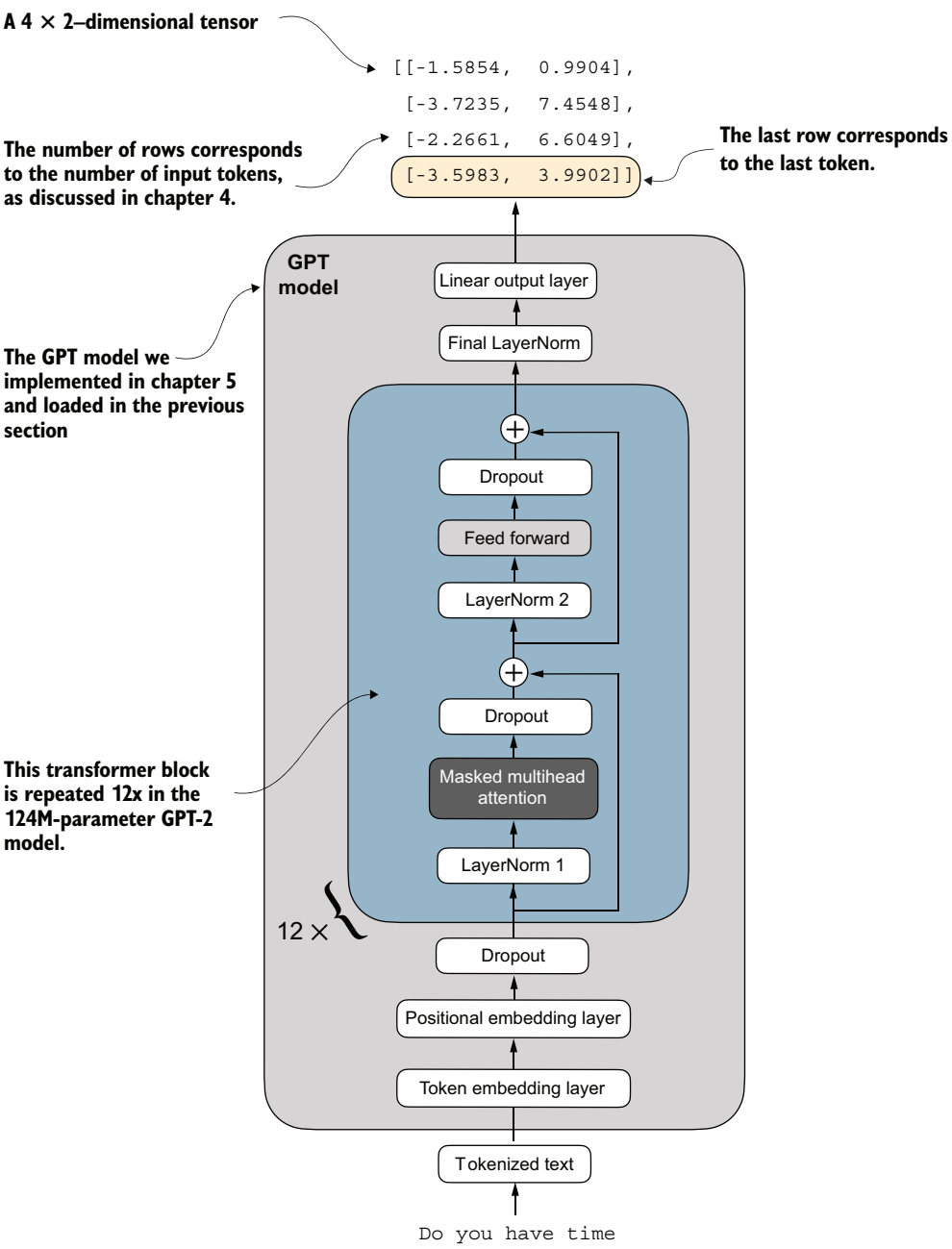

Figure 6.11 The GPT model with a four-token example input and output. The output tensor consists of two columns due to the modified output layer. We are only interested in the last row corresponding to the last token when fine-tuning the model for spam classification.

token's focus to its current position and the those before it, ensuring that each token can only be influenced by itself and the preceding tokens, as illustrated in figure 6.12.

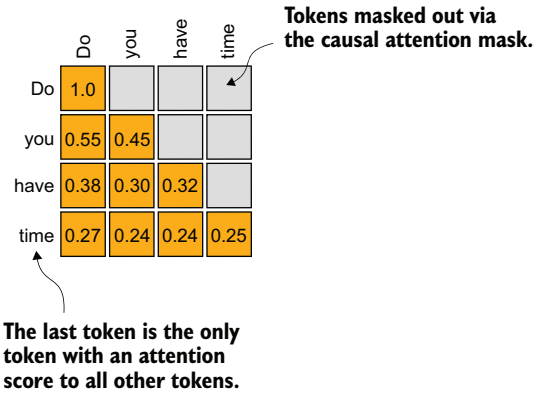

Figure 6.12 The causal attention mechanism, where the attention scores between input tokens are displayed in a matrix format. The empty cells indicate masked positions due to the causal attention mask, preventing tokens from attending to future tokens. The values in the cells represent attention scores; the last token, `time`, is the only one that computes attention scores for all preceding tokens.

Given the causal attention mask setup in figure 6.12, the last token in a sequence accumulates the most information since it is the only token with access to data from all the previous tokens. Therefore, in our spam classification task, we focus on this last token during the fine-tuning process.

We are now ready to transform the last token into class label predictions and calculate the model's initial prediction accuracy. Subsequently, we will fine-tune the model for the spam classification task.

> **Exercise 6.3 Fine-tuning the first vs. last token**
> Try fine-tuning the first output token. Notice the changes in predictive performance compared to fine-tuning the last output token.

6.6 Calculating the classification loss and accuracy

Only one small task remains before we fine-tune the model: we must implement the model evaluation functions used during fine-tuning, as illustrated in figure 6.13.

Before implementing the evaluation utilities, let's briefly discuss how we convert the model outputs into class label predictions. We previously computed the token ID of the next token generated by the LLM by converting the 50,257 outputs into probabilities via the `softmax` function and then returning the position of the highest probability via the `argmax` function. We take the same approach here to calculate whether the model outputs a "spam" or "not spam" prediction for a given input, as shown in figure 6.14. The only difference is that we work with 2-dimensional instead of 50,257-dimensional outputs.

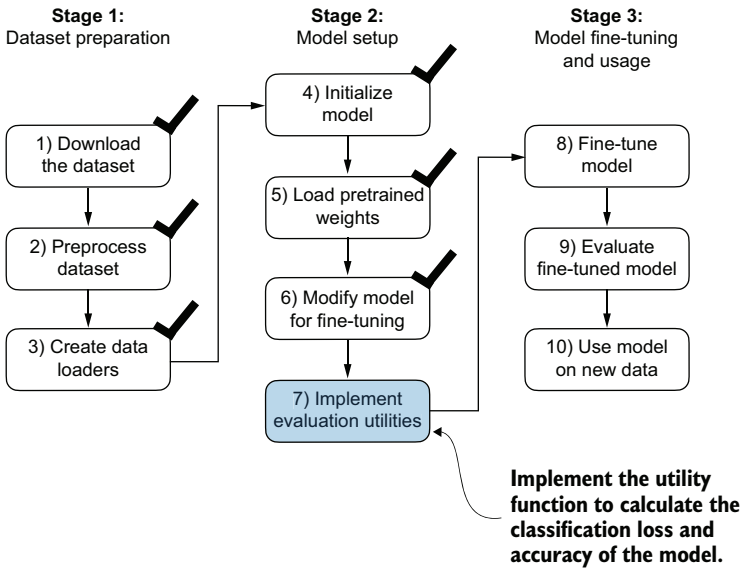

Figure 6.13 The three-stage process for classification fine-tuning the LLM. We've completed the first six steps. We are now ready to undertake the last step of stage 2: implementing the functions to evaluate the model's performance to classify spam messages before, during, and after the fine-tuning.

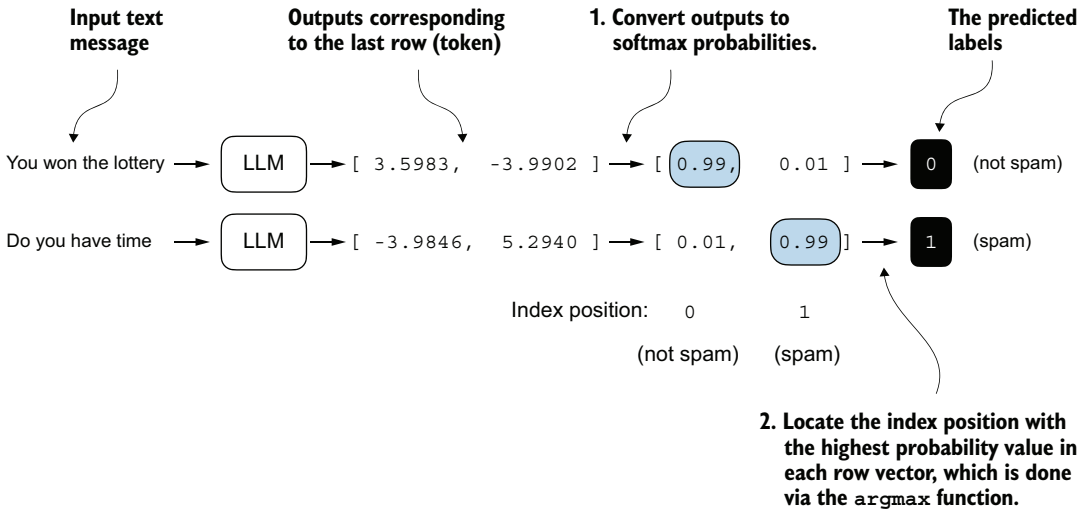

Figure 6.14 The model outputs corresponding to the last token are converted into probability scores for each input text. The class labels are obtained by looking up the index position of the highest probability score. The model predicts the spam labels incorrectly because it has not yet been trained.