

# *contents*

---

<i>preface</i>	<i>xi</i>
<i>acknowledgments</i>	<i>xiii</i>
<i>about this book</i>	<i>xv</i>
<i>about the author</i>	<i>xix</i>
<i>about the cover illustration</i>	<i>xx</i>

<b>1</b>	<b><i>Understanding large language models</i></b>	<b>1</b>
1.1	What is an LLM?	2
1.2	Applications of LLMs	4
1.3	Stages of building and using LLMs	5
1.4	Introducing the transformer architecture	7
1.5	Utilizing large datasets	10
1.6	A closer look at the GPT architecture	12
1.7	Building a large language model	14
<b>2</b>	<b><i>Working with text data</i></b>	<b>17</b>
2.1	Understanding word embeddings	18
2.2	Tokenizing text	21
2.3	Converting tokens into token IDs	24
2.4	Adding special context tokens	29

- 2.5 Byte pair encoding 33
- 2.6 Data sampling with a sliding window 35
- 2.7 Creating token embeddings 41
- 2.8 Encoding word positions 43

### 3 Coding attention mechanisms 50

- 3.1 The problem with modeling long sequences 52
- 3.2 Capturing data dependencies with attention mechanisms 54
- 3.3 Attending to different parts of the input with self-attention 55
  - A simple self-attention mechanism without trainable weights 56*
  - Computing attention weights for all input tokens 61*
- 3.4 Implementing self-attention with trainable weights 64
  - Computing the attention weights step by step 65* ▪ *Implementing a compact self-attention Python class 70*
- 3.5 Hiding future words with causal attention 74
  - Applying a causal attention mask 75* ▪ *Masking additional attention weights with dropout 78* ▪ *Implementing a compact causal attention class 80*
- 3.6 Extending single-head attention to multi-head attention 82
  - Stacking multiple single-head attention layers 82* ▪ *Implementing multi-head attention with weight splits 86*

### 4 Implementing a GPT model from scratch to generate text 92

- 4.1 Coding an LLM architecture 93
- 4.2 Normalizing activations with layer normalization 99
- 4.3 Implementing a feed forward network with GELU activations 105
- 4.4 Adding shortcut connections 109
- 4.5 Connecting attention and linear layers in a transformer block 113
- 4.6 Coding the GPT model 117
- 4.7 Generating text 122

<b>5</b>	<b>Pretraining on unlabeled data</b>	<b>128</b>
5.1	Evaluating generative text models	129
	<i>Using GPT to generate text</i>	130
	<i>Calculating the text generation loss</i>	132
	<i>Calculating the training and validation set losses</i>	140
5.2	Training an LLM	146
5.3	Decoding strategies to control randomness	151
	<i>Temperature scaling</i>	152
	<i>Top-k sampling</i>	155
	<i>Modifying the text generation function</i>	157
5.4	Loading and saving model weights in PyTorch	159
5.5	Loading pretrained weights from OpenAI	160
<b>6</b>	<b>Fine-tuning for classification</b>	<b>169</b>
6.1	Different categories of fine-tuning	170
6.2	Preparing the dataset	172
6.3	Creating data loaders	175
6.4	Initializing a model with pretrained weights	181
6.5	Adding a classification head	183
6.6	Calculating the classification loss and accuracy	190
6.7	Fine-tuning the model on supervised data	195
6.8	Using the LLM as a spam classifier	200
<b>7</b>	<b>Fine-tuning to follow instructions</b>	<b>204</b>
7.1	Introduction to instruction fine-tuning	205
7.2	Preparing a dataset for supervised instruction fine-tuning	207
7.3	Organizing data into training batches	211
7.4	Creating data loaders for an instruction dataset	223
7.5	Loading a pretrained LLM	226
7.6	Fine-tuning the LLM on instruction data	229
7.7	Extracting and saving responses	233
7.8	Evaluating the fine-tuned LLM	238
7.9	Conclusions	247
	<i>What's next?</i>	247
	<i>Staying up to date in a fast-moving field</i>	248
	<i>Final words</i>	248

<i>appendix A</i>	<i>Introduction to PyTorch</i>	251
<i>appendix B</i>	<i>References and further reading</i>	289
<i>appendix C</i>	<i>Exercise solutions</i>	300
<i>appendix D</i>	<i>Adding bells and whistles to the training loop</i>	313
<i>appendix E</i>	<i>Parameter-efficient fine-tuning with LoRA</i>	322
	<i>index</i>	337