

Inhalt

Vorwort	11
Über dieses Buch	13
1 LLMs verstehen	19
1.1 Was ist ein LLM?	20
1.2 Anwendungen von LLMs	22
1.3 Phasen beim Erstellen und Verwenden von LLMs	24
1.4 Einführung in die Transformer-Architektur	26
1.5 Große Datensätze nutzen	30
1.6 Die GPT-Architektur unter der Lupe	31
1.7 Ein großes Sprachmodell aufbauen	34
1.8 Zusammenfassung	35
2 Mit Textdaten arbeiten	37
2.1 Wort-Embeddings	38
2.2 Text tokenisieren	41
2.3 Tokens in Token-IDs konvertieren	45
2.4 Spezielle Kontexttokens hinzufügen	50
2.5 Bytepaar-Codierung	54
2.6 Daten-Sampling mit einem gleitenden Fenster	56
2.7 Token-Embeddings erzeugen	64
2.8 Wortpositionen codieren	67
2.9 Zusammenfassung	72

3	Attention-Mechanismen programmieren	75
3.1	Das Problem beim Modellieren langer Sequenzen	77
3.2	Datenabhängigkeiten mit Attention-Mechanismen erfassen	79
3.3	Verschiedene Teile der Eingabe mit Self-Attention berücksichtigen	81
3.3.1	Ein einfacher Self-Attention-Mechanismus ohne trainierbare Gewichte	81
3.3.2	Attention-Gewichte für alle Eingabetokens berechnen	87
3.4	Self-Attention mit trainierbaren Gewichten implementieren	90
3.4.1	Attention-Gewichte Schritt für Schritt berechnen	91
3.4.2	Eine kompakte Python-Klasse für Self-Attention implementieren	97
3.5	Zukünftige Wörter mit kausaler Attention ausblenden	102
3.5.1	Eine kausale Attention-Maske anwenden	103
3.5.2	Zusätzliche Attention-Gewichte mit Dropout maskieren ...	106
3.5.3	Eine kompakte Klasse für kausale Attention implementieren	109
3.6	Single-Head-Attention zur Multi-Head-Attention erweitern	111
3.6.1	Mehrere Single-Head-Attention-Schichten stapeln	112
3.6.2	Multi-Head-Attention mit Gewichtsteilungen implementieren	115
3.7	Zusammenfassung	121
4	Ein GPT-Modell von Grund auf neu erstellen, um Text zu generieren	123
4.1	Eine LLM-Architektur programmieren	124
4.2	Aktivierungen mit Schichtnormalisierung normalisieren	131
4.3	Ein Feedforward-Netz mit GELU-Aktivierungen implementieren ...	138
4.4	Shortcut-Verbindungen hinzufügen	142
4.5	Attention und lineare Schichten in einem Transformer-Block verbinden	147
4.6	Das GPT-Modell programmieren	151
4.7	Text generieren	157
4.8	Zusammenfassung	163

5	Vortraining mit ungelabelten Daten	165
5.1	Generative Textmodelle bewerten	166
5.1.1	Text mithilfe von GPT erzeugen	167
5.1.2	Den Texterzeugungsverlust berechnen	170
5.1.3	Die Verluste der Trainings- und Validierungsdatensätze berechnen	178
5.2	Ein LLM trainieren	185
5.3	Decodierungsstrategien, um Zufälligkeit zu steuern	192
5.3.1	Temperaturskalierung	193
5.3.2	Top-k-Sampling	196
5.3.3	Die Funktion zur Textgenerierung modifizieren	199
5.4	Modellgewichte in PyTorch laden und speichern	201
5.5	Vortrainierte Gewichte von OpenAI laden	203
5.6	Zusammenfassung	210
6	Feintuning zur Klassifizierung	213
6.1	Verschiedene Kategorien des Feintunings	214
6.2	Den Datensatz vorbereiten	216
6.3	DataLoader erstellen	220
6.4	Ein Modell mit vortrainierten Gewichten initialisieren	227
6.5	Einen Klassifizierungskopf hinzufügen	229
6.6	Klassifizierungsverlust und -genauigkeit berechnen	237
6.7	Das Modell mit überwachten Daten feintunen	242
6.8	Das LLM als Spam-Klassifizierer verwenden	248
6.9	Zusammenfassung	251
7	Feintuning, um Anweisungen zu befolgen	253
7.1	Einführung in die Anweisungsoptimierung	254
7.2	Einen Datensatz für die Anweisungsoptimierung vorbereiten	256
7.3	Daten in Trainingsstapeln organisieren	260
7.4	DataLoader für einen Anweisungsdatensatz erstellen	274
7.5	Ein vortrainiertes LLM laden	277
7.6	Das LLM mit Anweisungsdaten feintunen	281

7.7	Antworten extrahieren und speichern	286
7.8	Das feingetunte LLM bewerten	291
7.9	Fazit	301
7.9.1	Was kommt als Nächstes?	302
7.9.2	In einem sich schnell entwickelnden Bereich auf dem neuesten Stand bleiben	302
7.9.3	Ein paar Worte zum Schluss	303
7.10	Zusammenfassung	303
A	Einführung in PyTorch	305
A.1	Was ist PyTorch?	305
A.1.1	Die drei Kernkomponenten von PyTorch	306
A.1.2	Deep Learning definieren	307
A.1.3	PyTorch installieren	309
A.2	Tensoren	313
A.2.1	Skalare, Vektoren, Matrizen und Tensoren	314
A.2.2	Tensor-Datentypen	314
A.2.3	Allgemeine PyTorch-Tensor-Operationen	315
A.3	Modelle als Berechnungsgraphen sehen	317
A.4	Automatisches Differenzieren leicht gemacht	319
A.4.1	Partielle Ableitungen und Gradienten	320
A.5	Mehrschichtige neuronale Netze implementieren	321
A.6	Effiziente DataLoader einrichten	327
A.7	Eine typische Trainingsschleife	333
A.8	Modelle speichern und laden	338
A.9	Die Trainingsperformance mit GPUs optimieren	338
A.9.1	PyTorch-Berechnungen auf GPU-Geräten	338
A.9.2	Training auf einer einzelnen GPU	340
A.9.3	Training mit mehreren GPUs	342
A.10	Zusammenfassung	349
B	Referenzen und weiterführende Literatur	351
C	Lösungen zu den Übungen	365

D	Die Trainingsschleife mit allem Drum und Dran	381
D.1	Aufwärmen der Lernrate	383
D.2	Cosinus-Decay	385
D.3	Gradienten-Clipping	386
D.4	Die modifizierte Trainingsfunktion	388
E	Parametereffizientes Feintuning mit LoRA	393
E.1	Einführung in LoRA	393
E.2	Den Datensatz vorbereiten	395
E.3	Das Modell initialisieren	398
E.4	Parametereffizientes Feintuning mit LoRA	400
	Index	409