instruction fine-tuning 7, 170, 322
  instruction following, creating data loaders for instruction dataset 224–226
  'instruction' object 208
  instruction–response pairs 207
  loading pretrained LLMs 226–229
  overview 205

K

keepdim parameter 101

L

LayerNorm 103, 115, 117, 119
layer normalization 99–105
learning rate warmup 313–314
__len__ method 271
LIMA dataset 296
Linear layers 95, 107, 329–330, 332–333
Linear layer weights 330
LinearWithLoRA layer 330–331, 333
LitGPT 249
LLama 2 model 141
Llama 3 model 238
llama.cpp library 238
LLMs (large language models) 17–18
  applications of 4
  building and using 5–7, 14
  coding architecture 93–99
  coding attention mechanisms, causal attention mechanism 74–82
  fine-tuning 230–233, 238–247, 295
  fine-tuning for classification 183–194, 200
  implementing GPT model, implementing feed forward network with GELU activations 105–109
  instruction fine-tuning, loading pretrained LLMs 226–229
  overview of 1–4
  pretraining 132, 140, 142, 146–151, 159
  training function 313, 319–321
  training loop, gradient clipping 317
  transformer architecture 7–10
  utilizing large datasets 10
  working with text data, word embeddings 18–20
loading, pretrained weights from OpenAI 160–167
load_state_dict method 160
load_weights_into_gpt function 165–166, 182
logistic regression loss function 293
logits tensor 139
LoRALayer class 329–330
LoRA (low-rank adaptation) 247, 322
  parameter-efficient fine-tuning 324, 326
loss.backward() function 112
losses 140, 142
loss metric 132
lr (learning rate) 275

M

machine learning 253
Machine Learning Q and AI (Raschka) 290
macOS 282
main function 286
masked attention 74
.matmul method 261
matrices 258–261
max_length 38, 141, 178, 306
minbpe repository 291
model_configs table 164
model.eval() function 160
model.named_parameters() function 112
model.parameters() method 129
model_response 238
model.train() setting 276
model weights, loading and saving in PyTorch 159
Module base class 265
mps device 224
mp.spawn() call 286
multi-head attention 80, 82–91
  implementing with weight splits 86–91
  stacking multiple single-head attention layers 82–85
MultiHeadAttention class 86–87, 90–91, 292
MultiHeadAttentionWrapper class 83–87, 90
multilayer neural networks, implementing 265–269
multinomial function 153–155
multiprocessing.spawn function 284
multiprocessing submodule 284

N

NeuralNetwork model 284
neural networks
  implementing feed forward network with GELU activations 105–109
  implementing multilayer neural networks 265–269
NEW_CONFIG dictionary 164
n_heads 95

Licensed to Jason Walsh <manning@jwalsh.net>

nn.Linear layers 72
nn.Module 71, 97
numel() method 120
num_heads dimension 88
num_tokens dimension 88

O

Ollama application 238, 241
Ollama Llama 3 method 309
ollama run command 242
ollama run llama3 command 240–241
ollama serve command 239–242
OLMo 294
one-dimensional tensor (vector) 259
OpenAI, loading pretrained weights from 160–167
OpenAI's GPT-3 Language Model: A Technical Overview 293
optimizer.step() method 276
optimizer.zero_grad() method 276
out_head 97
output layer nodes 183
'output' object 208

P

parameter-efficient fine-tuning 322
  LoRA (low-rank adaptation) 322
  preparing dataset 324
parameters 129
  calculating 302
params dictionary 162, 164–165
partial derivatives 263
partial function 224
peak_lr 314
perplexity 139
Phi-3 model 297
PHUDGE model 297
pip installer 33
plot_losses function 232
plot_values function 199
pos_embeddings 47
Post-LayerNorm 115
preference fine-tuning 298
Pre-LayerNorm 115
pretokenizes 212
pretrained weights, initializing model with 181
pretraining 7
  calculating text generation loss 132
  calculating training and validation set losses 140, 142
  decoding strategies to control randomness 151–159
  loading and saving model weights in PyTorch 159
  loading pretrained weights from OpenAI 160–167
  on unlabeled data 128
  training LLMs 146–151
  using GPT to generate text 130
print_gradients function 112
print_sampled_tokens function 155, 304
print statement 24
Prometheus model 297
prompt styles 209
.pth extension 159
Python version 254
PyTorch
  and Torch 256
  automatic differentiation 263–265
  computation graphs 261
  data loaders 210
  dataset objects 325
  efficient data loaders 270–274
  implementing multilayer neural networks 265–269
  installing 254–257
  loading and saving model weights in 159
  optimizing training performance with GPUs 279–288
  overview 251–257
  saving and loading models 278
  training loops 274–278
  understanding tensors 258–261
  with a NumPy-like API 258

Q

qkv_bias 95
Q query matrix 88
query_llama function 243
query_model function 242–243

R

random_split function 175
rank argument 286
raw text 6
register_buffer 81
re library 22
ReLU (rectified linear unit) 100, 105
.replace() method 235
replace_linear_with_lora function 330, 332

Licensed to Jason Walsh <manning@jwalsh.net>

.reshape method 260–261
re.split command 22
responses, extracting and saving 233–238
retrieval-augmented generation 19
r/LocalLLaMA subreddit 248
RMSNorm 292
RNNs (recurrent neural networks) 52

S

saving and loading models 278
scalars 258–261
scaled dot-product attention 64
scaled_dot_product function 292
scale parameter 103
sci_mode parameter 102
SelfAttention class 90
self-attention mechanism 55–64
  computing attention weights for all input tokens 61–64
  implementing with trainable weights 64–74
  without trainable weights 56–61
SelfAttention_v1 class 71, 73
SelfAttention_v2 class 73
self.out_proj layer 90
self.register_buffer() call 81
self.use_shortcut attribute 111
Sequential class 267
set_printoptions method 277
settings dictionary 162, 164
SGD (stochastic gradient descent) 275
.shape attribute 260, 271
shift parameter 103
shortcut connections 109–113
SimpleTokenizerV1 class 27
SimpleTokenizerV2 class 29, 31, 33
single-head attention, stacking multiple layers 82–85
sliding window 35–41
softmax function 269, 276
softmax_naive function 60
SpamDataset class 176, 178
spawn function 286
special context tokens 29–32
state_dict 160, 279
stride setting 39
strip() function 229
supervised data, fine-tuning model on 195–200
supervised instruction fine-tuning 205
  preparing dataset for 207–211
supervised learning 253
SwiGLU (Swish-gated linear unit) 105

T

target_chunk tensor 38
targets tensor 139
temperature scaling 151–152, 154–155
tensor2d 259
tensor3d 259
Tensor class 258
tensor library 252
tensors 258–261
  common tensor operations 260
  scalars, vectors, matrices, and tensors 258–261
  tensor data types 259
  three-dimensional tensor 259
  two-dimensional tensor 259
test_data set 246
test_loader 272
test_set dictionary 237–238
text completion 205
text data 17
  adding special context tokens 29–32
  converting tokens into token IDs 24–29
  creating token embeddings 42–43
  encoding word positions 43–47
  sliding window 35–41
  tokenization, byte pair encoding 33–35
  word embeddings 18–20
text_data 314
text generation 122
  using GPT to generate text 130
text generation function, modifying 157
text generation loss 132
text_to_token_ids function 131
tiktoken package 176, 178
.T method 261
.to() method 259, 280
token_embedding_layer 46–47
token embeddings 42–43
token IDs 24–29
token_ids_to_text function 131
tokenization, byte pair encoding 33–35
tokenizing text 21–24
top-k sampling 151, 155–156
torch.argmax function 125
torchaudio library 255
torch.manual_seed(123) 272
torch.nn.Linear layers 267
torch.no_grad() context manager 269
torch.save function 159
torch.sum method 277
torch.tensor function 258
torchvision library 255

Licensed to Jason Walsh <manning@jwalsh.net>