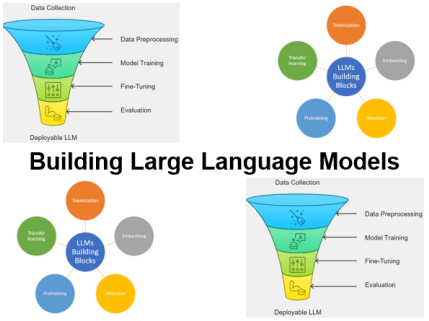
Large Language Models Bouwen

Duur: 4 dagen **Prijs: € 3200**

Open Rooster



Doelgroep Cursus Large Language Models Bouwen

De cursus Large Language Models Bouwen is bedoeld voor engineers die transformer-based LLM's willen ontwerpen.

Voorkennis Cursus Large Language Models Bouwen

Deelnemers dienen vertrouwd te zijn met Python. Ervaring met PyTorch of een vergelijkbaar Deep Learning-framework is een plus.

Uitvoering Training Large Language Models Bouwen

De training combineert bondige theorie met begeleide, hands-on labs. Via code-alongs bouw je een mini-GPT, bereid je datasets voor, voer je pretraining en fine-tuning uit en deploy je modellen.

Certificaat Large Language Models Bouwen

Na afloop ontvangen deelnemers een certificaat van deelname aan de cursus Large Language Models Bouwen.

Inhoud Large Language Models Bouwen

De cursus Large Language Models Bouwen van SpiralTrain leert je hoe je met PyTorch en modern tooling transformer-based LLMs ontwerpt, traint, fine-tuned en deployed. Je leert robuuste text pipelines en tokenizers op te zetten, een GPT-style model met pretraining te implementeren, instruction tuning en RAG toe te passen, en modellen in production op te leveren.

Intro LLM

De cursus Large Language Models bouwen start met wat LLM's zijn, waar ze worden ingezet en gaat in op de lifecycle van bouwen versus gebruiken. De Transformer/GPT-architectuur wordt geintroduceerd evenals hoe modellen leren van grote datasets en wanneer je klassieke QA gebruikt versus RAG.

Werken met Tekstdata

Je leert ruwe tekst om te zetten naar modelklare tensors dor middel van tokenization token→ID mapping, speciale/contexttokens en sliding-window sampling. Embeddings en positional encodings worden behandeld en hoe om te gaan om met onbekende woorden en een basis zinsstructuur.

Attention-mechanisme

Deze module demystificeert self-attention voor long-sequence modeling: queries, keys, values en causal masking om toekomstige tokens te verbergen. Positional encoding, multi-head attention en gestapelde lagen worden toegvoegd toe om afhankelijkheden in de input op te vangen.

PyTorch Deep Learning

Deze module behandelt PyTorch-basics—tensors en training loops evenals tooling om model kwaliteit te meten. Ingegaan wordt op feature scaling/normalization, activatie- en loss-functies en backpropagation.

Neurale Netwerken

Vervolgens bouw je MLP's en CNN's in PyTorch, kies je passende activaties en losses en implementeer je backpropagation. Ook NLP-specifieke preprocessing wordt behandeld evenals end-to-end binaire en multiclass-classificatie.

GPT van scratch

Daarna implementeer je een minimale GPT met layer normalization, residual (skip) connections en attention + feed-forward (GELU) blocks met weight tying. Je koppelt transformer blocks en genereert tekst om het model in actie te zien.

Pretraining

Je pretraint de LLM op ongelabelde tekst met next-token prediction en volgt training- versus validation-losses. Je verkent decoding strategies (bijv. temperature, top-k), beheerst randomness voor reproduceerbaarheid en slaat PyTorch-weights op/laadt ze in.

Tuning voor Classificatie

We bereiden datasets en dataloaders voor, initialiseren vanuit pretrained weights en voegen een classification head met softmax toe. Je traint en evalueert met loss/accuracy, met als voorbeeld een LLM-based spam-classification.

Modules Large Language Models Bouwen

Module 1 : LLM Intro	Module 2 : Working with Text Data	Module 3 : Attentions Mechanism
What is an LLM?	Word Embeddings	Modeling Long Sequences
Applications of LLMs	Decoders and Encoders	Capturing Data Dependencies
Stages of Building LLMs	Decoder Only Transformer	Attention Mechanisms
Stages of Using LLMs	Tokenizing text	Attending Different Input Parts
Transformer Architecture	Convert Tokens into IDs	Using Self-Attention
Utilizing Large Datasets	Special Context Tokens	Trainable Weights
GPT Architecture Internals	Understand Sentence Structure	Hiding Future Words
Learn Language Patterns	Byte Pair Encoding	Positional Encoding
Retrieval Augmented Generation	Unknown Words	Causal Attention
Question and Answer Systems	Sampling with Sliding Window	Masking Weights with Dropout
QA versus RAG	Creating Token Embeddings	Multihead Attention
Building an LLM	Encoding Word Positions	Stacking Attentions Layers
Module 4 : Pytorch Deep Learning	**Module 5 : Neural Networks**	**Module 6 : GPT from scratch**
Deep Learning Intro	Neural Networks Intro	Coding an LLM Architecture
Overview of PyTorch	Building NN with PyTorch	Layer Normalization
PyTorch Tensors	Multiple Layers of Arrays	Normalizing Activations
Tensor Operations	Convolutional Neural Networks	Feed Forward Network
Model Evaluation Metrics	Activation Functions	GELU Activations
Feature Scaling	Loss Functions	Adding Shortcut Connections
Feature Normalization	Backpropagation	Connecting Attention
Categorical Features	Natural Language Processing	Weight Tying
Activation Functions	Stopword Removal	Linear Layers in Transformer Block
Loss Functions	Binary Classification	Coding the GPT Model
Backpropagation	Multi-class Classification	Generating Text
Module 7 : Pretraining	**Module 8 : Tuning for Classification**	**Module 9 : Fine-Tuning**
Pretraining on Unlabeled Data	Categories of Fine-Tuning	Instruction Fine-tuning
Calculating Text Generation Loss	Preparing the Dataset	Supervised Instruction
Training Losses	Creating Data Loaders	Preparing a Dataset
Validation Set Losses	Top-k Sampling	Organizing Training Batches
Training an LLM	Soft-Max Function	Creating Data Loaders
Decoding Strategies	Initializing with Pretrained Weights	Loading a pretrained LLM
Control Randomness	Adding Classification Head	Fine-tuning the LLM
Temperature Scaling	Classification Loss and Accuracy	Extracting and Saving Responses
Saving Model Weights in PyTorch	Fine-tuning on Supervised Data	Evaluating Fine-tuned LLM
Loading Pretrained Weights	Using LLM as Spam Classifier	Fine Tuning with LoRA