Creating a Large Language Model (LLM) in Python

Course code: PYTHON_LLM

This course is designed for programmers who want to understand how large language models (LLMs) work "inside" and work their way practically from simple statistical language models to RNN/LSTMs to transformers and mini GPTs. In the second part of the course, participants will learn how to work with ready-made models (Hugging Face), perform fine-tuning (including LoRA) and build a practical application on top of custom documents using RAG (retrieval-augmented generation). Production aspects are also included: latency, optimization, quantization and deployment as an API including Docker.

Who the course is for:

- Programmers with a basic knowledge of Python who want to understand LLM and be able to use it practically.
- Developers who want to implement and train small models, and then work with existing LLMs (fine-tuning, RAG, deployment).
- Data/ML enthusiasts who want to get a comprehensive practical overview from "zero" to application.

Required entry-level skills

- Basic knowledge of Python at the PYTHON_INTRO course level.
- Basic knowledge of NumPy at the level of the PYTHON_DATAN course
- Basic linear algebra is an advantage (not a requirement)

Teaching methods

- Expert lecture with practical demonstrations, exercises on computers.
- Work in Jupyter Notebook and scripts, continuous mini-projects.
- Emphasis on understanding of principles and reproducibility.

Study materials

- Presentation of material covered in print or online form.
- Notebooks and reference implementations for each day.
- Sample datasets and templates for training, evaluation and deployment.

Curriculum

Fundamentals of neural networks and NLP

- What is a language model
- Probability and prediction of the next word/token
- Tokenization
- Embeddings
- Neural network (perceptron, layers, activation)
- Backpropagation (intuition)
- Python + NumPy
- Unigram model implementation
- Bigram model implementation
- Small network training in PyTorch
- Output of the day: small language model predicting the next word on small text patterns Recurrent networks and first text generation
- RNN
- LSTM
- The vanishing gradient problem
- Teacher forcing
- Sampling (temperature, top-k)
- LSTM model implementation in PyTorch

GOPAS Praha
Na Strži 2097/63
140 00 Praha 4 - Krč
Tel.: +420 226 201 390
info@gopas.cz

GOPAS Brno
Nové sady 996/25
602 00 Brno
Tel.: +420 530 513 590
info@gopas.cz

GOPAS Bratislava
Dr. Vladimíra Clementisa 10
Bratislava, 821 02
Tel.: +421 902 903 132
info@gopas.sk

- Training on a small dataset (e.g. Shakespeare)
- Text generation
- Output of the day: short text generation model Transformer architecture
- Attention mechanism
- Self-attention
- Multi-head attention
- Positional encoding
- Encoder vs Decoder
- Why the transformer is scalable
- Mini-transformer implementation
- Creating a small GPT-like model
- Training on a small dataset
- Output of the day: a working mini GPT model Training, fine-tuning and working with finished models
- Pretraining vs Fine-tuning
- Transfer learning
- LoRA and parameter-efficient fine-tuning (PEFT)
- Tokenizers (BPE)
- Using the Hugging Face Transformers library
- Small Model Fine-tuning
- Working with models (e.g. LLaMA / compatible open-source models as available)
- Creating a custom chatbot script
- Output of the day: fine-tuned model on custom data RAG, deployment and production aspects
- Embeddings for search
- Vector databases (FAISS)
- RAG architecture
- Latency and optimization
- Quantization of models
- Deployment (API, Docker)
- Embedding generation
- Storage in FAISS
- RAG pipeline implementation
- Creating a simple API (FastAPI)
- Final project: internal chatbot over custom documents

Technologies used

- Python 3.11+
- PyTorch
- Hugging Face Transformers
- FAISS
- FastAPI
- Jupyter Notebook

What participants will take away

- Understand how LLMs work internally and what their building blocks are
- Can build a simple transformer and mini GPT on a small dataset
- Can fine-tune a model including LoRA/PEFT
- Can build a RAG application on top of custom documents
- Can deploy the model as an API and understands basic production aspects (latency, optimization, quantization)

GOPAS Praha
Na Strži 2097/63
140 00 Praha 4 - Krč
Tel.: +420 226 201 390
info@gopas.cz

GOPAS Brno
Nové sady 996/25
602 00 Brno
Tel.: +420 530 513 590
info@gopas.cz

GOPAS Bratislava
Dr. Vladimíra Clementisa 10
Bratislava, 821 02
Tel.: +421 902 903 132
info@gopas.sk