AI-413: Open-source LLM Application Developer

Planned Course Length: 24 training hours

Course Description:

Artificial intelligence has become an extremely important area for IT professionals and engineers in the past 10-20 years with the scientific breakthroughs and practical applications of deep learning and more recently of generative AI systems, especially with its Large Language Model (LLM) variant such as OpenAI's ChatGPT and Google's Bard and many open-source models. Due to its importance and impact on every aspect of our lives, understanding the concepts, functionalities and practical usage of AI systems is quickly becoming essential for all IT and other technical professionals as well as for managers with technical background.

This training focuses on LLM concepts as well as open-source LLM Prompt Engineering and Application Development and teaches participants the following topics:

- Introduction to LLM based applications
- The Foundations: Neural Networks, Deep Learning, CNN, RNN, Transfer Learning
- "Attention is all you need" – The Transformer Architecture
- The 3-phase training process of LLMs (pre-training, fine-tuning, RLHF)
- Basics of prompt engineering
- Advanced prompt engineering techniques
- Fine-tuning Open-source LLM models
- Retriever Augmented Generation (RAG)
- Creating LLM chains with LangChain
- Fast Web Interface Prototyping for LLMs (Gradio or Streamlit)
- Debugging and Evaluating LLM-based apps (Weights & Biases and Vellum)
- Basics of AI Threats and LLM Security (optional)

Besides gaining a basic understanding of the theory of Large Language Models (LLMs) models, students will also make extensive lab exercises using the Python based PyTorch deep learning framework and the Hugging Face ecosystem.

This training is part of the AI portfolio of Component Soft which explores essential AI topics, such as:

- AI-110 Intro to Large Language Model (LLMs) and LLM-based apps.
- AI-202 Deep Learning Foundations with PyTorch
- AI-413: Open-source LLM Application Developer
- AI-423: ChatGPT/GPT4 Application Developer

Structure: 50% theory, 50% hands on lab exercises

Target audience: Software developers and all types of IT and telecom professionals who want to understand the basic concepts of Large Language Models (LLMs) and want to develop LLM-based applications.

Prerequisites: Basic understanding of AI, Machine Learning and Deep Learning concepts. Basic Python programming skills. Some experience in using The Python based PyTorch deep-learning frameworks is an advantage.

Suggested preliminary course: AI-202 Deep Learning Basics with PyTorch

Detailed Course Outline

PART I. Basic LLM Concepts

Module 1. Introduction to LLM based applications

- Main usage areas of LLM-based applications
- Main types of LLM-based applications
- Building blocks of LLLM-based applications
- Lab: Testing a simple LLM-based application

Module 2. The Foundations: Neural Networks, Deep Learning, CNN, RNN, Transfer Learning

- From human neural cells to artificial neural networks
- Deep Neural Networks
- More advanced structures: Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs)
- Transfer Learning
- Lab: Playing with a small neural network written in Python and PyTorch

Module 3. “Attention is all you need” – The Transformer Architecture

- Intuition of the transformer model
- Main elements of transformers: tokenizer, embeddings, encoder, decoder
- Variations on the transformer architecture
- Popular transformer models
- Lab: Testing popular LLM foundational models

Module 4. The 3-phase training process of LLMs (pre-training, fine-tuning, RLHF)

- **Pre-training of LLMs**
- How does pre-training basically work?
- Training data set, computational and financial challenges
- **LLM Fine-tuning techniques.**
- How does fine-tuning basically work?
- Parameter efficient fine-tuning (PEFT) with LoRA and quantized parameters
- **Reinforcement Learning with Human feedback (RLHF)**
- Why do we need RLHF in the first place?
- Methods and main steps of RLHF
- Lab: Examining an LLM family before and after fine-tuning and RLHF

PART II. Prompt Engineering for App. Developers

Module 5. Basics of prompt engineering

- What is prompt engineering?
- Prompt engineering terminology and concepts
- The “Just Ask” Principle, Zero-shot prompts
- Prompts with Few-shot prompting

- Prompt Chaining
- Chain of Thought Prompting
- Prompts with Personas
- Lab: Practicing basic prompt techniques via Python APIs in Jupyter notebooks

Module 6. Advanced prompt engineering techniques

- Techniques related to Prompt Structure and Clarity
- Techniques related to Specificity and Information
- Techniques related to User Interaction and Engagement
- Techniques related to Content and Language Style
- Techniques related to Complex Tasks and Coding Prompts
- Lab: Using advanced prompt engineering techniques via Python APIs in Jupyter notebooks

PART III. Application Development with LLMs

Module 7. Fine-tuning Open-source LLM models

- Typical goals of fine-tuning LLMs
- Difference between prompt engineering and fine tuning
- When to use which?
- Parts of the Training Process
- The HuggingFace trainer API
- Preparing your dataset
- Creating a fine-tuned model
- Evaluating a fine-tuned model
- Lab: Fine-tuning an open-source LLM

Module 8. Retriever Augmented Generation (RAG)

- What is Retriever Augmented Generation (RAG)?
- How does RAG work?
- Syntactic vs. Semantic Similarity
- Text embedding
- Vector Databases
- Lab: Using Retriever Augmented Generation (RAG) in an LLM app

Module 9. Creating LLM chains with LangChain

- What are LLM chains?
- LangChain architecture
- Main Building Blocks: Models, Prompts and Output Parsers
- Building LLM chains from building blocks
- LangChain Memory
- LangChain Agents
- Lab: Using LangChain in an open-source LLM app.

Module 10. Fast Web Interface Prototyping for LLMs (Gradio or Streamlit)

- Main features of LLM Web interfaces
- Example web app for simple LLM tasks
- Creating our own chatbot with a web interface
- Lab: Creating a simple LLM web interface with Gradio or Streamlit

Module 11. Debugging and Evaluating LLM-based apps (Weights & Biases and Vellum)

- Main techniques of debugging LLMs
- Features of Weights & Biases debugging and evaluation tools
- Using Weights & Biases tools with Pytorch and LLMs
- Challenges of evaluating LLMs
- Evaluating the quality of prompts and answers with different LLMs
- Testing semantic searches
- Testing versioning and monitoring LLM deployments
- Prototyping, deploying, versioning, and monitoring LLM chains
- Explicit and implicit human evaluation of LLMs
- Lab: Example usage of Weights and Biases and Vellum.ai Tools for LLM debugging and testing

Module 12. Basics of AI Threats and LLM Security (optional)

- Basics of AI Threats and LLM Security
- Top 10 AI & LLM Threats and ways of protection against them:
- 1. Prompt Injection
- 2. Insecure Output Handling
- 3. Training Data Poisoning
- 4. Model Denial of Service
- 5. Supply Chain Vulnerabilities
- 6. Sensitive Information Disclosure
- 7. Insecure Plugin Design
- 8. Excessive Agency
- 9. Overreliance
- 10- Model Theft
- Lab: Examples of LLM Threats and Security hardenings
- Length: 4 hours