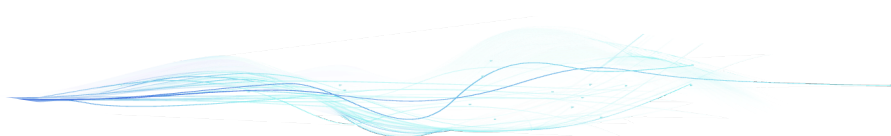




CONTECO

Workshop



**LLM-Integration:
LangChain, Haystack
& Co**

LLM-Integration: LangChain, Haystack & Co

Kursdauer

5 Tage

Zielgruppe

Softwareentwickler:innen, KI-Praktiker:innen und Tech Leads, die bestehende Large Language Models (LLMs) sinnvoll in eigene Anwendungen integrieren möchten – mit Fokus auf Retrieval-Augmented Generation (RAG), Prompt Engineering, sicheren LLM-Workflows und schnellem Prototyping. Der Kurs richtet sich an alle, die LLMs nicht selbst trainieren, sondern produktiv nutzen wollen.

Schulungsziel

Nach Abschluss dieser Schulung sind die Teilnehmenden in der Lage, bestehende LLMs wie GPT, Claude oder LLaMA über APIs und Open-Source-Frameworks in ihre Softwarelösungen zu integrieren. Sie lernen, strukturierte Prompts zu erstellen, Retrieval-Techniken für kontextbasierte Antworten zu verwenden und LLM-Features mit Tools wie LangChain, Haystack, Gradio oder Streamlit umzusetzen. Sicherheitsaspekte wie Prompt Injection und datenschutzkonforme Architekturentscheidungen werden genauso behandelt wie konkrete Anwendungsszenarien (z. B. Chatbots, Developer Assistants, Wissensdatenbanken). Am Ende steht ein funktionsfähiger LLM-Prototyp inklusive RAG-Integration.

Schulungsbeschreibung

Diese fünftägige Schulung bietet einen tiefgehenden, praxisorientierten Einstieg in die Integration von LLMs in moderne Softwareprojekte. Im Mittelpunkt steht die pragmatische Nutzung vorhandener Modelle über APIs, Open-Source-Bibliotheken und No-Code-Plattformen.

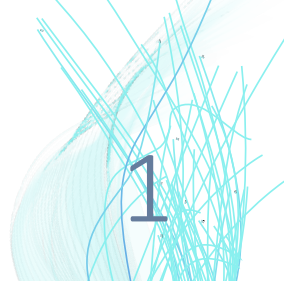
Der erste Tag vermittelt ein technisches Verständnis für LLMs, typische Fehlerquellen und Einschränkungen. Danach werden die Grundlagen und Strategien des Prompt Engineerings erarbeitet, um systematisch strukturierte Prompts für verschiedenste Aufgaben zu gestalten – von Codegenerierung über Refactoring bis hin zu Datenextraktion.

An den Tagen drei und vier stehen LangChain, Haystack und RAG im Fokus: Wie lassen sich LLMs über Dokumente mit Vektorsuche kombinieren? Welche Datenformate sind nötig? Welche Rolle spielen Chroma, FAISS, Pinecone? Parallel wird demonstriert, wie man mit Gradio und Streamlit in kürzester Zeit interaktive Chatinterfaces oder Agenten-Tools erstellt – auch multimodal (Text, Bild, Sprache).

Am fünften Tag entwerfen und präsentieren die Teilnehmenden ein eigenes LLM-Projekt – z. B. ein Developer Assistant, ein interner FAQ-Chatbot oder eine Dokumenten-Suchmaschine.

Wer teilnehmen sollte

Die Schulung richtet sich an Entwickler:innen, die moderne LLMs produktiv einsetzen möchten – sei es in Form von Chatbots, Automatisierung oder Retrieval-Systemen. Vorausgesetzt werden solide Python-Kenntnisse und Interesse an APIs, JSON, Datenverarbeitung und Machine Learning-Technologien. Erfahrungen mit Webframeworks (Flask, FastAPI) oder Tools wie Docker sind hilfreich, aber nicht zwingend.



Tag 1: LLM-Grundlagen & Herausforderungen

- 1. Einordnung von LLMs in der Softwareentwicklung**
 - Grenzen und Stärken von LLMs
 - Typische Probleme: Halluzination, Bias, Kontextbeschränkung, API-Limits
- 2. Technische Grundlagen von LLMs**
 - Transformer, GPT-Architektur, Training vs. Fine-Tuning
 - Unterschied Closed vs. Open-Source-Modelle
- 3. Einsatzszenarien**
 - Codegen, Dokumentation, Konversationsschnittstellen
 - Unterschiede zwischen LLM, Agent und RAG
- 4. Hands-on: Quickstart mit LLM**
 - Ollama oder LM Studio lokal installieren
 - Zugriff auf OpenAI API, Prompt Playground
 - Einfache Prompt-Session über ein Notebook

Tag 2: Prompt Engineering & Prompting-Frameworks

- 1. Grundlagen strukturierter Prompts**
 - Prompt-Struktur: System, Instruction, Output
 - Prompt-Typen: Zero-/One-/Few-Shot, Chain-of-Thought, ReAct
- 2. Best Practices im Prompt Engineering**
 - Rollen, Stil, Formatierung, Constraints, Delimiter
 - Debugging & Prompt-Dokumentation
- 3. Praxis-Prompts für Developer**
 - Code erklären, kommentieren, generieren, validieren
 - Datei- und Ordnerstruktur via Prompt beschreiben lassen
- 4. Hands-on: Prompt-Katalog erstellen**
 - Prompts für Tests, Reviews, Bugfixing
 - Playground-Parameter (Temperature, Top-P, Stop Sequences)

Tag 3: LangChain, Haystack & Vektordatenbanken

- 1. Retrieval-Augmented Generation (RAG)**
 - Architektur von RAG: Pipeline, Query, Context, Antwort
 - Vorteile: längerer Kontext, weniger Halluzinationen
- 2. Vektordatenbanken & Embeddings**
 - Chroma, FAISS, Pinecone
 - Embedding-Modelle (OpenAI, HuggingFace Transformers)
- 3. Frameworks im Fokus**
 - LangChain: Chain Types, Agents, Memory, Tools
 - Haystack: Pipeline-Builder, Nodes, Dokumentenspeicher
- 4. Hands-on**
 - Indexierung eigener Dokumente mit Chroma
 - Einbindung in LangChain-Pipeline mit GPT-API
 - Aufbau eines RAG-Chatbots (lokal oder über Colab)



Tag 4: Multimodalität, Sicherheit & Integration

- 1. Streamlit & Gradio für Rapid Prototyping**
 - Chatinterfaces mit Python bauen
 - Integration von Audio (Whisper) und Bild (CLIP, Stable Diffusion)
- 2. Prompt Injection & Jailbreaking**
 - Wie funktionieren Prompt-Angriffe?
 - Abwehrstrategien und Sicherheitsrichtlinien
- 3. Backup-Strategien & Model-Switching**
 - Fallback zu anderen Modellen (Claude, Mistral, Gemini)
 - Modellkaskadierung & Kostenkontrolle
- 4. Hands-on: Sicherheit & Erweiterung**
 - Prompt Injection ausprobieren & absichern
 - Multimodales Interface mit Streamlit + Whisper/DALL·E

Tag 5: Abschlussprojekt – Eigene LLM-Anwendung

- 1. LLM-Agenten & KI-Workflows**
 - Tool-Verknüpfung mit LangChain Agents (z. B. Websuche, Taschenrechner, Datei-Parser)
 - Integration mit Make.com, Zapier, n8n
- 2. Auswahl & Hosting von Modellen**
 - OpenAI, Claude, DeepSeek, LLaMA – was, wann, wie?
 - Open-Source-LLMs lokal deployen
- 3. Hands-on-Projekt**
 - Baue einen funktionalen LLM-Prototyp:
 - RAG + GPT + Frontend (Streamlit oder Gradio)
 - Use-Case: Developer Assistant, interner Helpdesk, Dokumentensuche
- 4. Präsentation & Ausblick**
 - Vorstellung der Projekte
 - Feedbackrunde & nächste Schritte: LLMOps, Monitoring, Prompt-Analytics

