**Building Generative AI Applications with Large Language Models**

**Building AI Applications with Large Language Models**

**Hands-on Generative AI Engineering with Large Language Models**

**Part 0: Introduction to the course**

* Course structure
* Learning paths

**Part 1: Software prerequisites for Python project**

* IDEs
  + VS Code
  + PyCharm
* Terminals
  + Windows: PowerShell, …
  + Mac OS: iTerm2, …
  + Linux: Bash, …
* Python installation
  + Python installer
  + Anaconda distributions
* Python environment
  + `venv`
  + `conda`
* Python packages
  + Pypi, pip
  + Anaconda, conda

**Part 2: Introduction to Transformer – Attention Is All You Need**

* Introduction to NLP before and after Transformer’s arrival
* Explaining Transformer’s block-by-block
* Explaining Transformer’s training process
* Explaining Transformer’s inference process

**Part 3: Implementing of Transformer in Python and PyTorch**

* Build a transformer package
* Write train and inference scripts
* Experiment with notebook files

**Part 4: Generative AI with Hugging Face Ecosystem**

* Introduction to Hugging Face
* Hugging Face Hubs
  + Models
  + Datasets
  + Spaces
* Hugging Face Libraries
  + Transformers
  + Datasets,
  + Evaluate, …
* Practical guides with HF
  + Fine-tune a pre-trained LM with HF
  + HF Notebook guides

**Part 5: Components to Build LLM-based Web Applications**

* LangChain / LlamaIndex
* Open source / Private LLM
* Vector Embedding
* Vector database
* Prompt Engineering
* Streamlit / Gradio

**Part 6: Building LLM-based Web-Applications**

* Task-specific AI Assistants
  + Culinary AI assistant
  + Marketing AI assistant
  + Customer AI assistant
  + SQL-querying AI assistant
  + Travel AI assistant
  + Summarization AI assistant
  + Interview AI assistant
* Simple AI Chatbot
* RAG (Retrieval Augmented Generation)-based AI Chatbot
* Agent-based AI Chatbot
  + AI Chatbot with Math problems
  + AI Chatbot with Search problems

**Part 7: Serving LLM-based Web Applications**

* Create the Frontend and Backend as two separate services
* Communicate between frontend and backend using a REST API
* Serve the application with Docker
  + Install, run and enable communication between Frontend and Backend in a single Docker container
* Use-case
  + An LLM-based song recommendation app

**Part 8: Thank you**