National University of Engineering (UNI)
School of Computer Science
Syllabus 2027-I

1. COURSE

CS3E2. Applied LLM Engineering (Elective)

2. GENERAL INFORMATION

2.1 Course	:	CS3E2. Applied LLM Engineering
2.2 Cycle	:	CS300 Cycle.
2.3 Credits	:	3
2.4 Hours	:	2 HT; 2 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Elective
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	• AI264. Deep Learning. (7^{th} Sem) • AI262. Natural Language Processing.

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Large language models (LLMs) have become a foundational technology for building modern software applications, yet effectively engineering systems around them requires skills that go well beyond classical machine learning or software engineering: prompting, retrieval-augmented generation, parameter-efficient fine-tuning, agentic tool-use workflows, and the evaluation, serving, and governance practices that make an LLM-based application trustworthy and production-ready.

This course gives students hands-on experience across the full LLM application-engineering stack. Students begin by reviewing the transformer architecture, tokenization, and the pretraining/inference pipeline that underlie modern LLMs, then progress through prompt engineering, retrieval-augmented generation, and parameter-efficient fine-tuning (LoRA/PEFT) as the three principal techniques for adapting an LLM's behavior to a specific task without retraining it from scratch.

The course then turns to agentic workflows and tool-use architectures, in which an LLM plans and executes multi-step tasks using external tools, and to the evaluation methods (benchmarks, hallucination detection, safety/red-teaming) needed to assess whether an LLM-based system is good enough to deploy. The course closes with LLM Ops (serving, latency/cost optimization, caching, quantization), responsible-AI practices (bias, privacy, prompt-injection security, governance), and a capstone project in which students integrate these techniques into a single end-to-end LLM-powered application.

5. GOALS

- Explain the architecture, pretraining, and inference pipeline of modern large language models (LLMs).
- Design effective prompts and retrieval-augmented generation (RAG) pipelines to ground an LLM's output in external knowledge.
- Apply parameter-efficient fine-tuning techniques (LoRA/PEFT) to adapt a pretrained LLM to a target task or domain.
- Design agentic, tool-using LLM workflows and evaluate the quality, safety, and robustness of LLM-based systems.
- Operate LLM-based applications responsibly in production, addressing serving/cost trade-offs, security risks, and governance requirements, and integrate these techniques into a deployable end-to-end application.

6. COMPETENCES

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Usage)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

7. TOPICS

Unit 1: Large Language Model Foundations (8 hours)	
Competences Expected: AG-C07	
Topics	**Learning Outcomes**
• Subword tokenization schemes (BPE, WordPiece, byte-level tokenization) and their effect on vocabulary size, sequence length and multilingual coverage Natural Language Processing. • Decoder-only transformer architecture for autoregressive language modeling: causal self-attention masking, positional encodings, and the stack of transformer blocks used in modern LLMs Transformer Architectures and Generative Models. • The pretraining objective (next-token prediction) over large unlabeled text corpora, and scaling laws relating model size, data size and compute to pretraining loss Neural Network Training and Optimization. • Autoregressive inference and decoding strategies: greedy decoding, beam search, temperature and top-k/top-p sampling, and the KV-cache that makes incremental decoding efficient.	• Explain how a decoder-only transformer processes a tokenized input sequence to produce the next-token probability distribution [Explain] • Compare subword tokenization schemes in terms of vocabulary size, sequence length and handling of out-of-vocabulary words [Compare] • Apply different decoding strategies (greedy, beam search, sampling) to a pretrained LLM and analyze the effect on output quality and diversity [Apply] • Analyze how model size, training data size and compute budget affect a LLM's pretraining loss given a set of scaling-law measurements [Analyze]
Readings : [Vas+17], [Ras24]	

Unit 2: Prompt Engineering Techniques (6 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Zero-shot and few-shot prompting: framing a task through instructions alone versus providing in-context input/output demonstrations. • Chain-of-thought and step-by-step reasoning prompts, and self-consistency across multiple sampled reasoning paths. • System/role prompts and structured output formatting (e.g., requesting JSON or XML-constrained responses). • Prompt sensitivity and robustness: how small wording or ordering changes affect model output, and iterative prompt refinement Agentic Loop Engineering.	• Design a few-shot prompt with in-context demonstrations for a given classification or extraction task [Design] • Apply chain-of-thought prompting to improve a model's performance on a multi-step reasoning task [Apply] • Evaluate the sensitivity of a prompt to wording and ordering changes and refine it to reduce output variance [Evaluate]
Readings : [Whi+23], [Wei+22a]	

Unit 3: Retrieval-Augmented Generation (RAG) (8 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Document chunking strategies and dense-vector embedding of text chunks for semantic retrieval. • Vector similarity search (cosine/dot-product similarity, approximate nearest-neighbor indexes) and vector database/index selection NoSQL Systems, Semi-structured and Unstructured Databases. • The retrieve-then-generate pipeline: query embedding, top-k retrieval, context assembly, and prompt construction for the generator model. • Advanced RAG patterns: re-ranking retrieved chunks, hybrid (sparse+dense) retrieval, and multi-hop/iterative retrieval Natural Language Processing.	• Design a chunking and embedding strategy for a given document collection to support semantic retrieval [Design] • Implement a retrieve-then-generate pipeline that grounds an LLM's answer in retrieved context [Implement] • Compare dense, sparse and hybrid retrieval strategies in terms of recall and relevance for a given corpus [Compare] • Evaluate the groundedness of a RAG system's answers against the retrieved source documents [Evaluate]
Readings : [Lew+20], [Gao+24]	

Unit 4: Fine-Tuning and Parameter-Efficient Methods (8 hours)	
Competences Expected: AG-C07,AG-C09	
Topics	**Learning Outcomes**
• Full fine-tuning of a pretrained LLM versus the cost and overfitting risks of updating all of its parameters on a small downstream dataset Regularization Techniques for Deep Networks. • Low-Rank Adaptation (LoRA) and related adapter-based parameter-efficient fine-tuning (PEFT) methods: low-rank weight updates, adapter placement, and merging adapters back into the base model. • Quantized fine-tuning (e.g., QLoRA) for adapting large models under limited GPU memory. • Instruction tuning and preference-based fine-tuning (e.g., RLHF/DPO) to align a fine-tuned model's behavior with desired instructions and preferences.	• Compare full fine-tuning and LoRA-based parameter-efficient fine-tuning in terms of compute cost, memory footprint and task performance [Compare] • Implement a LoRA fine-tuning run on a pretrained LLM for a target downstream task [Implement] • Determine an appropriate adapter rank and placement for a given fine-tuning budget and task [Determine] • Explain how instruction tuning and preference-based fine-tuning change a base model's behavior relative to its pretrained state [Explain]
Readings : [Hu+22], [Det+23]	

Unit 5: Agentic Loop Engineering (8 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• Loop engineering as a discipline: designing the iterative observe-act-check cycle an agent runs through, as distinct from writing a single prompt • The Gauntlet Loop pattern (Matt Shumer, 2026): a lead agent breaks a goal and a quality bar into the smallest independently judgeable pieces, then fans out builder and critic sub-agents with fresh context • Builder/critic separation: an agent that created a result is usually worse at objectively judging its own quality than at explaining it, hence the need for an independent critic • Tool use by the agent (reading and writing files, shell, search, running tests) as the way to act on the repository instead of merely generating text • Model Context Protocol (MCP) as a standardized interface between agents and data sources or tools: servers, resources, prompts and exposed tools • Executable specification: artifacts versioned in the repository (specifications, plans and task lists) that the agent consumes and updates	• Explain how loop engineering differs from engineering a single prompt [Explain] • Design a Gauntlet-Loop-style loop (builder agent + independent critic agent + blind comparison against a quality bar) to iteratively refine an AI-generated result [Design] • Compare agent autonomy levels (inline assistance, multi-file editing, unattended execution) in terms of risk and benefit [Compare] • Design and configure an MCP server that exposes to an agent the resources and actions required for a development task [Design]
Readings : [Yao+23]	

Unit 6: Evaluating LLM Systems: Benchmarks, Hallucination and Safety (6 hours)	
Competences Expected: AG-C02,AG-C07	
Topics	**Learning Outcomes**
• Standardized LLM benchmarks (e.g., multi-task knowledge and reasoning benchmarks) and automatic evaluation metrics for open-ended generation (e.g., LLM-as-judge) Machine Learning Data, Representations and Evaluation. • Human evaluation protocols: pairwise preference judgments, rubric-based scoring, and inter-rater agreement. • Hallucination: causes, detection techniques (e.g., fact-checking against a retrieved source, self-consistency checks), and mitigation strategies LLM-based Coding Agents. • Adversarial and safety testing (red teaming): jailbreak attempts, toxicity/bias probes, and safety benchmark suites Neural Networks and Ethics.	• Design an evaluation protocol (benchmark selection plus human evaluation rubric) for a given LLM-based application [Design] • Detect hallucinated content in an LLM's output by checking it against retrieved source documents [Detect] • Assess an LLM-based application's robustness to a red-teaming/jailbreak attempt and propose mitigations [Assess]
Readings : [LBL+23]	

Unit 7: LLM Serving, Latency and Cost Optimization (8 hours)	
Competences Expected: AG-C09	
Topics	**Learning Outcomes**
• LLM serving architecture: request batching, continuous/dynamic batching, and the KV-cache's role in incremental decoding throughput Performance Testing and Benchmarking. • Model quantization (e.g., int8/int4 post-training quantization) and its trade-offs between memory footprint, latency and output quality Neural Network Training and Optimization. • Caching strategies for LLM-serving systems: prompt/response caching, semantic caching, and prefix/KV-cache reuse across requests. • Cost and latency optimization trade-offs: model size selection, hardware accelerator choice, and request routing across model tiers Performance Testing and Benchmarking.	• Explain how continuous batching and KV-cache reuse improve the throughput of an LLM-serving system [Explain] • Apply a quantization technique to a pretrained LLM and measure the resulting latency, memory and quality trade-offs [Apply] • Design a caching strategy for a given LLM-serving workload to reduce redundant inference calls [Design] • Estimate the serving cost of an LLM-based application given expected request volume, model size and hardware [Estimate]
Readings : [Kwo+23]	

Unit 8: AI and Hardware Security (6 hours)	
Competences Expected: AG-C02	
Topics	**Learning Outcomes**
• Impact of AI on security and privacy: using AI to bolster defenses as well as address increased adversarial capabilities due to AI AI Applications and Societal Impact, Additional AI Applications and Societal Impact, Society, Ethics, and the Profession, System Design. • Security risks in building and operating AI/machine learning systems (e.g., algorithm bias, knowledge corpus bias, training corpus bias, copyright violation) AI Applications and Societal Impact, Additional AI Applications and Societal Impact, Machine Learning Formulation and Evaluation, Deep Learning, RL and Ethics, Introduction to AI: Concepts and Intelligent Behavior, Problem Characteristics and Agent Nature, Machine Learning Fundamentals, Machine Learning Data, Representations and Evaluation, Neural Networks and Ethics. • Professional ethical considerations when designing secure systems and maintaining privacy; ethical hacking Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. • Ethical implications of emerging technologies (AI, automation, data privacy) • Direct and indirect prompt injection: attacker-controlled instructions embedded in user input or in retrieved/tool-returned content that hijack an LLM application's behavior Injection and Input Validation. • Security governance: organizational objectives and general risk assessment.	• Show how an adversary could use machine learning algorithms to reduce the security of a system [Analyze] • Apply a set of privacy regulations to design a system that protects privacy [Apply] • Evaluate the ethical dimensions of a project with significant social or environmental impacts in a different cultural context [Assessment] • Design mitigations (instruction hierarchy, input/output filtering, least-privilege tool access) to reduce an LLM application's prompt-injection attack surface [Design]
Readings : [OWA25], [Wei+22b]	

Unit 9: LLM-Powered Application Capstone (6 hours)	
Competences Expected: AG-C02,AG-C09	
Topics	**Learning Outcomes**
• Integrating a prompting/RAG/agentic component, a serving layer and an evaluation loop into a single end-to-end LLM-powered application Coding Practices. • Iterative, evaluation-driven refinement of an LLM-powered application: using evaluation results and user feedback to guide changes to prompts, retrieval, or fine-tuning Large-Scale Construction and Process. • Deployment considerations for an LLM-powered application: packaging, monitoring in production, and a rollback/fallback plan for model or infrastructure failures.	• Integrate prompting, retrieval and/or fine-tuned components with a serving layer into a working end-to-end LLM-powered application [Integrate] • Use evaluation results and user feedback to iteratively refine a deployed LLM-powered application [Use] • Plan a deployment and rollback strategy for an LLM-powered application, considering monitoring and failure modes [Plan]
Readings : [Alt24]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

Theory Sessions:
The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

Lab Sessions:
In order to verify their competences, several activities including active learning and roleplay will be developed during lab sessions.

Oral Presentations:
Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is considered as the mark of a qualified practice. The use of our virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

The final grade NF depends on the following grades:

- PE: Partial Exam.
- FE: Final Exam.
- P: Permanent Evaluation (Average of practical work and graded assignments).

The final grade NF is calculated as follows:

$$NF = \frac{PE + FE + P}{3}$$

10. BASIC BIBLIOGRAPHY

[Vas+17] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[Lew+20] Patrick Lewis et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[Hu+22] Edward J. Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations (ICLR)*. 2022.

[Wei+22a] Jason Wei et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[Wei+22b] Laura Weidinger et al. "Taxonomy of Risks posed by Language Models". In: *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*. 2022.

[Det+23] Tim Dettmers et al. "QLoRA: Efficient Finetuning of Quantized LLMs". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[Kwo+23] Woosuk Kwon et al. "Efficient Memory Management for Large Language Model Serving with PagedAttention". In: *ACM Symposium on Operating Systems Principles (SOSP)*. 2023.

[LBL+23] Percy Liang, Rishi Bommasani, Tony Lee, et al. "Holistic Evaluation of Language Models". In: *Transactions on Machine Learning Research (TMLR)* (2023).

[Whi+23] Jules White et al. "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT". In: *arXiv preprint arXiv:2302.11382* (2023).

[Yao+23] Shunyu Yao et al. "ReAct: Synergizing Reasoning and Acting in Language Models". In: *International Conference on Learning Representations (ICLR)*. 2023.

[Alt24] Valentina Alto. *Building LLM Powered Applications: Create Intelligent Apps and Agents with Large Language Models*. Packt Publishing, 2024.

[Gao+24] Yunfan Gao et al. "Retrieval-Augmented Generation for Large Language Models: A Survey". In: *arXiv preprint arXiv:2312.10997* (2024).

[Ras24] Sebastian Raschka. *Build a Large Language Model (From Scratch)*. Manning Publications, 2024.

[OWA25] OWASP Foundation. *OWASP Top 10 for Large Language Model Applications*. `https://owasp.org/www-project-top-10-for-large-language-model-applications/`. 2025.