Final Project: LLM Engineering & Prompting

Pretraining • Fine-Tuning • Data Design • Evaluation

Project Overview

This final project offers you the opportunity to gain hands-on experience with large language models through either engineering-focused (mechanistic) work or user-focused (prompting) exploration. You will have access to two fully functional in-house LLMs and will build practical skills that transfer to real-world AI systems. You may also propose an open-ended, LLM-related project.

Team Size: 1–3 students

Focus Areas: Language modeling, instruction tuning, data design, exploration, and evaluation

Choose Your Path

In this course, you can choose your path to mastery based on your interests and career goals. Both paths require rigor and curiosity, and will give you highly valuable, complementary skills in the age of LLMs.

You will choose one core track for your main project, but we encourage exploration across both.

Engineer Track (Mechanistic Focus)

Are you fascinated by how the AI 'brain' is built and functions? This track will have you performing 'brain surgery' on our models—removing parts to see what breaks, analyzing training dynamics, and optimizing them for speed and performance.

Example Tasks:

- Ablation Studies: Remove or restore attention heads and/or layers to measure performance changes and identify which components matter most
- Training Dynamics: Visualize loss curves and embedding space shifts at different training stages
- Model Optimization: Extend or modify one of our in-house LLMs to excel at a specific new capability
- Efficiency Analysis: Compare full fine-tuning vs. parameter-efficient methods (LoRA) vs. reinforcement-learning–based post-training (e.g., GRPO)

User Track (Prompting Focus)

Are you more interested in the art of conversation with AI—learning its quirks, discovering how to get the best results, and testing its limits through clever dialogue? This track will turn you into a prompt architect and safety tester.

Example Tasks:

- Prompt Optimization: Systematically test few-shot vs. zero-shot vs. chain-of-thought prompting strategies
- Reasoning Analysis: Compare different reasoning elicitation techniques and their effectiveness
- Red Teaming: Attempt to identify model vulnerabilities and test safety boundaries
- Evaluation Design: Build comprehensive test suites to measure model capabilities and limitations

Project Options

You have three distinct tracks for your final project. Project scope should scale with team size—larger teams are expected to deliver proportionally greater depth or breadth.

Option 1: TinyStories LLM (Data-Centric Track)

Based on Eldan & Li (2023), explore how small models develop capabilities through carefully curated data. This track emphasizes the critical role of data quality and design in model performance.

Potential Directions:

Personality Engineering

Curate three distinct instruction-tuning datasets (~500 examples each), such as Pirate, Shakespearean, or Technical Writer personas. Fine-tune the base model on each and perform comparative analysis of emergent 'voice' characteristics, consistency, and quality.

Efficiency Study

Transition from full fine-tuning to Parameter-Efficient Fine-Tuning (PEFT) methods such as LoRA or IA^3. Compare compute cost, training time, inference speed, memory usage, and output quality across methods.

Failure Analysis & Patching

Run the model on diverse test cases, categorize error types (e.g., coherence breaks, repetitive phrases, factual inconsistencies), and design a targeted 'patch' dataset to fix specific failure modes. Measure improvement quantitatively.

Option 2: Arithmetic LLM (Optimization Track)

Investigate trade-offs between mathematical reasoning, parameter efficiency, and reinforcement learning alignment. This track focuses on understanding model optimization and reasoning capabilities.

Potential Directions:

LoRA Rank Analysis

Evaluate 2–3 provided LoRA checkpoints with different ranks (e.g., r = 2, 8, 32). Create visualizations plotting Rank vs. Task Performance and Rank vs. Inference Speed to identify efficiency/accuracy sweet spots. Analyze the practical implications of rank selection.

GRPO Reward Ablation

Propose and test alternative reward signals (e.g., step-wise correctness, length penalty, or chain-of-thought template matching). Analyze whether certain rewards cause the model to produce 'nonsense logic' or other undesired behaviors. Document the relationship between reward design and model outputs.

Out-of-Distribution (OOD) Stress Testing

If the model was trained on 3-digit arithmetic, systematically probe where reasoning collapses (e.g., 4-digit, 5-digit, decimal inputs, negative numbers, edge cases like division by zero). Document failure patterns, create visualizations of performance degradation, and hypothesize underlying causes.

Option 3: Open-Ended Project

You may also propose a custom project that matches your interests. This option requires approval from the instructional team and is best suited for students who already have a domain problem to solve or a well-defined idea they want to explore.

Example Ideas:

- Fine-tune a pre-trained model from Hugging Face for your task. Refer to hf_tutorials.html for guidance.
- Model Merging: Merge the two in-house models into a single “generalist” model using techniques such as model averaging or task arithmetic, then evaluate multi-task performance.
- Tool-Augmented Reasoning: Give the Arithmetic LLM access to a calculator tool (API) and measure improvement on harder calculations; analyze when and why the model chooses to use the tool.

- Interactive Web Interface: Build a chat UI with advanced prompting controls (system prompt editing, few-shot example banks, temperature control) and run a small user study to evaluate effectiveness.

Approval Requirements

Submit a one-page proposal containing:

1. Clear research question or engineering goal with motivation
2. Methodology and technical approach with specific steps
3. Brief related work review (2–3 relevant papers with key takeaways)

Team Size & Expectations

Project scope should scale appropriately with team size. We evaluate based on the depth and breadth of work relative to the number of team members.

Individual (1 person):

Expected workload approximately equals 2 standard coding assignments. Focus on depth within a single direction or technique. A well-executed, focused project is preferred over a shallow exploration of multiple areas.

Group (2–3 people):

Should deliver proportionally greater depth or scope. Examples include comparing multiple techniques systematically, building an end-to-end application with multiple components, or conducting comprehensive ablation studies across multiple dimensions.

Individual Contributions:

All teams must clearly document individual contributions in the final report and repository README. Specify who worked on which components, analyses, and sections of the writeup.

Deliverables

1. Final Presentation

Time Allocation: 5 minutes per person + 2~5 minutes Q&A per group (peer-graded)

Your presentation should tell a clear, compelling story about your project. Focus on engaging your audience and demonstrating your key insights.

Required Components:

- Live Demo: If possible, showcase your main finding or key innovation through a live demonstration or interactive visualization
- Key Insights: Share what worked, what surprised you, and what your biggest technical challenge was—be honest and reflective
- Results Summary: Present 2–3 of your strongest findings with clear supporting evidence (graphs, metrics, qualitative examples)
- Lessons Learned: Conclude with key takeaways. Specifically answer: 'What would you do differently after taking this course when using or building LLMs?'

Evaluation Criteria:

Peer grading will emphasize clarity of communication, technical depth, storytelling effectiveness, and quality of insights. Strong presentations balance technical rigor with accessibility.

2. GitHub Repository (Recruiter-Ready)

Your code repository must be clean, reproducible, and professional—of a quality you could confidently share with a potential employer or include in your portfolio.

Required Components:

- README.md: Clear overview with project goals, motivation, setup instructions, and example usage. Include either a one-command reproduction script or detailed step-by-step instructions.
- Environment Setup: Provide dependency files (requirements.txt, environment.yml, or pyproject.toml) for easy environment recreation. Pin versions for reproducibility.
- Organized Code: Use logical directory structure (e.g., src/, data/, scripts/, notebooks/, configs/, results/) with clear purposes for each directory.
- Documented Scripts: All training and evaluation scripts should include argparse/click interfaces with help text and usage examples in the README.
- Output Management: Clearly document where results, logs, and checkpoints are saved. Include instructions for regenerating all figures and tables from the paper.
- License: Include an open-source license (MIT recommended) if you plan to share the repository publicly.
- Individual Contributions: Document which team members worked on which components in the README.

Code Quality Expectations:

Write clean, well-commented code with meaningful variable names. Include docstrings for functions. Your code should be easy for others to understand and build upon.

3. Technical Report

Include a concise technical report (4–6 pages) in your repository as a PDF. The report should follow a standard research paper structure and be suitable for sharing with technical audiences.

Suggested Structure:

- Introduction: Motivation, research question, and contributions
- Background: Brief overview of relevant concepts and related work
- Methodology: Detailed description of your approach, experimental setup, and implementation details
- Results: Quantitative and qualitative findings with visualizations
- Discussion: Analysis of results, limitations, and future directions
- Conclusion: Key takeaways and broader implications
- Individual Contributions: Clear breakdown of who did what

Submission Logistics & Deadlines

Important Dates:

- Feb 16: Project proposals due — 1 page submitted to Canvas
- Day before presentation: WIP slides due on Canvas for review
- Presentation date: Final slides due (PDF or PowerPoint) on Canvas by end of day
- March 17: GitHub repository link and technical report PDF due on Canvas

Submission Requirements:

- Presentation slides: Upload to Canvas (PDF or PowerPoint format)
- Presentation schedule: Sign up for a presentation slot using this link
- GitHub repository: Submit the repository URL on Canvas along with your technical report PDF

Late Policy:

No late days can be used for the final project. All submissions must be on time. Plan accordingly and start early.

Free Rider Policy:
For team projects, all members are expected to contribute meaningfully and consistently. Teams must keep a brief record of contributions (e.g., commit history, task checklist, meeting notes) and complete the required peer evaluation.

If a free rider is identified (through peer evaluations, contribution records, or instructor review), the instructional team may take reduce the student's project grade (including assigning an individual grade different from the team grade)

If you anticipate an imbalance in workload, **contact the instructional team early**—do not wait until the final week.

Grading Rubric

Your project will be evaluated holistically across the following dimensions. Expectations are calibrated to team size and chosen track difficulty.

Component	Evaluation Criteria
Presentation (20%)	Clarity of communication, storytelling effectiveness, technical depth, engagement, quality of demo/visualizations
Technical Execution (30%)	Code quality, implementation correctness, appropriate methodology, handling of edge cases, debugging/problem-solving
Experimental Design (20%)	Clear hypotheses, appropriate baselines/controls, comprehensive evaluation metrics, statistical rigor
Analysis & Insights (20%)	Depth of analysis, quality of insights, connection to broader implications, critical reflection on limitations
Reproducibility (10%)	README quality, code organization, documentation, ease of setup, dependency management

Tips for Success

- Start Early: Begin exploring the models and brainstorming ideas as soon as the project is released. Last-minute work leads to shallow results.
- Focus on Depth: A deep, well-executed exploration of one direction is better than a shallow survey of many techniques.
- Document as You Go: Keep a research log of experiments, hypotheses, and results. This makes writing the final report much easier.
- Seek Feedback: Use office hours to discuss ideas, troubleshoot issues, and get feedback on your approach.
- Make It Portfolio-Worthy: Treat this as an opportunity to create work you can showcase to employers or graduate schools.

- Learn Transferable Skills: Focus on developing workflows and intuitions that will serve you beyond this specific assignment.

We're excited to see what you build! This project is your opportunity to dive deep into LLMs and develop practical skills that will serve you throughout your career. Choose a direction that genuinely interests you, and don't hesitate to ask for guidance along the way.

Good luck, and happy experimenting!