LLMs & Learning

CSE-6040

What Are LLMs?

- Neural networks built on **Transformer** architecture (Attention mechanisms), see attention-is-all-you-need
- Predict the next token based on prior tokens + training data probabilities
- Core loop: *predict* -> *sample* -> *repeat* (scaled up for ChatGPT, etc.)

Recommended Learning Path (MSA Program)

Course	Focus
CSE 6040	NumPy & linear algebra foundations
ISYE 6740	ML fundamentals: backpropagation, gradient descent, forward pass
CSE 8803	NLP + Transformer architecture + Attention mechanism

- This course path will teach you the background math and codes, but it won't teach you the *"how to use it"* or best way to use it
- Combined with OOP skills -> understand microgpt (~200 lines of pure Python, no libraries) from Andrej Karpathy
- **Key takeaway:** LLM technology is not magic - it's understandable.

Output Quality

- Stochasticity is inherent
  - same prompt, different temperature -> different outputs. That's why LLMs aren't deterministic guarantees.
- Prompt quality directly shapes every step
  - early tokens influence what comes later through the Attention mechanism. A well-crafted prompt steers the entire chain of predictions.
- "Hallucinations" happen naturally
  - at each step, the model picks from a distribution, not ground truth. Errors compound over long generations.

Summary For the Typical User / Consumer

- Most people use provider APIs (ChatGPT, Gemini, Claude, Qwen, etc.)
- Output quality depends on:
  - Quality of training data
  - Specificity of user input prompt
- **The only knob you control = your prompt**
  - Better + more specific prompts -> better outputs (not guaranteed - LLMs are stochastic)

The Landscape: Models vs. Harnesses

Comparing a chat interface to a coding agent is like comparing a Tesla on Autopilot to a manual BMW - they serve different purposes.

Layer	What It Is	Examples
The Model (Engine)	The underlying neural network trained on data.	GPT-4o, Claude 3.5 Sonnet, Gemini 1.5 Pro
The Harness (Vehicle)	The interface, application, or system built around the model.	ChatGPT, Claude Code, Gemini CLI, Cursor

Key Takeaway:
Evaluate the *model* for raw reasoning, but evaluate the *harness* for workflow efficiency.

What Consumers Can Do

- Refine the quality and specificity of input prompts
- Develop skills to evaluate/audit LLM-generated output

Effects of AI-Generated Solutions

Positive

- Faster problem resolution (with caveats)

Negative / Risks

- More complex solutions
- More solutions that aren't feasible for humans to understand in a reasonable time
- *Wrong* solutions that just happen to work

Effects of AI-Generated Solutions (Cont.)

Positive

- Access to broad knowledge without extensive research time

Negative / Risks

- **Stochastic outputs** - not guaranteed correct
- **Homogenized creativity** - trained on existing data - similar logic flows & patterns. e.g. the-100000-whys-of-ai
- **Group-think vulnerability** - over-reliance by users lacking domain depth
- **Bias** from training data or provider-injected signals (culture, policy, geopolitics)
- **Outdated knowledge** when underlying distributions shift
- **Sycophantic ability** LLMs can take short-cuts / workarounds to please humans

Here Be Dragons

- **Early days**: Human-AI interaction is still not well understood
- **Productivity vs. risk**: GenAI boosts output, but may erode learning and critical thinking
- **Prompting requires expertise**: Good questions need domain knowledge, built only through hands-on experience
- **No expertise, no vetting**: Without that foundation, we can't judge if AI's answers are actually right

Below are some additional resources that you can use should you be interested in learning about the impacts of GenAI use

- https://www.microsoft.com/en-us/research/wp-content/uploads/2025/01/lee_2025_ai_critical_thinking_survey.pdf
- https://www.computer.org/csdl/magazine/co/2025/08/11104179/28MaT9KU2ly
- https://bmjgroup.com/overreliance-on-ai-risks-eroding-new-and-future-doctors-critical-thinking-while-reinforcing-existing-bias/
- https://academic.oup.com/pnasnexus/article/4/10/pgaf316/8303888
- https://www.theguardian.com/technology/2025/oct/24/sycophantic-ai-chatbots-tell-users-what-they-want-to-hear-study-shows
- https://www.anthropic.com/research/AI-assistance-coding-skills
- https://www.nature.com/articles/s41598-025-98385-2
- https://arstechnica.com/ai/2026/05/influential-study-touting-chatgpt-in-education-retracted-over-red-flags/
- https://ergosphere.blog/posts/the-machines-are-fine/

For Students & Knowledge Workers: The "How"

How to interact with the current AI landscape:

- **Be Platform Agnostic:** Learn to use multiple LLMs (OpenAI, Anthropic, Google, open-source). Treat them like different car brands or visualization tools (Tableau vs. PowerBI).
- **Match the Tool to the Task:** Recognize when to use a web UI (exploration) versus a CLI/IDE integration (production/coding).
- **Build Domain Knowledge:** You must possess broad and specific knowledge to prompt effectively and spot subtle errors.
- **Master the Prompt:** Output quality scales directly with your ability to write clear, constrained, and context-rich instructions.

For Students & Knowledge Workers: The "What"

What to use LLMs for in your daily workflow:

- **Accelerate Skill Growth:** Use LLMs to expand your own depth and width, not to replace your critical thinking.
- **Audit and Refine:** Have the AI critique your working solutions to find optimizations or edge cases.
- **Generate Targeted Practice:** Create tailored study guides or practice problems constrained to specific frameworks (e.g., NumPy, Pandas).
- **Vet Against Reality:** Always cross-reference LLM outputs against official documentation, official source code, or trusted publications.

Mental Models: How to Think About LLMs

- **The "Compliance Attorney" Analogy:** LLMs require hyper-specific phrasing. If you don't use the exact right words in the precise order, you won't get the correct answer. You must constrain the prompt to get what you need.
- **Wisdom Over Syntax (The Architect):** The most valuable skill is understanding business needs and system constraints, not memorizing syntax.
  - *Example:* A developer provided a 9-page prompt detailing architectural constraints. The LLM handled the syntax, completing a 300-hour migration in 2.5 hours.
- **Learn via Familiarity:** To learn how to prompt (and spot "hallucinations"), ask the LLM to solve a non-coding task you already know how to do perfectly (e.g., watering a strangely shaped lawn).

Sample Prompts for Learning

Context	Prompt
Critique a working solution	*"Audit my solution and suggest improvements given constraints* `<xyz>` *."*
Find more efficient methods	*"Audit my solution and suggest more efficient approaches using only* `<xyz>` *(e.g., pandas, SQL, numpy)."*
Generate practice problems	*"Evaluate these problems & solutions, then generate similar practice questions restricted to* `<xyz>` *."*
Identify underlying themes	*"What is the general theme here? What should I focus on?"*

Sample Prompts for Learning (Cont.)

Context	Prompt
Learn a package / create study guide (via NotebookLM / Gemini Notebook or RAG)	*"Faithfully respond from provided sources. Include details, code examples."* *"Create a report combining our chat + official docs with detailed examples."* *"Generate high-level overview - dive into specific API mechanics with code."*

> ***Tip:*** *Use NotebookLM (aka Gemini Notebook)'s Mind Map and Data Table features for structured learning.*