# Introduction to Large Language Models

Spring 2026

**LLM Concepts:**
Prompting, In-Context Learning, RAG, Fine Tuning

(Some slides adapted from Ralph Grishman at NYU,
Yejin Choi at UWashington, N. Tomura at UDepaul, Jurafsky and Martin,
CS224N, CS224, CME295 at Stanford and other resourses on the web)

# Prompting

- **Prompting**
- **Prompting** is simply giving instructions to the model.
- You tell the model *what you want*, and it uses the knowledge stored in its parameters.
- **Characteristics**
  - No extra information is provided except your instructions.
  - The model relies entirely on what it already knows from pretraining.
  - Can be **zero-shot** (“Do X”) or **few-shot** (“Here are 2 examples, now do X”).
- **When to use:**
  - When the task is simple.
  - When the model already knows the necessary information.
  - For general reasoning, summarization, writing, translation, etc.

# In-Context Learning

- **In-Context Learning (ICL)**
  - ICL is when you teach the model a pattern *within the prompt* using examples.
  - You temporarily “program” the model by showing sample input-output pairs.
- **Characteristics**
  - Model parameters do **not** change — learning is temporary.
  - It recognizes the structure from the examples you include.
  - It is basically **few-shot prompting**, but with emphasis on the learning behavior.
- **When to use:**
  - When the model needs to follow a very specific structure or style.
  - When doing classification, labeling, or task-specific formatting.
  - When you want the model to imitate a pattern without fine-tuning.

# In Context Learning

- **In-Context Learning (ICL)** is the ability of large language models (LLMs) to **adapt their behavior at inference time using only the prompt**, **without updating model parameters**. The model infers the task, format, or reasoning pattern from the context and continues it coherently.
- **What ICL is (and is not)**
  - ✅ **Is**: inference-time adaptation via the prompt
  - ❌ **Is not**: fine-tuning, training, or parameter updates
  - ❌ **Is not**: long-term memory or learning stored across prompts
- **How ICL works (mechanism)**
- LLMs are trained to predict the next token given a long sequence. At inference:
  - Instructions, examples, and reasoning are just **tokens** in the same sequence.
  - Self-attention lets the model **compare the query to examples** and **replicate the pattern**.
  - No gradients are computed; behavior changes are **temporary**.

# ICL setups (by number of examples)

- ICL is the **umbrella phenomenon**; these are common **settings**:
  - **Zero-shot ICL**: no examples, just instructions
    Classify sentiment: “The movie was great.” → ?
  - **One-shot ICL**: exactly one example
  - **Few-shot ICL**: a small set (typically 2–10) of examples
- All are ICL; they differ only in **how much context you provide**.
- **Chain-of-Thought (CoT) as ICL**
- **CoT** is ICL with **reasoning tokens** included (explicitly or by instruction):
  - Few-shot CoT: give examples with step-by-step reasoning
  - Zero-shot CoT: ask “Let’s think step by step”
- CoT works by **scaffolding intermediate states** inside the context, which improves multi-step tasks (math, logic).
- **Scaffolding** refers to **structuring the interaction or reasoning process so the model can perform a complex task more reliably**, by breaking the task into manageable steps or providing intermediate guidance—*without changing model parameters*.

# RAG

- **Retrieval-Augmented Generation (RAG)**
  - RAG connects a language model to an external knowledge source (IR System).
- The system:
  - **Retrieves** relevant documents using vector search.
  - Places those documents **into the prompt**.
  - The LLM uses them to generate an answer.
- **Characteristics**
  - Overcomes model knowledge cutoff.
  - Provides **fresh**, **domain-specific**, or **proprietary** information.
  - The model becomes a reasoning engine on top of your data.
- **When to use:**
  - When answering questions about PDFs, internal documents, websites, databases.
  - When information is **not** stored in the model's parameters.
  - For enterprise, research, legal, medical, or technical applications.

# RAG — Retrieval Augmented Generation

- **Retrieval-Augmented Generation (RAG)** is an **inference-time approach** that improves LLM outputs by **retrieving external information** and **conditioning generation on it**—*without updating model weights*.
- **Big idea (intuition):**
  - **If the model might not know—or might hallucinate—retrieve the facts first, then generate.**
  - RAG gives the model **fresh, private, or domain-specific knowledge** at query time and asks it to **reason and write grounded answers** using that context.
- **How RAG works (pipeline)**
  - **Query** from the user
  - **Retrieve** relevant documents (search / vector DB)
  - **Augment** the prompt with retrieved passages
  - **Generate** an answer grounded in those passages

User Question
↓
Retriever (BM25 / embeddings)
↓
Relevant Documents
↓
Prompt = Instruction + Retrieved Context
↓
LLM → Answer

# RAG

- **What problem RAG solves**
  - **Knowledge gaps**: facts not in the model or beyond its cutoff
  - **Freshness**: recent events, live data
  - **Private data**: company docs, internal wikis
  - **Hallucinations**: grounding answers in cited sources

    RAG fixes *information availability*, not reasoning ability.
- **What RAG does *not* do**
  - ❌ Does **not** teach new tasks (that's ICL / fine-tuning)
  - ❌ Does **not** change model preferences or safety (that's fine-tuning/DPO)
  - ❌ Does **not** guarantee correctness if retrieval is poor

# Prompt pattern for RAG

```
You are a helpful assistant. Use the context to answer accurately.


Context:

[Doc 1]

[Doc 2]


Question:

...


Answer (cite sources if possible):
```

Good prompting matters: without it, models may ignore context or copy verbatim.

# RAG vs. In Context Learning (ICL)

- ICL teaches the model how to do a task from the prompt.
- RAG gives the model the information it needs to answer.

| Aspect | ICL | RAG |
|---|---|---|
| Model parameters | ❌ Unchanged | ❌ Unchanged |
| Prompt content | Instructions, examples, reasoning | Retrieved documents |
| External data | ❌ No | ✅ Yes |
| Persistence | None (per prompt) | None (per prompt) |

# Fine-Tuning (for Large Language Models)

- **Fine-tuning** is a **training-time process** where a pretrained language model's parameters are **updated with gradient descent** to make the model better at specific tasks, domains, or preferences. Unlike prompting, ICL, or RAG, **fine-tuning permanently changes the model.**
- **Goal:** Specialize to a domain (law, medicine, codebase, style).
  - Data: domain-specific texts or tasks
  - Techniques: full fine-tuning or **parameter-efficient** methods (PEFT) (e.g., LoRA)
  - Instead of updating all weights, PEFT updates a small subset
- **Pros:** better accuracy and vocabulary for the domain
  **Cons:** dataset curation cost; risk of overfitting if small

# Fine tuning vs ICL vs RAG

**Fine-Tuning**

- Train on thousands of labeled or preference-ranked examples
- Result:
  - Shorter prompts
  - More reliable outputs
  - Persistent behavior

| Dimension | ICL | RAG | Fine-Tuning |
|---|---|---|---|
| Time | Inference | Inference | Training |
| Updates weights | ❌ | ❌ | ✅ |
| Adds knowledge | ❌ | ✅ | ✅ (indirectly) |
| Teaches new tasks | ✅ | ❌ | ✅ |
| Reduces hallucinations | ❌ | ✅ | ✅ |
| Cost to try | Very low | Medium | High |