Agents, Prompts & RAG — Complete Exam Notes

> **Source:** Stanford CS230 Deep Learning — "Beyond LLM" Lecture
> **Purpose:** Structured exam-prep notes — beginner-friendly, concept-focused, exam-oriented
> **Note:** All key AI terms are in **bold** and explained simply.

Table of Contents

---

1. Why Base LLMs Are Not Enough

The Core Problem

A **base model** (also called a **pre-trained model** or **vanilla LLM**) is an AI that has been trained on a large amount of general data. Think of it as a very well-read person who has studied millions of books and websites. While incredibly capable, using it straight "out of the box" for real-world tasks comes with serious limitations.

> **Analogy:** Imagine hiring a brilliant generalist who has read every book ever written — but they have never worked in your specific company, don't know about events from the last year, and sometimes confidently say things that are wrong. That's a base LLM.

Key Limitations of a Base LLM

- **Lacks domain-specific knowledge** — A general model may not know your company's internal data, a rare medical condition, or a niche industry's vocabulary. It only knows what was in its training data.
- **No current information (Knowledge Cutoff)** — The model was trained up to a specific date and knows nothing about events after that. Social media trends, new laws, or recent product launches are invisible to it.

  > **Example:** When someone posted a made-up word ("Covfefe") on Twitter, recommendation systems powered by LLMs broke because they had never seen the word before.

- **Performs poorly on narrow, specialized tasks** — Enterprise applications often need high precision. A general model might do "fine" but not "good enough" for legal, medical, or scientific tasks.

- **Hard to control** — LLMs can produce biased, offensive, or politically charged content. Even well-funded teams (like OpenAI or xAI) struggle to fully control their models.
- **Hallucinations** — The model confidently generates false information. For medical diagnoses or legal documents, this is dangerous.

> **Key Term: Hallucination** — When an AI makes up information and presents it as fact. E.g., it might cite a research paper that does not exist.

- **Limited context window** — The model can only "remember" a limited amount of text in one conversation.

> **Key Term: Context Window** — The maximum amount of text (measured in **tokens**) that an AI can read and process at one time. 200,000 tokens ≈ about 2 books.

- **Needle in a Haystack Problem** — Even within its context window, the AI can struggle to find one specific fact buried deep inside a very long document.

> **Key Term: Needle in a Haystack** — A benchmark test where a single fact is hidden inside a massive document, and the model must find it. Tests how well the AI attends to specific details in long text.

- **Lack of sourcing** — A plain LLM won't tell you where it got its answer from, making it hard to verify.

Two Ways to Improve LLMs

Think of improvement along two dimensions:

Dimension	Description	Example
Improve the base model	Upgrade to a newer, smarter model	Moving from GPT-3.5 to GPT-4 to GPT-4o
Engineer around the model	Use techniques on top of the model	Better prompts, RAG, agentic workflows

This lecture focuses on the second dimension — making the most out of the model you already have.

---

2. Prompt Engineering

What is Prompt Engineering?

Prompt engineering is the skill of writing better instructions (called **prompts**) to get better results from an AI. A study from Harvard and Wharton showed that consultants who were trained in prompt engineering outperformed those who were not — even when both groups had access to the same AI tool.

> **Key Finding: The Jagged Frontier** — There is a boundary where AI genuinely helps performance (inside the frontier) and where it actually makes things worse because humans rely on it blindly (outside the frontier). Knowing which side you're on matters.

Basic Prompt Design

Bad prompt:

> "Summarize this document."

Better prompt:

> "Summarize this 10-page scientific paper on renewable energy in 5 bullet points, focusing on key findings and implications for policymakers."

The better prompt tells the AI: the topic, the length, the format, and the audience.

Prompt Templates

A **prompt template** is a reusable prompt structure where you fill in specific details depending on the situation.

> **Example:** A company stores user metadata (name, role, language preference) in an HR system. A prompt template automatically inserts this information to personalize every AI response for each user.

- Used in code to automatically scale and personalize AI responses for thousands of users
- Many AI tools (like ChatGPT) already use hidden system-level prompt templates in the background that you don't see

---

Zero-Shot vs. Few-Shot Prompting

These are two fundamental prompting strategies:

Term	Meaning	When to Use
Zero-Shot Prompting	Give the AI a task with NO examples	Simple, general tasks
Few-Shot Prompting	Give the AI a task WITH examples to guide it	Specialized or ambiguous tasks

Example — Zero-Shot:

> "Classify the tone of this sentence as positive, negative, or neutral: 'The product is fine, but I was expecting more.'"

The AI might classify this differently depending on the industry — it's ambiguous.

Example — Few-Shot:

> "Here are examples:
> 'This exceeded my expectations completely.' → Positive
> 'It's OK, but I wish it had more features.' → Negative

> 'The service was adequate.' → Neutral
> Now classify: 'The product is fine, but I was expecting more.'"

Now the AI has context. It will respond: **Negative** — because it learned from your examples what "Negative" means *to your business*.

> **Why Few-Shot is powerful:** You're essentially building a mini dataset directly inside the prompt, aligning the AI to your specific definition of terms — without modifying the model itself.

Chain of Thought Prompting

Chain of Thought (CoT) is a technique where you tell the AI to think step-by-step rather than jumping directly to an answer. This improves performance on complex reasoning tasks.

> **Key Term: Chain of Thought (CoT)** — A prompting method that instructs the AI to break a problem into logical steps before reaching a conclusion.

How to use it:

- Add instructions like "Think step by step" or "Do not skip any step"
- Optionally provide the steps yourself:

  > "Step 1: Identify the three most important findings.
  > Step 2: Explain how each finding impacts policy.
  > Step 3: Write a 5-bullet summary."

Research (2023) showed this method significantly improves AI output quality.

Prompt Chaining

Prompt chaining is different from chain of thought. Instead of one prompt with step-by-step thinking, you break a complex task into **multiple separate prompts** — each handling one part.

> **Key Term: Prompt Chaining** — Splitting a complex task into a sequence of smaller, independent prompts where the output of one prompt feeds into the next.

Single (Unchained) Prompt:

> "Read this customer review and write a professional response that acknowledges their concern, explains the issue, and offers a resolution."

Chained Version:

1. **Prompt 1:** "Extract the key issues from this customer review."
2. **Prompt 2:** "Using these issues, draft an outline for a professional response."
3. **Prompt 3:** "Using this outline, write the full response."

Why chaining is better:

- You can test and improve each step independently
- Easier to find where things go wrong (debugging)

- You can identify which specific step is weakest and optimize just that one
- One downside: **adds latency** (extra time) — not ideal for real-time applications

---

Centaurs vs. Cyborgs (Work Style with AI)

Research found two types of AI users among professionals:

- **Centaurs** — Divide and delegate. Give a big chunk of work to AI, come back later to review. Common in enterprise automation.
- **Cyborgs** — Work in tight back-and-forth loops with AI, never fully delegating. Common among students and developers.

Neither is wrong — it depends on the task and context.

Evaluating Prompts (LLM Judges)

Once you have prompts, how do you know which one is better? There are several evaluation approaches:

- **Manual human rating** — The most accurate but slow and expensive
- **LLM Judges** — Use another AI model to evaluate your AI's output

Types of LLM Judges:

Type	How It Works
Pairwise Comparison	Show the LLM judge two outputs — it picks the better one
Single Answer Grading	Ask the LLM judge to rate one output from 1–5
Reference-Guided	Provide a rubric describing what a "5/5" and a "0/5" output looks like

> **Key Term: LLM Judge** — An AI model used to evaluate the quality of another AI model's output, based on criteria or a rubric you provide.

> **Key Term: Rubric** — A scoring guide explaining what a good answer looks like at each level (e.g., score 5 = concise, 3 key points, clear first sentence; score 0 = verbose and unhelpful).

---

3. Fine-Tuning

What is Fine-Tuning?

Fine-tuning means taking a pre-trained model and training it further on your own specific dataset. Unlike prompt engineering, fine-tuning actually **changes the model's internal weights (parameters)**.

> **Key Term: Fine-Tuning** — The process of taking an existing pre-trained AI model and re-training it on a smaller, specialized dataset so it learns to perform better on a specific task.

Why to Avoid Fine-Tuning (Usually)

The lecturer strongly advises against fine-tuning in most cases:

- **Requires large amounts of labeled data** — Collecting and labeling data is time-consuming and expensive
- **Risk of overfitting** — The model may become great at your specific task but lose its general usefulness
- **Time and cost** — By the time you finish fine-tuning, a newer base model may already outperform your result
- **Hard to maintain** — You can't easily swap in the latest model; your fine-tuned version becomes outdated quickly

> **Funny Example:** A developer fine-tuned a model on Slack messages to "speak like his team." When given the task "Write a 500-word blog post on prompt engineering," the model responded: "I shall work on that in the morning." The model learned to act like a human employee, not follow instructions.

When Fine-Tuning Makes Sense

Fine-tuning is still appropriate in specific situations:

- The task requires **repeated, very high-precision outputs** (e.g., legal briefs, medical reports, scientific formatting)
- The **general-purpose LLM consistently underperforms** on domain-specific language, no matter how well you prompt it
- You have abundant, high-quality labeled data and stable requirements

---

4. Retrieval-Augmented Generation (RAG)

What is RAG?

RAG (Retrieval-Augmented Generation) is a technique that connects an LLM to an external knowledge base (documents, databases, APIs) so it can look up relevant information before answering.

> **Simple definition:** RAG is like giving the AI a reference library to consult before answering, instead of relying only on what it memorized during training.

> **Key Term: RAG (Retrieval-Augmented Generation)** — A method that improves LLM answers by first retrieving relevant documents from an external database and then including them in the prompt so the AI can generate a grounded, sourced answer.

Problems RAG Solves

Problem	How RAG Fixes It
Knowledge cutoff	The knowledge base can be updated anytime
Hallucination	AI answers are grounded in real documents

Problem	How RAG Fixes It
Lack of sources	RAG can return exact document, chapter, and page
Small context window	Only relevant chunks are retrieved, not entire databases

---

How RAG Works — Step by Step

1. **Build a Knowledge Base**
    - You have a collection of documents (PDFs, reports, articles, etc.)
2. **Embed the Documents**
    - Each document is converted into a **vector** (a list of numbers) using an **embedding model**
    - These vectors capture the *meaning* of the document mathematically

> **Key Term: Embedding** — The process of converting text (or other data) into a numerical representation (a vector) that captures its meaning, so that similar texts end up with similar numbers.

3. **Store in a Vector Database**
    - All document vectors are stored in a **vector database** — a specialized database designed for fast similarity searches

> **Key Term: Vector Database** — A database that stores data as vectors (lists of numbers) and allows very fast searching for the most similar items using distance metrics.

4. **User Asks a Question**
    - The user's question is also converted into a vector using the same embedding model
5. **Retrieve Relevant Documents**
    - The system compares the question vector to all document vectors and finds the closest matches — the most relevant documents
6. **Build the Final Prompt**
    - The relevant documents are added into the prompt alongside the user question
    - A prompt template is added: *"Answer the user's question based only on the following documents. If the answer is not in the documents, say 'I don't know.'"*
7. **Generate the Answer**
    - The LLM reads the question and the retrieved documents and generates an accurate, sourced response

---

Advanced RAG Techniques

Basic RAG works well for simple cases, but researchers have developed improvements:

Chunking

- **Problem:** A large PDF embedded as one vector may lose important details buried inside it
- **Solution:** Split the document into smaller **chunks** (e.g., by chapter or paragraph), each embedded separately
- When retrieving, you can get the chunk *and* its parent document for better precision

> **Key Term: Chunking** — Splitting long documents into smaller pieces before embedding, so each piece can be retrieved independently based on relevance.

HyDE (Hypothetical Document Embeddings)

- **Problem:** A user's short question may not have a similar vector to a long document, even if the document answers the question
- **Solution:** First, use the LLM to generate a *fake, hypothetical answer document* based on the question. Then embed that fake document and compare it to the real ones.
- The fake document will be closer in style and structure to the real documents, producing better matches

> **Key Term: HyDE (Hypothetical Document Embeddings)** — A RAG technique where a fake/hallucinated document is generated from the user query and used as the search embedding, instead of embedding the query directly.

RAG vs. Fine-Tuning at a Glance

Feature	RAG	Fine-Tuning
Modifies model weights?	No	Yes
Knowledge updateable?	Yes (update the database)	No (must retrain)
Provides sources?	Yes	No
Best for	Dynamic, changing knowledge	Fixed, precision-critical tasks

---

5. Agentic AI Workflows

What is an Agentic AI Workflow?

An **agentic AI workflow** is a multi-step, automated process where an AI completes a complex task by breaking it into sub-steps, using tools, calling APIs, and making decisions along the way.

> **Key Term: Agentic AI Workflow** — A system where an AI doesn't just answer a single question, but plans and executes a series of actions (using tools, databases, and APIs) to complete an end-to-end task.

> **Why "workflow" and not just "agent"?** The term "agent" means very different things to different people. "Agentic workflow" is more specific — it's a sequence of prompts, tools, and resources organized into a pipeline.

Simple LLM vs. Agentic Workflow — Example

Simple (RAG-based):

> User: "What is your refund policy?"
> AI: "Refunds are available within 30 days of purchase." *(Looks up policy document)*

Agentic Workflow:

> User: "Can I get a refund for my order?"
> Step 1 — AI retrieves refund policy using RAG
> Step 2 — AI asks: "Can you provide your order number?"
> Step 3 — AI calls an API to check the order details
> Step 4 — AI responds: "Your order qualifies for a refund. The amount will be processed in 3–5 business days."

The agentic version is far more useful because it takes multiple actions and personalizes the response.

The Paradigm Shift: Traditional vs. Agentic AI Software

Aspect	Traditional Software	Agentic AI Software
Data	Structured (databases, forms, JSON)	Unstructured (text, images, free-form input)
Logic	Deterministic ("if X then Y")	Fuzzy (AI interprets and decides)
Design thinking	Boxes and microservices	Think like a manager — what roles/agents do I need?
Testing	Clear pass/fail	Harder — subjective quality matters

> **Key Term: Deterministic Software** — Software that always produces the same output for the same input. Traditional code is deterministic.

> **Key Term: Fuzzy Software** — Software where outputs vary and aren't always predictable — like an LLM that interprets natural language differently each time.

> **Human-in-the-Loop** — A design where a human can review, correct, or override AI decisions, especially for fuzzy systems where mistakes are costly. Example: An "appeal" button at the end of an AI-scored assessment.

Core Components of an AI Agent

Think of an agent as having four building blocks:

1. Prompts

The instructions and persona given to the agent. All the prompt engineering techniques from Section 2 apply here.

2. Memory

How the agent stores and retrieves information about the user and past conversations.

Memory Type	Description	Example
Working Memory (Core Memory)	Fast-access, immediately relevant information	User's name, current task
Archival Memory (Long-Term Memory)	Slower-access, stored for rare use	User's date of birth, past preferences

> **Why does this matter?** Every piece of information retrieved from memory adds processing time. If the agent checks your full history on every message, it slows significantly. Working memory keeps the most important things close and fast.

3. Tools

External functions the agent can call to take action in the real world.

- Examples: Flight search API, database lookup, payment processing API, email sender, web search
- The agent reads API documentation and learns how to call them correctly

4. Resources

External data sources the agent can read from (but not necessarily act on directly).

- Examples: A company's CRM, a Google Drive folder, a product catalog

---

Levels of Autonomy

Agents can be designed with different levels of independence:

Level	Description	Example
Low Autonomy	Steps are hard-coded by the developer	Agent always follows a fixed sequence
Semi-Autonomous	Tools are fixed, but agent decides the steps	Agent picks how to use available tools
Highly Autonomous	Agent decides steps AND can create/modify tools	Agent writes code, searches any API, builds its own solutions

---

MCP — Model Context Protocol

MCP (Model Context Protocol) is a protocol introduced by Anthropic that creates a standard way for agents to communicate with external services and other agents.

> **Key Term: MCP (Model Context Protocol)** — A standardized communication protocol that allows AI agents to talk to external tools, APIs, and other agents more efficiently — without needing to hard-

> code each connection manually.

API approach (old way):

- Developer manually writes instructions for every external API
- Does not scale — every new service needs new code

MCP approach (new way):

- A centralized protocol handles communication
- Your agent "negotiates" with the MCP server to discover what it needs
- Scales much better across multiple services and agents

> **Agent to Agent via MCP:** When agents communicate with each other, they treat each other like tools. One agent says: "Here is my input. Here is what I need." The other responds accordingly. This is essentially MCP.

Real-World Example: Credit Risk Memo Agent

From a McKinsey study: A financial institution used to spend 1–4 weeks and 20+ hours of manual work to create a credit risk memo. After introducing agentic AI:

- Relationship manager provides materials to the Gen AI agent
- Agent breaks the task into subtasks and assigns them to specialized agents
- Agents gather and analyze data from multiple sources
- A draft memo is produced
- Humans review and give feedback
- **Result: 20–60% time reduction**

---

6. Evaluating AI Systems (Evals)

Why Evaluation Matters

Building an agentic workflow is only half the job. You need to know if it actually works. **Evals** (evaluations) are the methods used to test and measure the performance of an AI system.

> **Key Term: Evals (Evaluations)** — Structured tests and measurements used to determine how well an AI system is performing, where it fails, and what to improve.

> **Key Term: LLM Traces** — Detailed logs that record every step, prompt, tool call, and response in an agentic workflow. Essential for debugging AI systems.

Without LLM traces, you cannot debug a multi-step AI pipeline — you have no visibility into where it failed.

Four Dimensions of Evals

1. End-to-End vs. Component-Based

Type	Description	Example
End-to-End	Measure overall user experience	"Did the user's address get updated successfully? User satisfaction score?"
Component-Based	Measure each step independently	"Did the database lookup return the correct order ID?"

Component-based evals are easier for debugging because they isolate the problem.

2. Objective vs. Subjective

Type	Description	Example
Objective	Clear right/wrong answer	"Did the agent extract the correct order ID from the user message?"
Subjective	Requires judgment — no single right answer	"Was the agent's tone friendly and polite?"

- Objective evals can be automated with Python code
- Subjective evals need human raters or LLM judges (with rubrics)

3. Quantitative vs. Qualitative

Type	Description	Example
Quantitative	Numbers and metrics	Success rate %, average latency in seconds
Qualitative	Descriptive insights	"Users seem confused by the confirmation message wording"

The Eval Workflow

1. **Error Analysis (Manual Start)** — Read through 20–30 real user interactions. Notice patterns. Are there tone problems? Wrong data lookups? Slow responses?
2. **Design Evals** — Based on problems noticed, create structured tests:
    - Objective tests: Automated Python checks
    - Subjective tests: LLM judges with rubrics
3. **Run Multiple Models Side by Side** — Test 3 different LLMs (e.g., GPT-4, Grok, LLaMA) with the same evals to compare which model performs best on your specific task.
4. **Fix and Iterate** — Adjust prompts, switch models, or improve tools based on results. Measure again.

Practical Case Study: Customer Support Agent

Task: User wants to change their shipping address.

Task Decomposition (how a human would do it):

1. Extract key information (order ID, new address)
2. Look up the customer record in the database
3. Check the policy — is address change allowed?
4. Draft and send a response email

Agentic Design:

Step	Technology
Extract info	Vanilla LLM prompt
Database lookup + update	Database tool
Draft email	LLM prompt
Send email	Email API tool

How to Evaluate It:

- **Objective:** Did the agent extract the correct order ID? Did the address actually update in the database?
- **Subjective:** Was the email tone professional and friendly?
- **Quantitative:** What % of address updates succeeded? What was the average response time?
- **Qualitative:** Are users confused about anything in the confirmation?

---

7. Multi-Agent Systems

What is a Multi-Agent System?

A **multi-agent system** is when multiple specialized AI agents work together to complete a large task. Each agent is an expert in one area.

> **Key Term: Multi-Agent System** — An AI architecture using multiple specialized agents that each handle a specific part of a complex task, often running in parallel.

Why Use Multiple Agents?

- **Parallelism** — Multiple agents can work simultaneously on independent parts of a task, saving time
- **Reusability** — A single specialized agent can be used by different teams in the same organization
- **Easier debugging** — A specialized agent is simpler to debug than one massive agent
- **Specialization** — Each agent is optimized for one job, making it more reliable at that job

---

Interaction Patterns

Pattern	Description	Best For
Hierarchical	An orchestrator agent receives user requests and delegates to specialized agents	Most real-world systems

Pattern	Description	Best For
Flat (All-to-All)	All agents can communicate directly with each other	When agents need to share information between themselves

> **Key Term: Orchestrator Agent** — The top-level agent in a hierarchical system that receives the user's request, breaks it into tasks, and assigns them to specialized agents.

Smart Home Example — Multi-Agent Design

A smart home agent system might look like this:

Agent	Job
Orchestrator	Communicates with the user, coordinates all other agents
Climate Control Agent	Manages temperature based on room, weather, and user preferences
Security Agent	Handles entry, recognizes users, sets permissions per family member
Energy Management Agent	Tracks energy usage, interacts with weather API for efficiency
Grocery Agent	Monitors fridge via camera, orders from Amazon when items run low
Notifications Agent	Sends alerts about system updates and recommendations

The orchestrator is hierarchically at the top. Some agents (e.g., Climate and Energy) may communicate directly with each other in a flat pattern for efficiency.

---

8. The Future of AI

Are LLMs Plateauing?

There is debate in the AI research community about whether LLMs are improving as fast as before:

- **Scaling laws** suggest that more compute + more data = better models (continues to be true)
- However, improvements are becoming harder to spot — recent model versions feel incremental
- The next breakthrough likely requires **architecture innovation** — discovering a new structure beyond the **Transformer** (the design breakthrough from 2017 that enabled modern LLMs)

> **Transformer Architecture** — The fundamental design of modern LLMs, introduced in 2017. It allows AI to process long sequences of words by focusing on relationships between all words simultaneously.

Key Future Trends

- **Architecture Search** — Researchers are actively trying to discover a better architecture than the Transformer. The discovery of such a design could drastically reduce compute cost, similar to how the Transformer changed everything.

- **Multimodality** — AI that can handle text, image, audio, and video together gets better at *all* of them. A model that understands what a cat looks like, sounds like, and how it's described in text is smarter about cats in every modality.
- **Combining Learning Methods** — Just as humans learn through instruction (supervised), observation (unsupervised), and trial and error (reinforcement learning) simultaneously, future AI will blend these approaches more naturally.
- **Falling Cost of Experimentation** — Code is increasingly cheap to write and throw away. This speeds up innovation but can also lead to less stable products.
- **Short Half-Life of Skills** — AI techniques change so fast that the specific methods you learn today may be obsolete in 2–3 years. The key is breadth of understanding so you can learn new specific methods quickly when needed.

Quick Reference Glossary

Term	Simple Definition
Base Model / Pre-trained Model	An AI trained on massive general data, used directly out of the box
Context Window	The maximum amount of text an LLM can process in one go
Hallucination	When AI confidently states something false or made up
Needle in a Haystack	A test to see if an AI can find one fact buried in a large document
Prompt Engineering	The skill of writing better instructions to get better AI results
Prompt Template	A reusable prompt structure with slots for variable information
Zero-Shot Prompting	Giving an AI a task with no examples
Few-Shot Prompting	Giving an AI a task with a few examples to guide response style
Chain of Thought (CoT)	Instructing the AI to reason step-by-step before answering
Prompt Chaining	Breaking a task into multiple sequential prompts, each feeding the next
LLM Judge	An AI model used to evaluate another AI model's output quality
Rubric	A scoring guide describing what good/bad AI output looks like at each score level
Fine-Tuning	Re-training a pre-trained model on your own dataset to specialize it
Overfitting	When a model gets too specialized on training data and loses general ability
RAG (Retrieval-Augmented Generation)	A method connecting LLMs to external knowledge bases for grounded answers
Embedding	Converting text into a numerical vector that captures meaning

Term	Simple Definition
Vector Database	A database storing embeddings, optimized for fast similarity search
Chunking	Splitting documents into smaller pieces before embedding for better retrieval precision
HyDE	A RAG technique that generates a fake answer document to improve search matching
Agentic AI Workflow	A multi-step AI pipeline that plans and executes tasks using prompts, tools, and APIs
Deterministic Software	Code that always gives the same output for the same input
Fuzzy Software	Software where AI interprets input and may give variable outputs
Working Memory	Fast-access memory for information the agent needs constantly
Archival Memory	Slow-access long-term memory for rarely needed information
Tool (in agents)	An external function or API an agent can call to take real-world actions
MCP (Model Context Protocol)	A standard protocol by Anthropic for agents to communicate with tools and other agents efficiently
Orchestrator Agent	The top-level agent that coordinates all other specialized agents
Multi-Agent System	Multiple specialized AI agents working together on different parts of a task
Evals (Evaluations)	Structured tests to measure AI performance and identify weaknesses
LLM Traces	Logs of every step in an agentic workflow — essential for debugging
End-to-End Eval	Measuring the overall final outcome of an AI system
Component-Based Eval	Measuring each individual step or component of a workflow separately
Transformer Architecture	The design breakthrough (2017) that made modern LLMs possible
Multimodal AI	AI that can process multiple types of input — text, images, audio, video
Centaur (AI workstyle)	User who delegates large tasks entirely to AI and reviews results
Cyborg (AI workstyle)	User who works in tight, rapid back-and-forth collaboration with AI
Jagged Frontier	The idea that AI helps performance inside a boundary but can hurt it outside that boundary
Human-in-the-Loop	A system design where a human reviews or overrides AI decisions at key points

End of Notes — Good luck on your exam!