AI, LLMs, and Agents

Presented by Reem Elmahdi

AI Agents vs. LLM Agents

AI Agent

Umbrella term for systems using optimization to choose good actions or solutions across various domains.

LLM Agent

Subset of AI agents powered by large language models, specialized in natural language processing (ChatGPT, Gemini, Llama).

01

Neural Networks

Foundation architecture

02

Transformers

Attention mechanisms

03

Backpropagation

Training optimization

Marketing often uses "AI agents" when actually referring to LLM agents.

Why LLMs Matter

The 2020s Boom

~90%

AI Investment

Recent AI excitement and investment attributed to LLMs

LLMs deliver broad capability across chat, reasoning-like behavior, code generation, summaries, and planning. The key innovation: they output text by predicting the next token at massive scale with rich contextual understanding.

From Autocomplete to LLMs

Core Intuition

1

Autocomplete

Predict the next word from prior words using historical patterns

2

LLMs

Same principle scaled up: longer inputs, deeper models, richer context

Chat-style outputs appear "intelligent" because of statistical regularities learned from vast training data, not true understanding.

How Autocomplete Works

N-grams to Contextual Signals

Conditional Probabilities

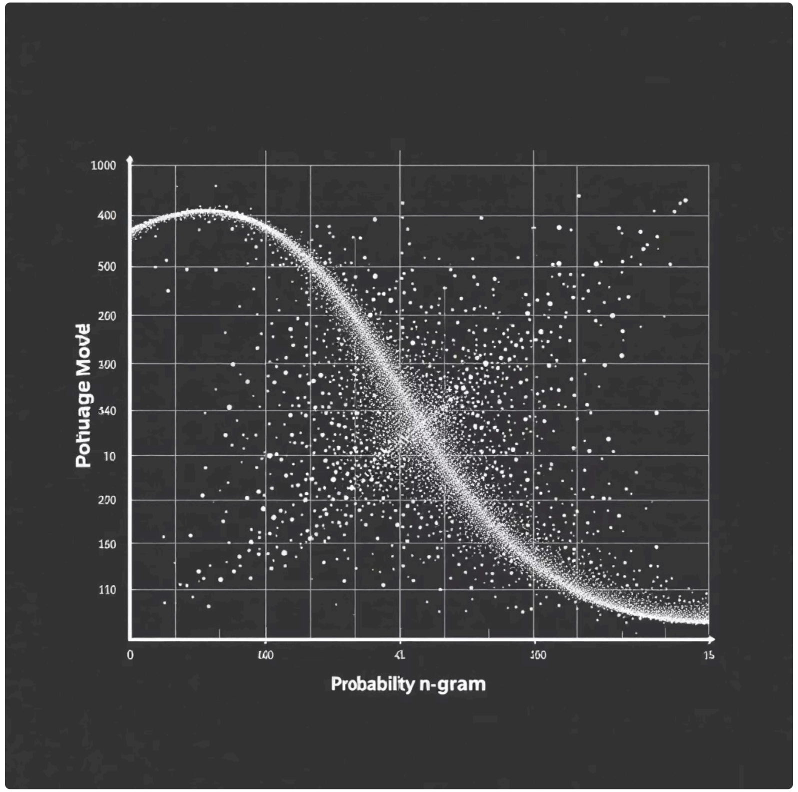
Learns probability distributions over possible next words. Example: "I like to eat" → {pizza: 66%, eggs: 33%} in toy data.

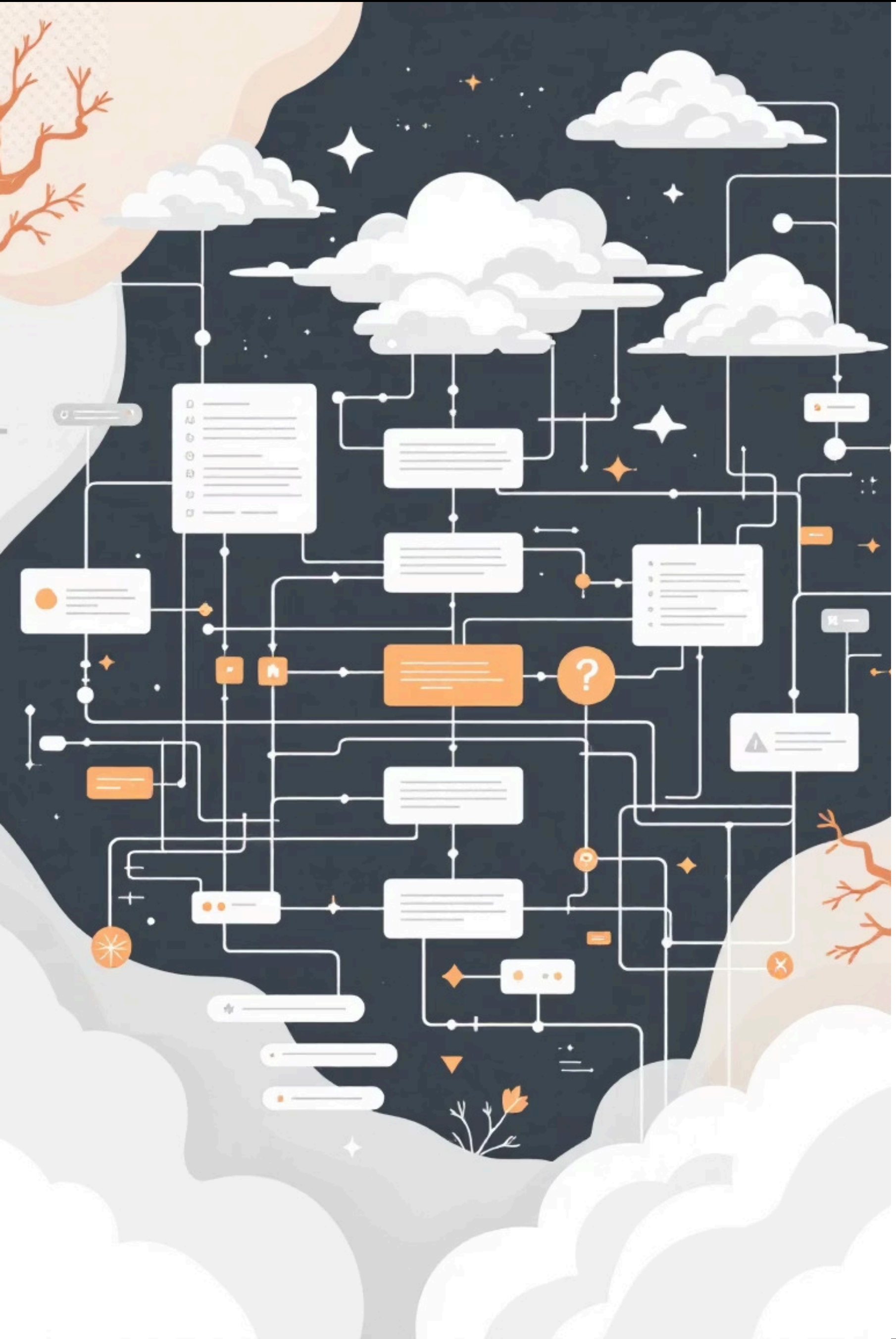
N-grams

Uses common word sequences and frequency patterns to predict next word.

Enhanced Signals

Real systems add recency weighting, location data, user history, and contextual signals to refine predictions.





What Makes an LLM Different

Scale & Tokens

Token Definition

Tokens ≈ pieces of words (~4 chars per token)

Rough conversion: words ≈ tokens × 0.75

Context Window

Legacy: 4,096 tokens ≈ 3,000 words

Modern: up to ~128k tokens

Generation Process

Outputs generated token-by-token, repeatedly conditioning on growing context

Training LLMs

Data, Compute, Parameters

Data

"Chunks of the internet"—web pages, transcripts, articles. More diverse = better performance.

Compute

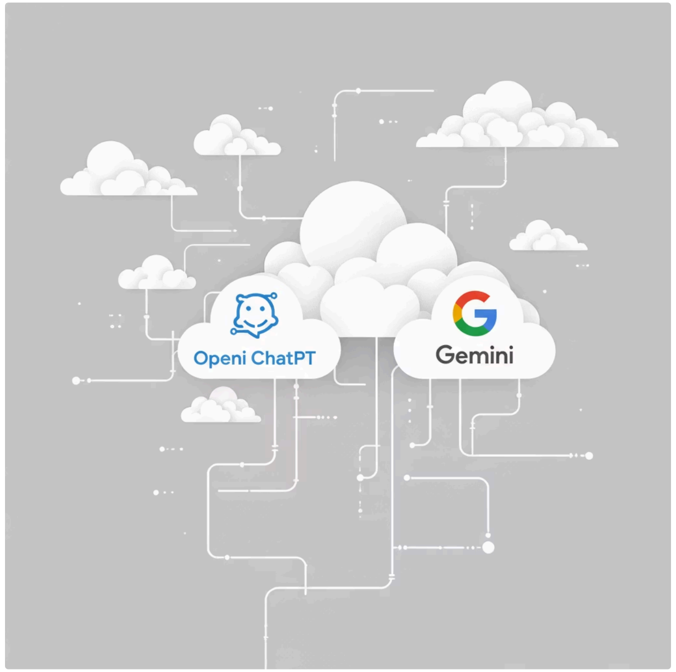
Huge GPU clusters for weeks. Example: Llama 3.1 trained on ~15T tokens using ~16k GPUs for ~54 days (cost: hundreds of millions).

Model Sizes

8B / 70B / 405B parameters. Bigger = generally more capable (but pricier to train and run).

Open vs. Proprietary

How You Use Them



Proprietary

Examples: ChatGPT, Gemini

- Closed weights
- Easy API access
- Often top performance

Trade-offs: Performance & convenience vs. control & cost

Open-Source

Examples: Llama family

- Download & run locally (e.g., LM Studio)
- Control & privacy
- Hardware-heavy for larger models

Prompt Engineering Essentials

Goal + context + role/style → better outputs. Prompts "coach" the model to produce desired results.

Iterative & Model-Specific

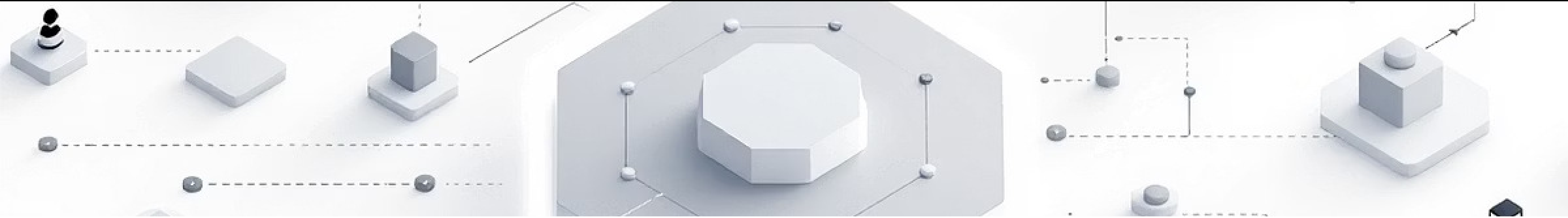
What works on ChatGPT may differ on Llama or Gemini. Experimentation is key.

Common Patterns

- Role prompts ("act as...")
- Constraints and boundaries
- Few-shot examples
- Stepwise instructions
- Evaluation/reflection loops

Community Resources

Curated prompt lists help kickstart—but always tailor to your specific task.



Agentic Design

Multi-Agent Systems

Orchestrate multiple specialized agents that collaborate to solve complex tasks through refined prompts, multi-stage pipelines, tool use (APIs/DBs/web), and conditional triggers.

Coder

Generates and debugs code

Retriever

Queries databases and APIs

Writer/Reviewer

Drafts and refines content

Planner

Coordinates task execution

Benefits: Efficiency, accuracy, enhanced capability, and lower human intervention once the system design (roles + communications) is established.