

Generative AI & Agentic Systems

Day 1: Foundations, Architecture, and The New Paradigm

Shikha Tyagi, M.Tech. IIT Delhi

Founder - AIJamic (AI consulting on building governance first solutions)

Meet Your Instructor: Shikha Tyagi



Shikha Tyagi

IIT Delhi (M.Tech)

Data Science and Agentic Systems Expert

Experience Highlights

- **12+ Years Total Experience** in the AI/ML Domain.
- **3 Years of Specialized Experience** in Generative AI and Large Language Models.
- Extensive background spanning **corporate finance to internet technology**, driving real-world AI adoption.

Previous Industry Contributions

American Express

PayTM

PeopleStrong

Yahoo

Course Roadmap: From Foundations to Capstone

01



Understanding the Fundamental

LLM architecture, history, and core prompt engineering.

02



Making Agents Powerful

RAG & Tool usage

03



Working with Frameworks

LangGraph, and building complex chains.

04



Observability & Fine Tuning

Tracing (LangSmith), evaluation, and custom model tuning.

05



Capstone Project

End-to-end implementation and final presentation.

Day 1 Agenda FOUNDATIONS & ARCHITECTURE

 Introduction GenAI and Agentic AI : AI Evolution and LLM ecosystem

 Prompt Engineering Techniques for consistent results

Lunch Break 1:15PM (40 Minute)

 Hands On Environment setup and first API calls

 RAG Introduction Grounding LLMs with external data

Q & A

Section 1: Introduction to GenAI and Agentic AI

Laying the Groundwork for the Modern AI Stack

- ✓ Evolution of AI: Decoding the AI Landscape
- ✓ Traditional AI vs Generative AI
- ✓ Generative AI vs Agentic AI
- ✓ LLM Ecosystem

The Evolution of AI

From Symbolic Logic to Generative Intelligence

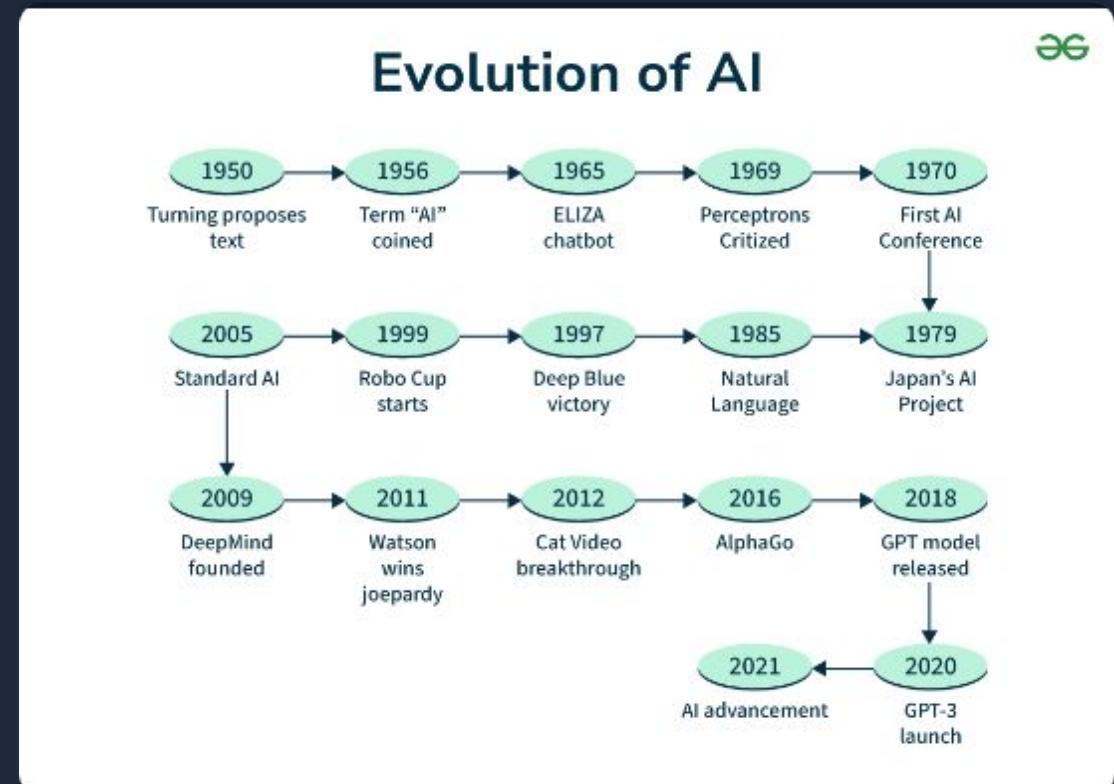


Artificial Intelligence (AI) has been there since?

- 1950
- 2000
- 2022

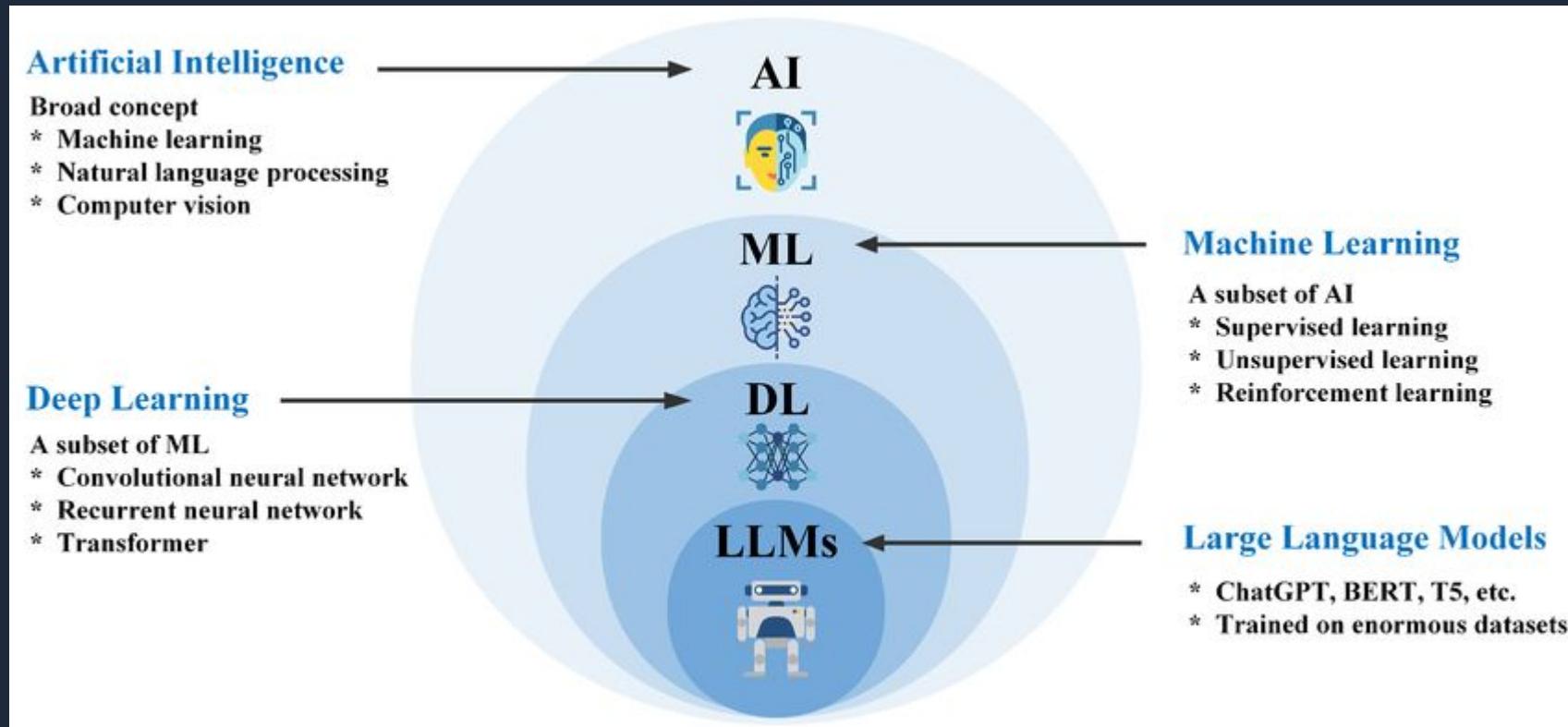
A Brief History of Intelligence

- › **1950s - Symbolic AI:** Rule-based systems and "Good Old-Fashioned AI" (GOFAI). Logic driven.
- › **1990s - Machine Learning:** Statistical methods allow systems to learn from data patterns (Spam filters, etc.).
- › **2012 - Deep Learning:** Neural networks with many layers achieve superhuman performance
- › **2022+ - Generative & Agentic:** Models that generate content and work autonomously.



The Nested Hierarchy: Understanding these terms requires visualizing them as concentric circles, not separate pillars

- › **AI (The Container):** Any machine mimicking intelligence.
- › **Deep Learning (DL):** The subset using neural networks.
- › **Machine Learning (ML):** The subset that learns from data.
- › **LLMs:** GenAI models specialized for *text*.



Defining the Layers



Artificial Intelligence

The broad science of mimicking human abilities. It includes everything from simple "if-then" logic rules to advanced neural networks.



Machine Learning

Algorithms that parse data, learn from it, and then apply what they've learned to make informed decisions.



Deep Learning

A specialized ML technique inspired by the human brain's structure. It uses multi-layered neural networks to learn patterns.



AI has gained a lot popularity in last 2-3 years?

Can you guess, Why?

| AI isn't new : Why the Hype Now?

The Tipping Point: Accessibility



Nov'2022 ChatGPT was introduced by OpenAI

Before, you needed expert skill and Python code to use AI.
Now, you just need English. The Natural Language Interface made AI accessible to everyone.

Let's Dive into GenAI/LLM



Quick Check

- Is GenAI and LLM same?

GenAI vs. LLMs: The Difference

- **Generative AI** is the broad *capability* of creating net-new artifacts (**Images, Music, Video, Text**).
- **Large Language Models (LLMs)** are the specific *tools* (models) designed to master **language and text generation**.

Think of GenAI as "Vehicles" and LLMs as "Sports Cars".

Not All GenAI is an LLM

The "Modality" Key

The primary differentiator is the **input/output modality**.

- **LLMs (GPT-4, Llama 3)**: Focus on text. They predict the next word in a sequence.
- **Image Models (Midjourney, DALL-E)**: Focus on using Diffusion techniques. They are GenAI, but *not* LLMs.
- **Audio Models (Suno)**: Focus on waveforms. Also GenAI, not LLMs.

Technical Comparison

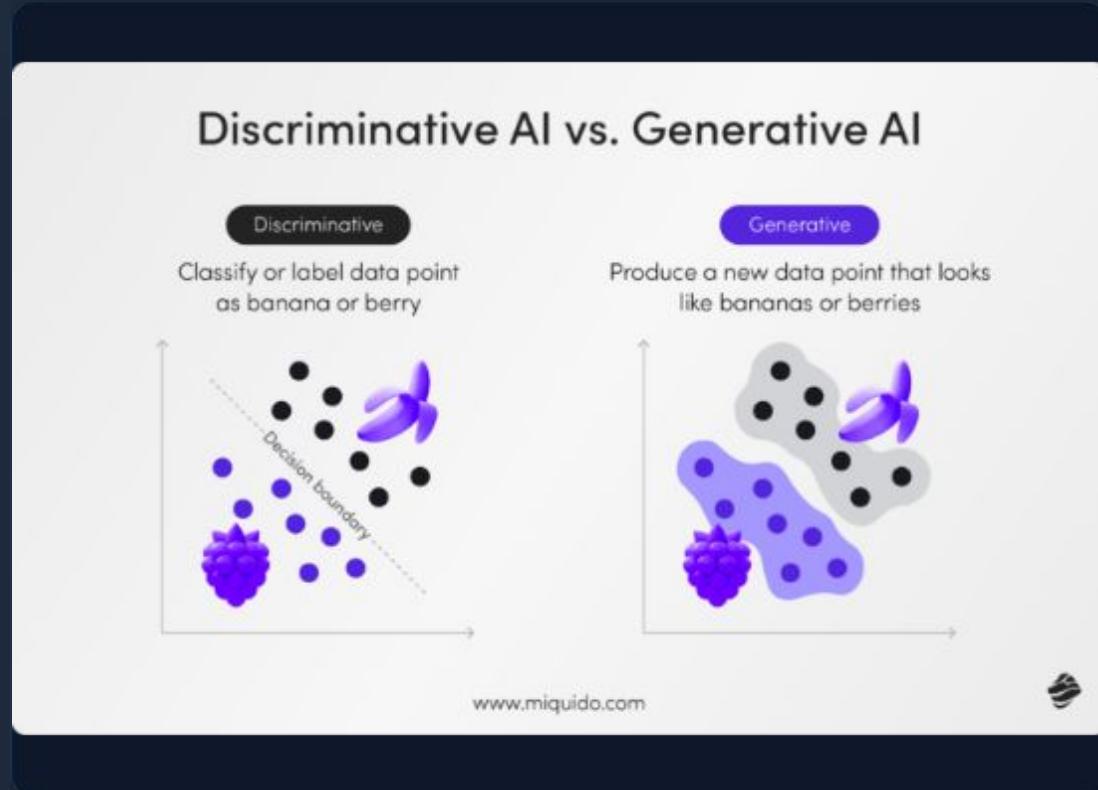
Feature	Large Language Model (LLM)	Generative AI (Broad)
Core Function	Predicting next token (text)	Generating new data distributions
Primary Architecture	Transformers (Attention Mechanism)	Transformers, GANs, VAEs, Diffusion
Training Data	Text, Code, Documents	Text, Images, Audio, Video, 3D
Example Models	GPT-4, Claude 3, Gemini	Stable Diffusion, Sora, Suno AI

Traditional ML vs Generative AI



- Anybody used any traditional AI technique?

Traditional ML vs. Generative AI



Traditional ML

Focuses on **classification** and **forecasting**.

E.g.

Classification: It draws a line between data classes.

$P(y|x)$: Probability of label y given input x .

Generative AI

Focuses on **creation**. It learns the distribution of data to generate new, similar data points.

$P(x)$: Probability of observing input x .

Technical Comparison

Aspect	Traditional Machine Learning	Generative AI
Primary Purpose	Makes predictions or classifications based on existing data	Generates new content such as text, images, code, or audio
Output Type	Numerical values, labels, or decisions	Human-like content and creative outputs
Learning Approach	Learns patterns from labeled or structured data	Learns underlying data distributions to create new data
Examples	Spam detection, credit scoring, price prediction	Chatbots, image generation, code assistants

Quick Check

- Why LLMs are so powerful?

The Source of Power: Unprecedented Data Scale

1. Training on the Internet

LLMs like **GPT** and **Gemini** are trained on **massive datasets scraped from the public internet**, books, code repositories, and specialized data sources.

Example:

GPT-3 used roughly 500 billion tokens from diverse sources:

- Filtered Common Crawl (400B+ tokens)
- WebText2
- Books
- Wikipedia
- This totaled around 570GB of filtered text (or much more raw data).
- This training formed a foundational understanding of language for tasks like ChatGPT.



Massive Data

Corpus

The Source of Power: Unprecedented Data Scale

2. Transformer Architecture

- **Architecture:** The **Transformer** (a Neural Network architecture) is highly efficient at processing this data in parallel.



This combination allows the model to absorb a vast "world model" of facts, linguistic structure, social norms, and common sense—a feat impossible for previous AI models.

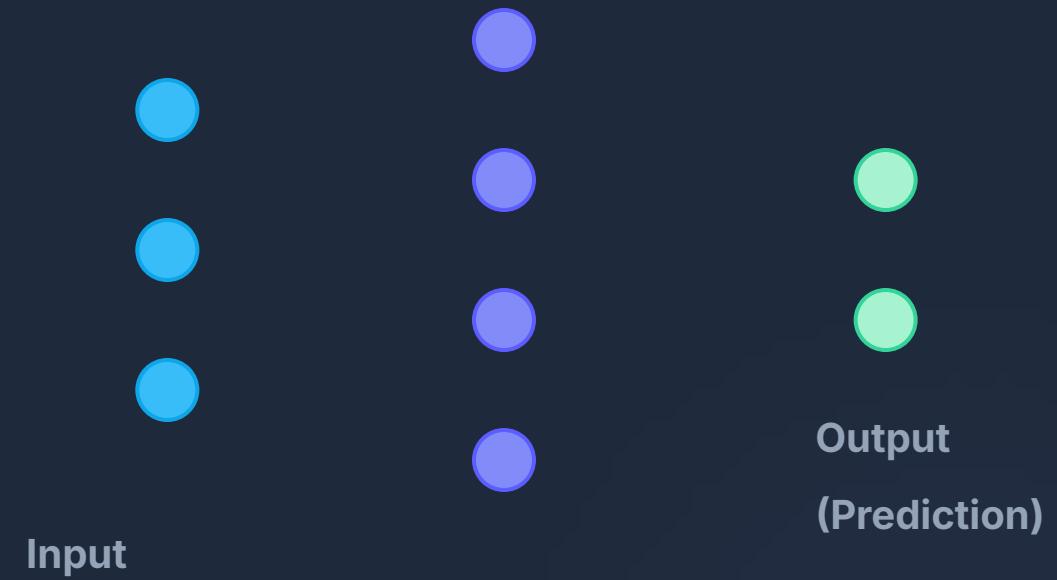
Quick Refresher of Neural Network

Neural Networks: The Computational Brain

Analogy: How LLMs Process Data

The core of every LLM is a complex Neural Network (specifically the Transformer architecture), inspired by the human brain.

- **Input Layer:** Receives tokens (words/code).
Analogous to the sensory input of the brain.
- **Hidden Layers:** The "thought" process where massive calculations (weights/biases) occur.
Analogous to the synaptic connections and gray matter.
- **Output Layer:** Produces the final prediction (the next word).
Analogous to the motor output or verbal response.



Transformers: The Architecture of LLMs

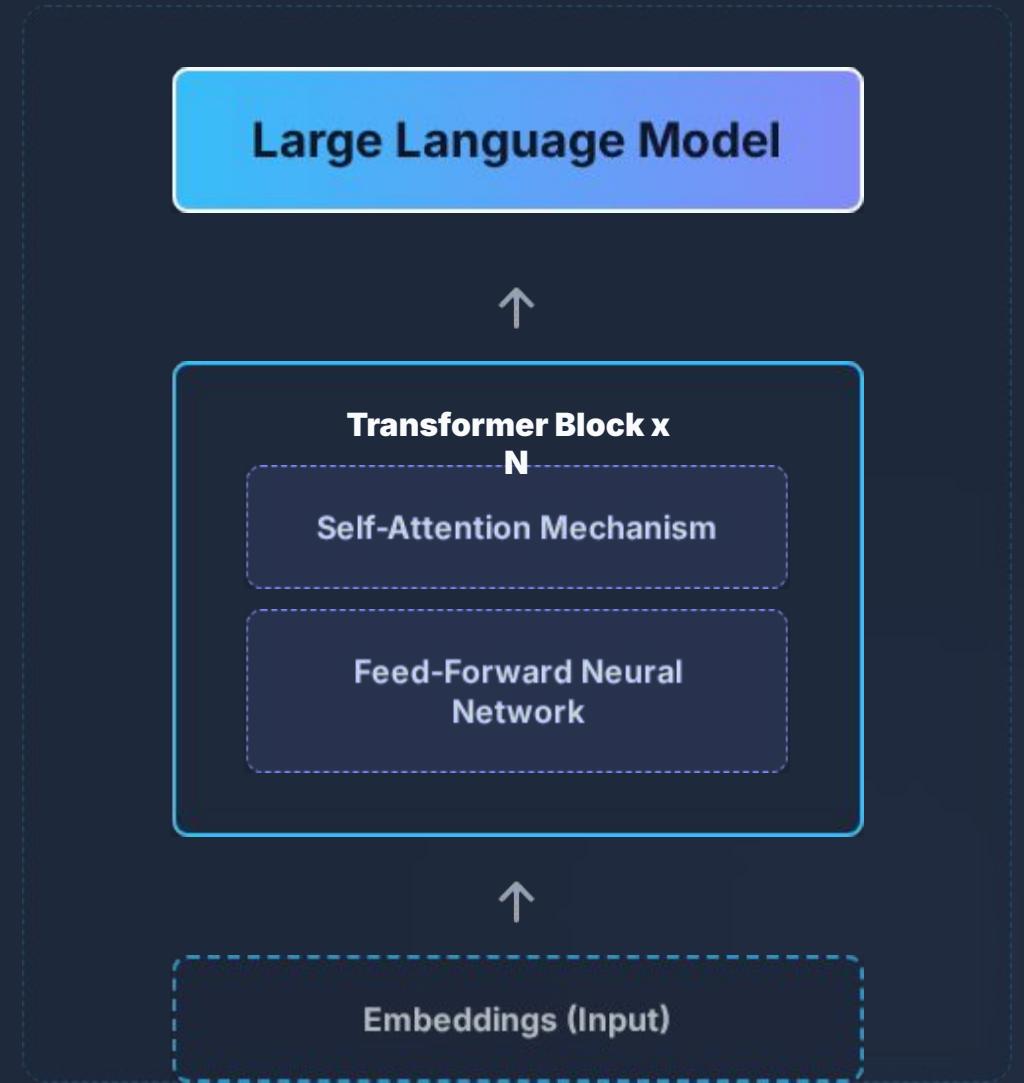
1. Built Using Neural Networks

The Transformer isn't separate from Neural Networks; it is a specific, highly effective **arrangement** of them. It combines:

- **Feed-Forward Networks (FFNs)**: Standard NN layers that process information.
- **Attention Layers**: The key innovation that lets the model "pay attention" to relevant parts of a sentence regardless of distance.

2. The Foundation of LLMs

An LLM (like GPT-4) is essentially a massive skyscraper built by stacking hundreds of these **Transformer Blocks** on top of each other, trained on trillions of parameters.

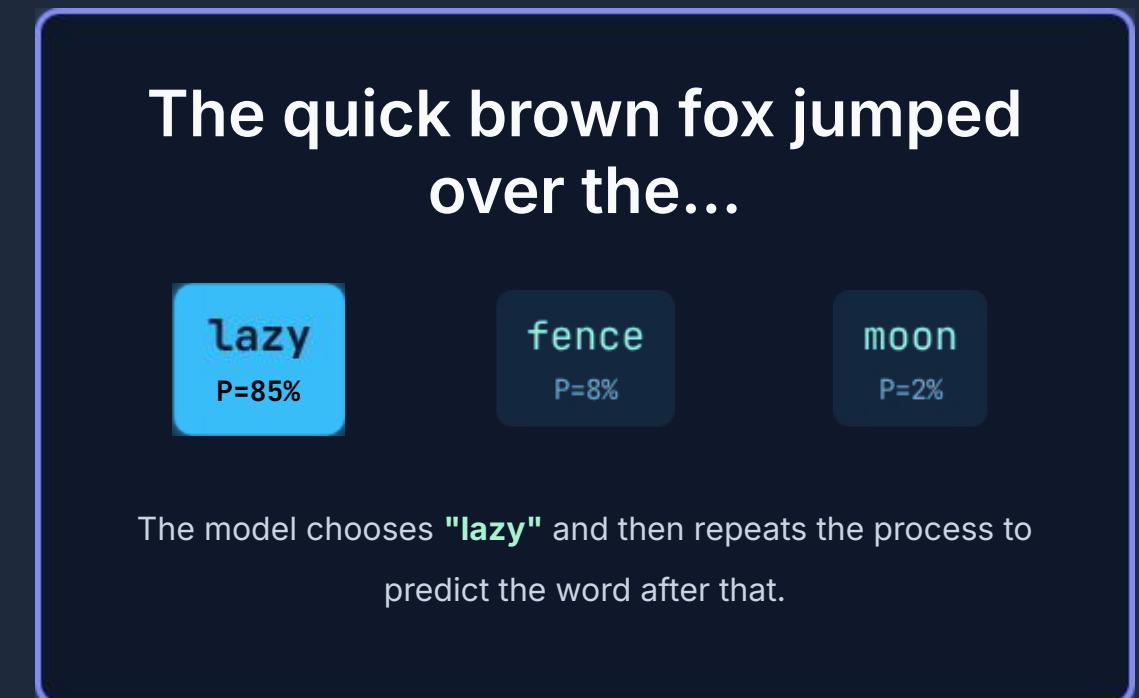


The Core Task: Next Word Prediction

LLMs are Advanced Auto-Complete

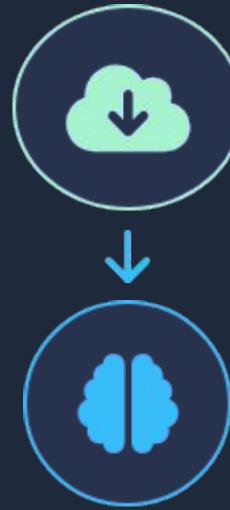
At its heart, a Large Language Model does only one thing, repeatedly: calculate the probability of the **next word (token)** in a sequence.

- **Simple Concept:** Just like your phone suggests the next word you type, LLMs process billions of data points to find the mathematically *most likely* next step.
- **Complex Result:** When this simple task is repeated thousands of times, the model generates complex prose, code, and arguments.



This fundamental mechanism is what allows the model to "speak" convincingly, reflecting the patterns and structures it learned from the internet.

Generalization: The Universal Solver



**Massive Data
Corpus**

**Transformer (LLM
Core)**



Code Generation

Generating, explaining, debugging, and refactoring complex code blocks.



Translation & Paraphrasing

Seamless translation across dozens of languages and tone/style shifting.



Reasoning & Math

Solving multi-step logic problems and complex math (especially with CoT).

ADVANCED CONCEPTS

GenAI vs. Agentic AI

From "Thinking" to "Doing": The Next Evolution



Quick Activity

Open ChatGPT and type:

What is the current temperature in Tokyo? Do not use any tool

Passive vs. Active Intelligence

Generative AI: "Brain in a Jar"

GenAI (like standard ChatGPT) is passive. It waits for a prompt, processes it using frozen knowledge, and outputs text. It cannot "touch" the world.

Agentic AI: "Brain with Hands"

Agents are active. They have tools (APIs, browsers) that allow them to execute code, fetch real-time data, and affect the environment.

Comparative Analysis

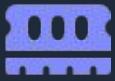
Feature	Generative AI	Agentic AI
Core Loop	Input → Output	Perceive → Think → Act
Environment	Isolated (Context Window)	Interactive (Web, OS, APIs)
Capability	Content Creation	Task Execution
Autonomy	Low (Requires prompting)	High (Goal-seeking)

Key Capabilities of Agents



Tool Use

The ability to interface with external software. Agents can query databases, send emails, or deploy code.



Memory & State

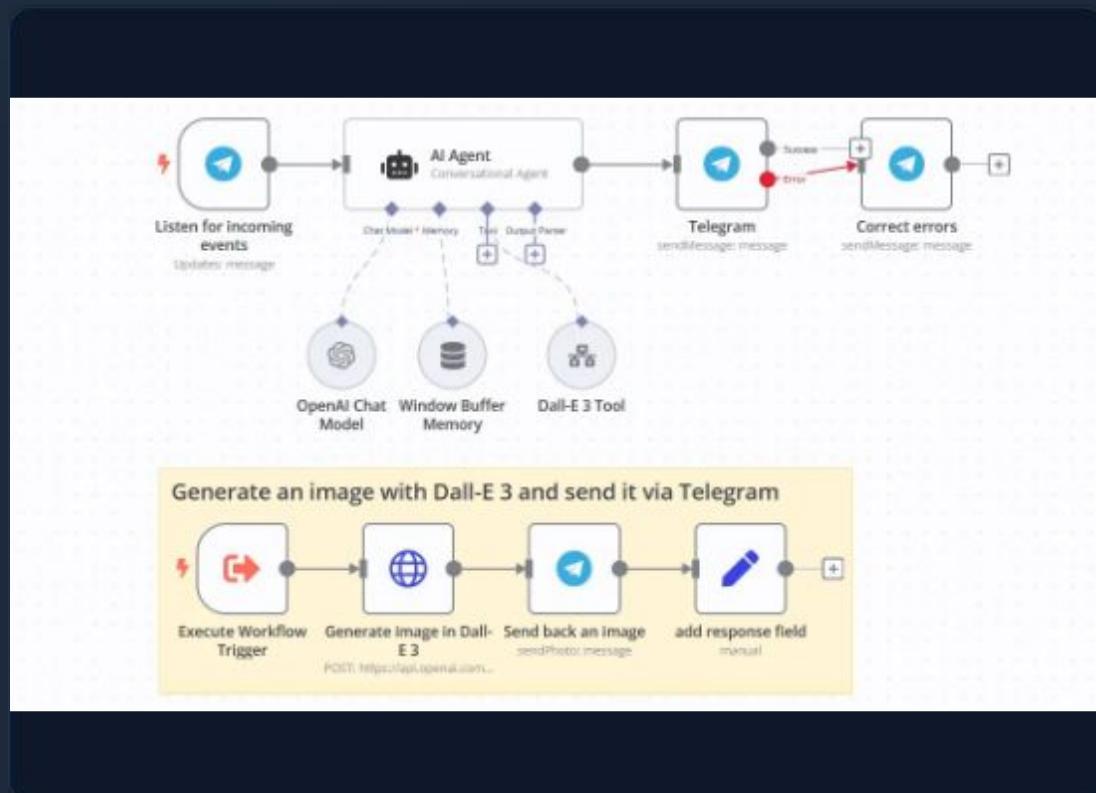
Agents maintain a state over time, remembering past actions and results to inform future decisions in a long-running task.



Planning

Decomposing a complex user goal (e.g., "Plan my wedding") into smaller, sequential, manageable sub-tasks.

Real World Scenario: "Plan a Trip"



The Difference in Outcome

GenAI Response:

"Here is a suggested itinerary for your trip to Tokyo..."
(Text only)

Agentic AI Response:

"I have checked your calendar, found available flights, booked the hotel, and sent the invites to your travel companions." (Action taken)

LLM Ecosystem

Making the "Build" vs. "Buy" Decision



What is Open source vs closed source?

Proprietary vs. Community

The core differences in ownership and deployment

Defining the Two Tiers

🔒 Enterprise/Proprietary Models

Closed-source models owned and primarily served via API by a single company.

- **Focus:** General intelligence, robustness, and state-of-the-art performance.

GPT-4 (OpenAI)

Gemini (Google)

Claude (Anthropic)

🔓 Open Source/Community Models

Model weights and architecture are publicly available, allowing full self-hosting and modification.

- **Focus:** Customization, privacy, cost control, and fine-tuning.

Llama (Meta)

Falcon

Mistral

The Enterprise Decision Matrix

Feature	Enterprise (GPT, Gemini)	Open Source (Llama, Mistral)
Performance (SOTA)	Highest, immediate access to cutting-edge research and largest models.	Rapidly catching up, but often a step behind the largest proprietary models.
Cost Model	Pay-per-token API cost; scales linearly with usage. High fixed cost if usage is high.	High upfront hardware/infrastructure cost; low marginal cost per query.
Customization/Control	Limited to RAG, prompt tuning, or specific fine-tuning APIs.	Full control over weights, architecture, quantization, and deep fine-tuning.
Data Security / Privacy	Relies on provider's guarantees (e.g., data not used for training). Data transmitted externally.	Data remains fully on-premise or in your private VPC. Maximum security.
Deployment Speed	Fastest: Instant API access, low setup friction.	Slow: Requires deep MLOps expertise and significant infrastructure provisioning.

Selecting the Right Path



Choose Enterprise If...

- > You need the absolute highest performance (SOTA).
- > Speed-to-market is the primary driver.
- > You lack dedicated MLOps/GPU infrastructure.
- > Query volume is low or unpredictable.



Choose Open Source If...

- > Data sovereignty (on-premise) is a strict requirement.
- > You need deep, specialized domain fine-tuning.
- > Query volume is extremely high and consistent (cost control).
- > You have the internal MLOps expertise to manage hosting.

MODEL DEPLOYMENT & API CONTROL

Controlling the Output

Key Generation Parameters for LLMs

Tuning the Response

Precision, Creativity, and Constraints

1. Max Tokens (The Guardrail)

What it Controls

The **maximum length** of the generated output, measured in tokens (roughly 3/4 of a word).

Range: 1 to Model Limit (e.g., 8192, 4096, etc.)

Why it Matters

- **Cost Control:** Directly limits API costs, which are typically per-token.
- **Latency:** Shorter outputs generate faster.
- **Flow Control:** Ensures the model doesn't endlessly ramble or hallucinate long, irrelevant passages.

2. Top-P (The Probability Mass Filter)

The model predicts many possible next words, each with a probability.

Top-p says:

“Only consider the smallest group of most likely words whose total probability is at least p.”

▼ What it Controls

Word	Probability
“AI”	0.40
“machine”	0.30
“model”	0.15
“robot”	0.10
“banana”	0.05

If top-p = 0.7

- Take words until total probability ≥ 0.7
- “AI” (0.40) + “machine” (0.30) = **0.70**
- Model samples **only from these two words**

✗ “robot” and “banana” are ignored

If top-p = 0.95

- Almost all reasonable words included
- More diversity, more creativity

3. Temperature

Temperature is a parameter of a function applied to the model's raw scores, which transforms the resulting probability distribution.

What it Controls

The **randomness** or **creativity** of the output by adjusting the probability distribution of the next token selection.

Range: 0.0 (Deterministic) to 1.0/2.0 (Highly Random)

Tuning Guide

- ****Low Temp (0.0 - 0.3):**** Favors high-probability tokens. Ideal for factual extraction, summaries, code generation, and predictable tasks.
- ****Mid Temp (0.5 - 0.7):**** Balances coherence and variation. Good for general chat and structured writing.
- ****High Temp (0.8+):**** Flattens the probability distribution. Best for brainstorming, poetry, creative fiction, and high variation.



Quick Check

When to use temperature closer to 0?

Section 1: Introduction to GenAI and Agentic AI

Laying the Groundwork for the Modern AI Stack

- ✓ Evolution of AI: Decoding the AI Landscape
- ✓ Traditional AI vs Generative AI
- ✓ Generative AI vs Agentic AI
- ✓ LLM Ecosystem





Quick Check:

How do we talk to LLMs?

The Four Components of a Prompt

1. Persona

The role the model adopts to frame its expertise and tone.

2. Task

The explicit action the model must perform (Summarize, Classify, Generate, Translate).

3. Context/Data

The input information or external knowledge the model should utilize.

4. Format/Constraint

Rules governing the output structure, length, or style.

Core Principles for Effective Prompts

◎ Be Specific and Direct

Avoid vague language. The model is a highly skilled intern—it needs explicit instructions, not context clues.

"Write about the moon." → "Write a 500-word academic abstract on lunar regolith composition."

Use Reference Materials

■ If possible, provide the relevant context or data within the prompt (RAG concept). LLMs are better at reasoning over provided text than retrieving facts.

Instruction:

Define Role and Persona

Setting the model's identity dramatically improves tone and relevance. Define *who* it is and *who* the audience is.

"Act as a Senior Financial Analyst," or "Respond as a children's book author."

Specify Format and Constraints

Always tell the model the desired output format (JSON, Markdown, bullet points, maximum word count, etc.).

Constraint:

PROMPT ENGINEERING

Task:

Open chatgpt and write prompt.

- Generate 2 line description of politics
- Act as a comedian and give me 2 line description on politics

Prompt Engineering Essentials



Zero-Shot

Asking the model to perform a task without any examples.
Relies entirely on its pre-trained knowledge base.



Few-Shot

Providing 2-3 examples of inputs and desired outputs in the prompt to guide the model's pattern matching.



Chain-of-Thought

Instructing the model to "think step-by-step." dramatically improves performance on complex logical tasks.

Achieving Structured Output (JSON)

Why JSON Matters

LLM output must be machine-readable (structured) for integration into downstream applications (Agents, APIs, Databases).

Method: Schema and Delimiters

1. **Set the Format:** Use a direct command like "Return the entire response as a single, valid JSON object."
2. **Provide the Schema:** Define the exact keys, value types, and structure required.
3. **Use Delimiters:** Use triple backticks () to separate the instruction from the data and the expected output.

[INSTRUCTION] Extract the person's name and role from the text. Return the output in JSON format with keys "name" and "title". text Jane Doe, the Chief Data Officer, led the meeting. json { "name": "Jane Doe", "title": "Chief Data Officer" }

Prompt for Specific Examples

Classification

Classification

How traditional ML classified text into categories?

Classification

How traditional ML classified text into categories?

- Collect data
- Preprocess data
- Train model
- Validate
- Repeat

Time Taken: Few days to weeks

Classification using Prompt Engineering

Role:Act as a hotel review analyst
Classify the text into neutral, negative, or positive
Text: I think the food was okay.

Coding: Prompt

"""

Table departments, columns = [DepartmentId, DepartmentName]

Table students, columns = [DepartmentId, StudentId, StudentName]

Create a MySQL query for all students in the Computer Science Department

"""

Information Extraction

Your task is to extract model names from machine learning paper abstracts. Your response is an array of the model names in the format `[\"model_name\"]`. If you don't find model names in the abstract or you are not sure, return `[\"NA\"]`

Abstract: {input}

Summarization

- Basic

“Summarize the following text in 5 bullet points.”

- With Constraints

“Summarize the text in under 100 words, preserving key facts and conclusions.”

- Audience-Aware

“Summarize this document for a non-technical executive audience.”

- Focused

“Summarize the text, focusing only on risks, outcomes, and recommendations.”

Reasoning & Analysis

Step-by-Step

“Analyze the problem step by step and explain your reasoning.”

Pros & Cons

“List the advantages and disadvantages of adopting this solution.”

Decision Support

“Based on the information, recommend the best option and justify your choice.”

Do you know what is hallucination?

Do you know what is hallucination?

<https://flyingbisons.com/blog/hallucinations-of-chatgpt-4-even-the-most-powerful-tool-has-a-weakness>

Strategies to Mitigate Hallucination

Grounding

Restrict the model to a defined corpus of documents (your internal data). If the answer isn't in the provided text, the model should state it doesn't know.

Set Strict Boundaries

Explicitly instruct the model not to invent information. "Only use facts provided in the [CONTEXT] section."

Chain-of-Thought (CoT)

Force the model to show its work using the phrase "Thought". This improves intermediate reasoning and exposes faulty logic.

Tune Parameters

Set generation parameters to low (0.0 to 0.3) or a temperature tight (0.8 to 0.9) to reduce randomness and favor high-probability tokens.

Quick Check

Do you know what is system prompt?

The AI's Rulebook vs. Your Requests

System Prompt

Role: The Director / The Constitution

This is the "hidden" instruction that sets the model's behavior, persona, and constraints. It persists throughout the conversation.

Use this to define WHO the AI is.

API Payload Example:

```
{  
  "role": "system",  
  "content": "You are a helpful data science  
tutor. Explain concepts simply."  
}
```

User Prompt

Role: The Actor's Queue / The Query

This is the dynamic input from the end-user. It drives the specific output for the current turn in the conversation.

Use this to define WHAT the AI needs to do right now.

API Payload Example:

```
{  
  "role": "user",  
  "content": "Explain the difference between  
L1 and L2 regularization."  
}
```

Lab 01: Environment Setup



Getting Started

Today's hands-on session will cover:

- Setting up Python virtual environments.
- Generating OpenAI API keys.
- Your first API call

FOUNDATION & GROUNDING

From Generation to Grounding

Why LLMs Need Retrieval-Augmented Generation (RAG)

**Open ChatGPT
Type “What is EOS”**

What is Context Window?

Why GenAI Alone is Not Enough

LLMs possess powerful reasoning and language synthesis capabilities, but they operate under critical constraints:

- › **Knowledge Cutoff:** Their knowledge is fixed at the time of pre-training (stale data).
- › **Private Data Barrier:** They cannot natively access secure, proprietary, or internal enterprise documents.
- › **Hallucination Risk:** When uncertain, they invent plausible-sounding but false facts based on learned linguistic patterns.



The Problem: Unreliable Memory

LLMs are excellent synthesizers, but poor fact databases. For high-stakes, data-driven applications, we must provide the facts ourselves.

If the answer requires current or proprietary information, the LLM will fail or hallucinate.

The Solution: Retrieval-Augmented Generation (RAG)

💡 What is RAG?

RAG is a design pattern that connects an LLM to external, up-to-date, or private data sources to ground its responses.

It works by: **Retrieving** relevant information from a knowledge base, **Augmenting** the user's prompt with that context, and then letting the LLM **Generate** the final answer.

This shifts the LLM's role from "all-knowing generator" to "reasoning and summarization engine."



RAG Formula

User Query + Retrieved Context → LLM
→ Grounded Answer



External Data 🔎 Retrieval → LLM Reasoning

The RAG Engine Components (The Data Pipeline)



Chunking

What: Breaking large documents (PDFs, reports) into smaller, manageable text segments (chunks).

Goal: Optimize context for retrieval. Chunks must be small enough to fit in the LLM context window but large enough to retain meaning.



Embeddings

What: Using a specialized model (e.g., Ada, MiniLM) to convert each text chunk into a dense numerical vector.

Goal: Map meaning to space. Semantically similar chunks are mapped close to each other in the vector space.



Vector Stores

What: A database optimized for storing these numerical vectors and enabling fast similarity searches.

Goal: Efficient retrieval. Calculates the distance (similarity) between the user's query vector and all stored document vectors.

Q & A

Open Floor for Discussion

Image Sources



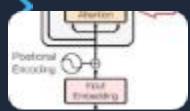
<https://media.geeksforgeeks.org/wp-content/uploads/20240402161005/Evolution-of-AI.png>

Source: www.geeksforgeeks.org



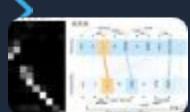
<https://www.miquido.com/wp-content/uploads/2025/01/discriminative-ai-vs.-generative-ai-700x432.png>

Source: www.miquido.com



https://builtin.com/sites/www.builtin.com/files/styles/ckeditor_optimize/public/inline-images/Transformer-neural-network-14.png

Source: builtin.com



<https://www.comet.com/site/wp-content/uploads/2023/07/Screen-Shot-2023-07-11-at-5.43.47-PM.png>

Source: www.comet.com



<https://easy-peasy.ai/cdn-cgi/image/quality=95,format=auto,width=800/https://media.easy-peasy.ai/d2e2d632-61d7-45eb-bc70-46654d70fa7c/3c31939b-bd08-43ce-bc3a-01d6bd713a24.png>

Source: easy-peasy.ai



<https://blog-cdn.skywork.ai/wp-content/uploads/2025/10/b5e0bdcfd65e489eb103175ca959659c-1024x683.jpg>

Source: skywork.ai

Image Sources

>



https://img.freepik.com/premium-photo/close-up-photo-software-developer-coding-intently-laptop-bright-minimalist-office-workspace-with-keyboard-mouse-concept-technology-programming-digital-industry_1962-8500.jpg?w=360

Source: www.freepik.com

Evaluating RAG Correctness Manually

⌘ The Target: Grounded Truth

The goal of RAG is to achieve **Groundedness**. The model should only output information that is directly supported by the context documents we provide.

If the retrieved context is bad, the answer will be bad. If the LLM ignores good context, the answer is still bad. This is a multi-step process.



Manual Evaluation Metrics

- › **Faithfulness:** Does the generated answer contain information verifiable in the retrieved chunks? (Anti-Hallucination Check)
- › **Answer Relevance:** Does the generated answer address the user's query directly? (Anti-Rambling Check)
- › **Context Relevance:** Were the chunks retrieved actually relevant to the question asked? (Retrieval Quality Check)