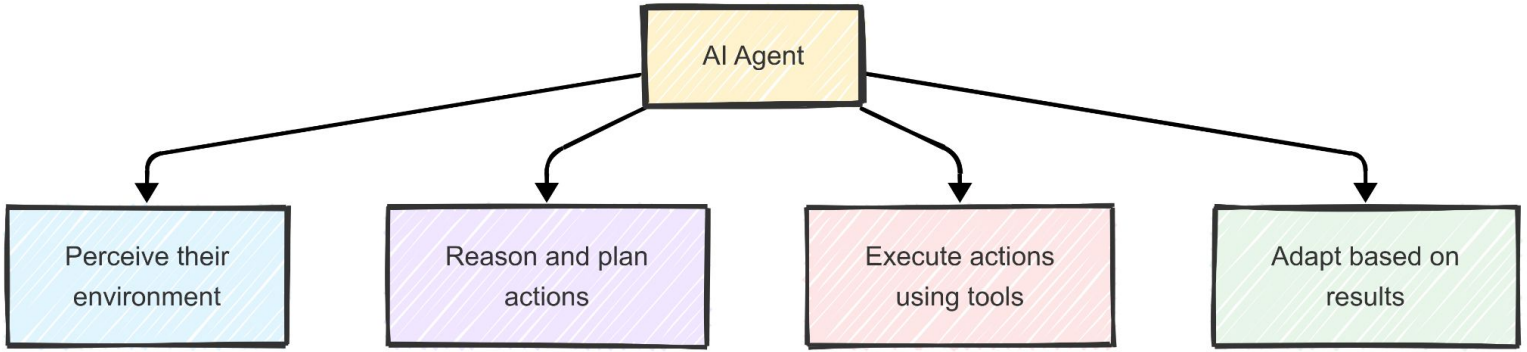
LLMs and Their Role in Agents

Sawradip Saha

Founder, RunAgent
Co-founder & VP of ML, Intelsense AI
30 Sep 2025

Quick Recap - What We Learned



Key Question from Last Class: What powers the "brain" that does the reasoning and planning?

The 5 Agent Types - A Quick Review

Agent Type	Key Feature	Example
Simple Reflex	Fixed rules, no memory	Basic thermostat
Model-Based	Tracks state	Cleaning robot with map
Goal-Based	Plans toward objectives	Route planning
Utility-Based	Optimizes tradeoffs	Smart home system
Learning	Adapts over time	Recommendation engine

The Big Question: How do we make agents smarter, more flexible, and better at reasoning?

What is an LLM?

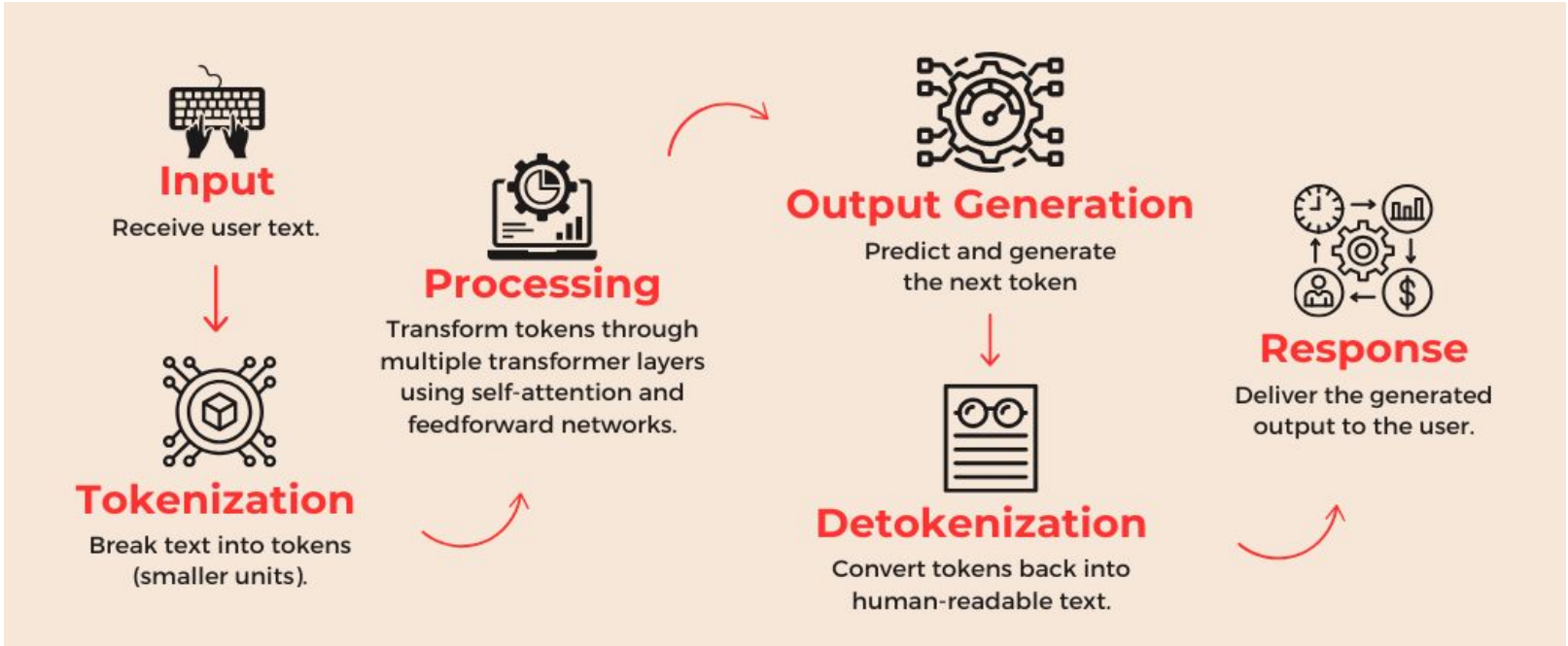
Definition: Large Language Model (LLM) - An AI system trained on vast amounts of text data to understand and generate human language.

Key Characteristics:

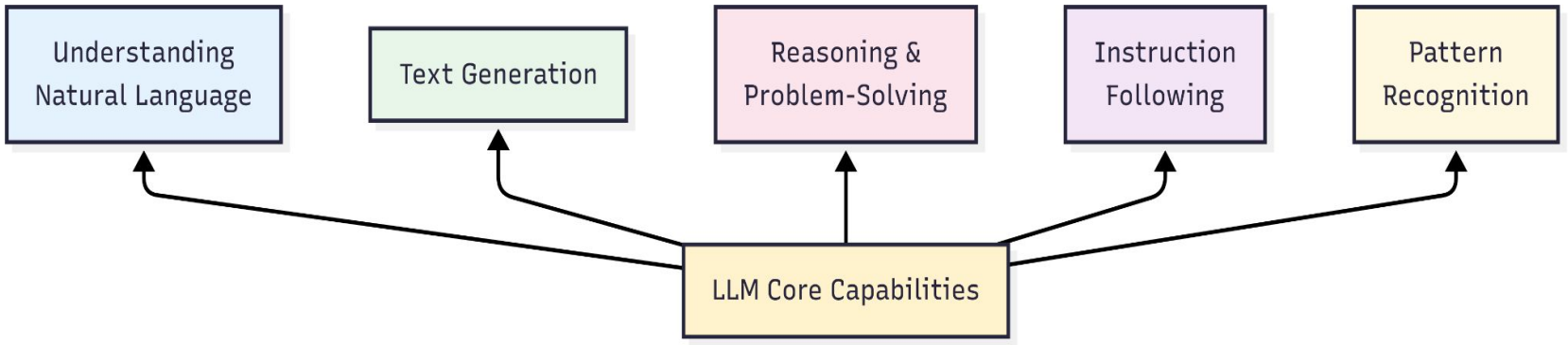
- "Large" = billions of parameters (weights/connections)
- "Language" = trained primarily on text
- "Model" = mathematical representation of patterns in data

Think of it as: A highly educated assistant that has read millions of books, articles, and websites, and can apply that knowledge to understand and respond to your questions.

How LLMs Work



What LLMs Are Good At



The Hallucination Problem

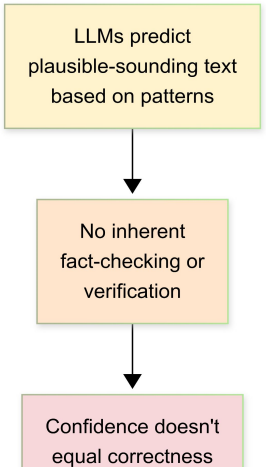
What is Hallucination? When an LLM generates information that sounds plausible but is factually incorrect or completely made up.

Example Scenario:

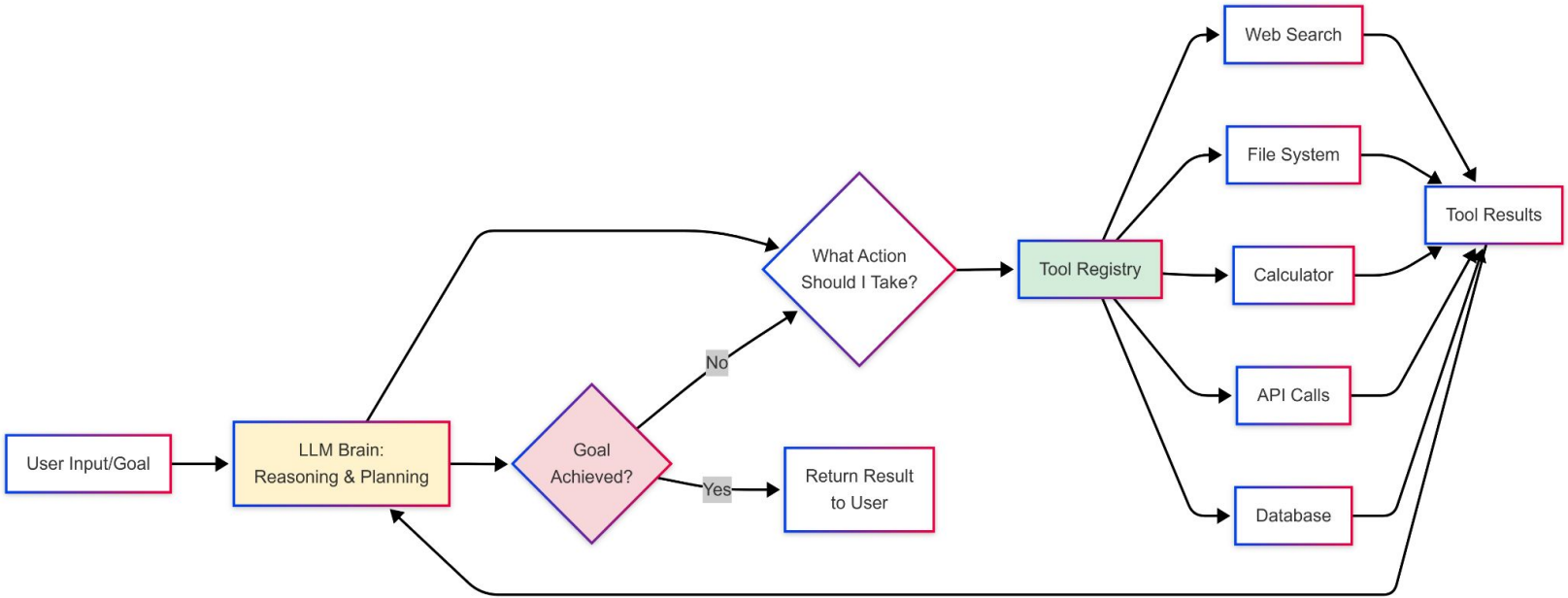
- Question: "What awards did the movie 'Midnight in Dhaka' win in 2023?"
- LLM might generate: "The film won Best Director at the Dhaka Film Festival and received 3 nominations at the Asia Pacific Awards..."
- Reality: **This movie doesn't exist!**

This is a CRITICAL limitation when building agents!

Why This Happens:



Agent Architecture with LLM Brain



Transition to Hands-On

From Theory to Practice

What We've Covered:

- LLM capabilities and limitations
- How LLMs become agent "brains"
- Agent architecture and decision loop

Now: Practical Implementation

Today's Hands-On Goals:

- Set up Python development environment
- Obtain free API access to Google Gemini
- Make your first LLM API call
- Build a simple tool-using agent

Environment Setup - UV Package Manager

What is UV?

- Modern, fast Python package and project manager
- Replaces traditional pip + virtualenv workflow
- Significantly faster dependency resolution
- Single tool for multiple tasks

Why UV Instead of Pip?

- 10-100x faster package installation
- Better dependency resolution
- Built-in virtual environment management
- Modern Python tooling standard

Install From: https://docs.astral.sh/uv/getting-started/installation/#__tabbed_1_2

Virtual Environments Explained

Why Virtual Environments Matter:

- Isolate project dependencies from system Python
- Prevent version conflicts between projects
- Easy to replicate environments across machines
- Clean project management

Virtual Environment Workflow:

- Create project directory
- Initialize virtual environment with UV
- Activate environment (differs by OS)
- Visual confirmation: prompt shows (venv)

Best Practices:

- One virtual environment per project
- Always activate before installing packages
- Include .venv in .gitignore
- Document dependencies

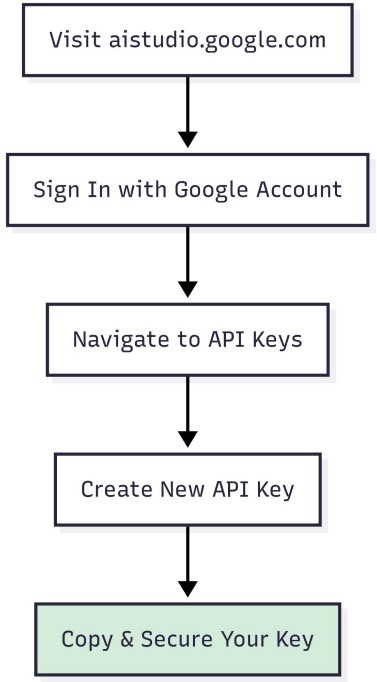
Getting Gemini API Access

Google AI Studio Overview:

- Free tier available for development
- Access to Gemini models (Gemini Pro, Gemini Pro Vision)
- Rate limits: 60 requests per minute (free tier)
- No credit card required

Security Important Points:

- Never commit API keys to version control
- Use environment variables (.env files)
- Add .env to .gitignore
- Rotate keys if exposed



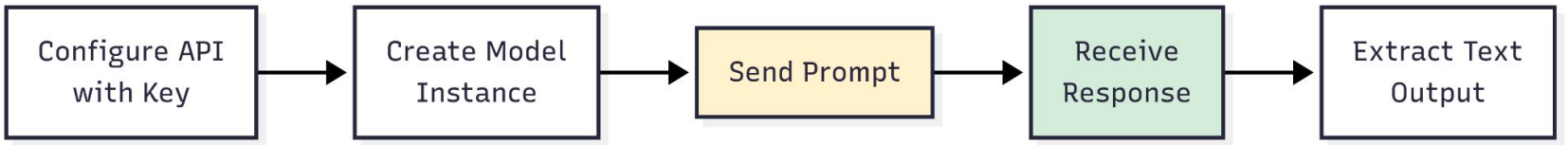
First LLM Call - Key Concepts

Essential Steps:

- Import and configure SDK with API key
- Select model (gemini-pro for text)
- Craft prompt/instruction
- Call generate_content method
- Parse response object

Expected Behavior:

- Synchronous request-response
- Response contains text and metadata
- Latency: 2-5 seconds typical
- Token usage tracked



Thank you! Questions?