LUMS AI SPRING SCHOOL 2026

Building Agentic AI Systems

From Generative Models to Multi-Agent Frameworks

Sher Muhammad Daudpota
Professor of Computer Science
Sukkur IB A University
sher@iba-suk.edu.pk

30-Minute Talk | LLMs · Agentic AI · AutoGen Framework · Case Study – Agentic AI in Higher Education

TALK ROADMAP

30 minutes · 6 sections

01 **LLMs Foundations** — 5 min

Transformers, prompting & capabilities

02 **Agentic AI** — 6 min

Reasoning, planning & tool use

03 **Frameworks** — 5 min

LangChain, CrewAI, AutoGen & more

04 **Case Study** — 7 min

AI agents in education & knowledge domains

05 **AutoGen Deep Dive** — 5 min

Implementation & hands-on demo

06 **Challenges & Future** — 2 min

Evaluation, reliability, ethics

01 | LARGE LANGUAGE MODELS

What is an LLM?

Neural networks trained on massive text corpora using the Transformer architecture (Vaswani et al., 2017).

Key capabilities:

- Text generation & completion
- Question answering & summarization
- Code generation & reasoning
- Multi-lingual & multi-modal tasks

LLM Evolution

2017	Transformer Architecture (Attention is All You Need)
2018	BERT — bidirectional pre-training
2020	GPT-3 — few-shot learning emerges
2022	ChatGPT — conversational AI goes mainstream
2023	GPT-4, Llama 2, Gemini — multimodal LLMs
2024+	Reasoning models, agents & tool-use at scale

01 | LARGE LANGUAGE MODELS

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

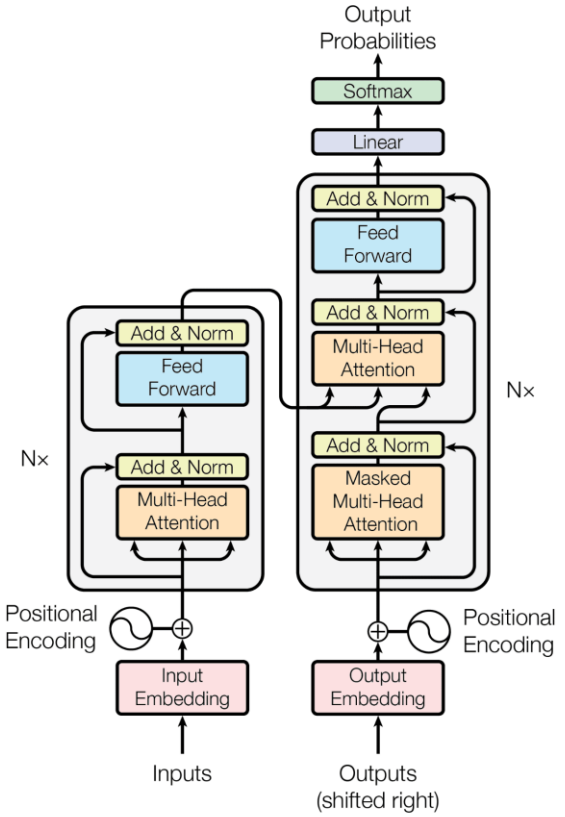
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

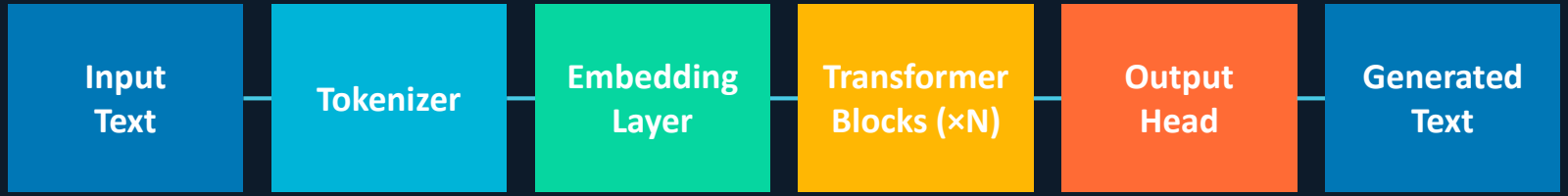
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.



01 | HOW LLMs WORK – RAG, Fine-tuning etc.



Self-Attention

Tokens attend to all other tokens — capturing context across the entire sequence.

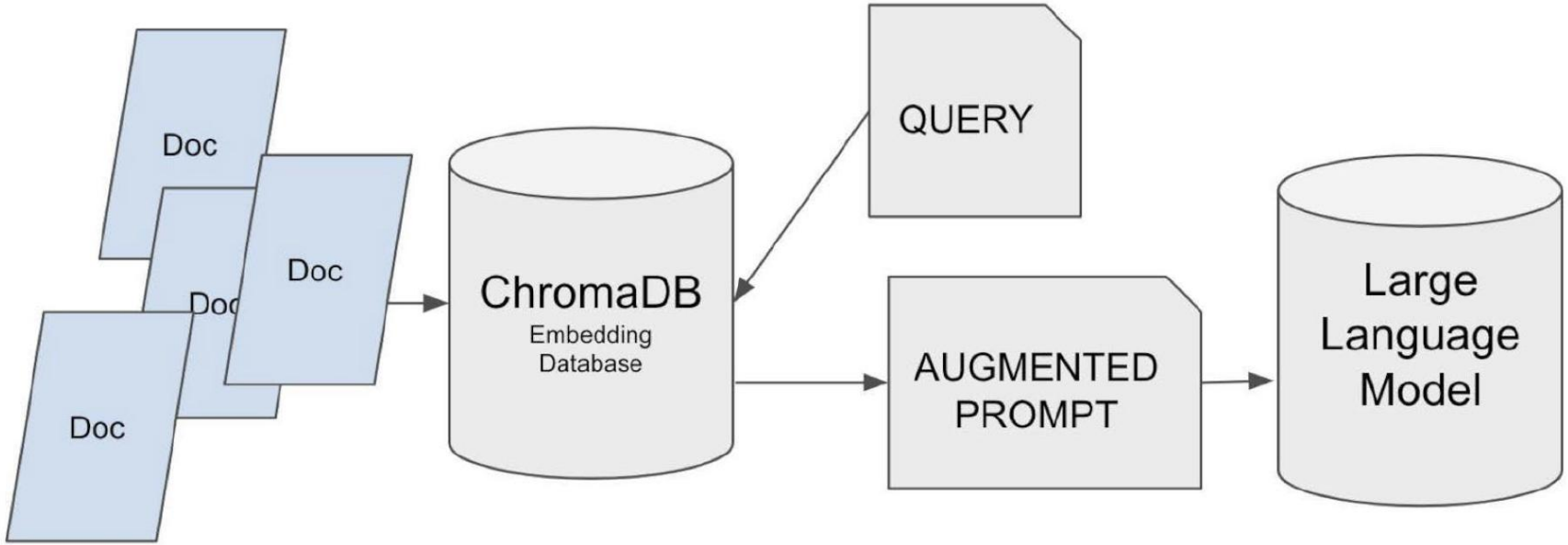
Pre-training

Next-token prediction on trillion-token corpora — learns grammar, facts, reasoning.

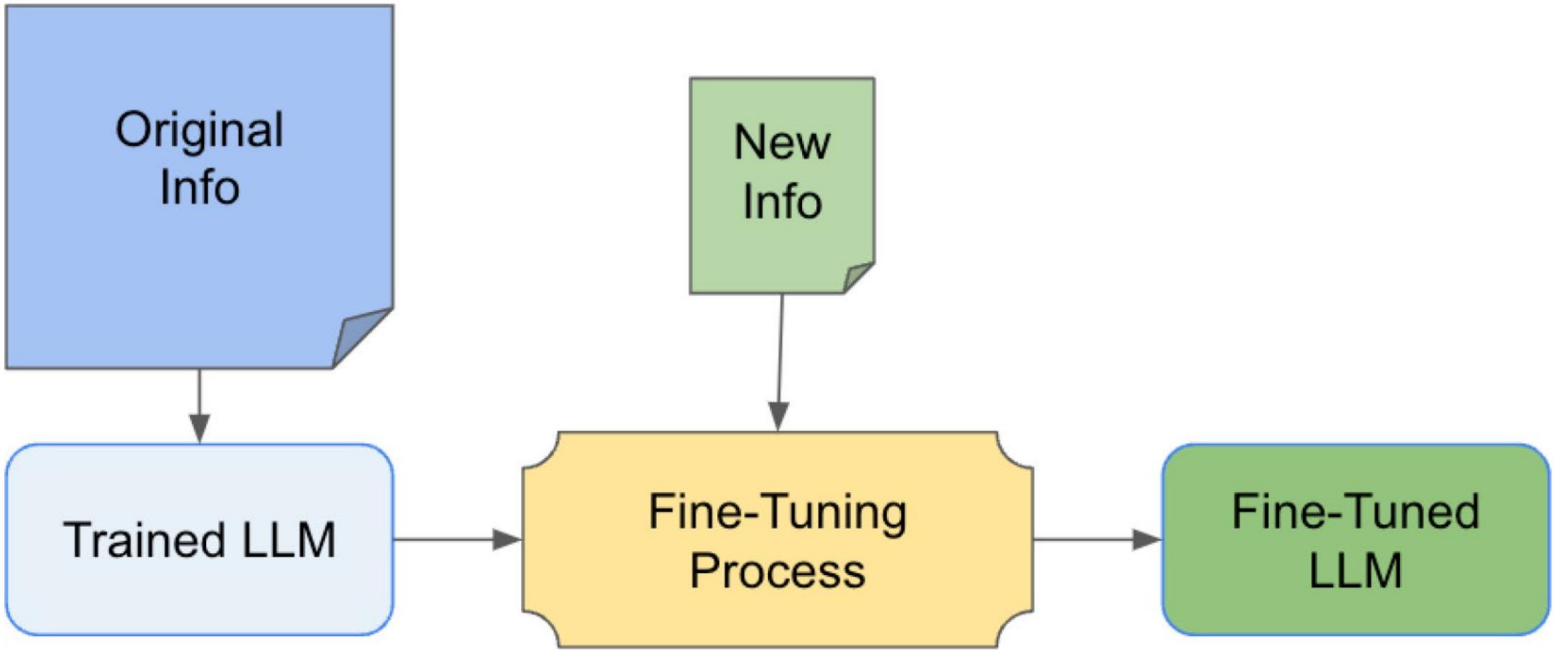
RAG, Fine-tuning & LoRA, QLoRA etc.

Instruction tuning + human feedback aligns models to follow instructions helpfully.

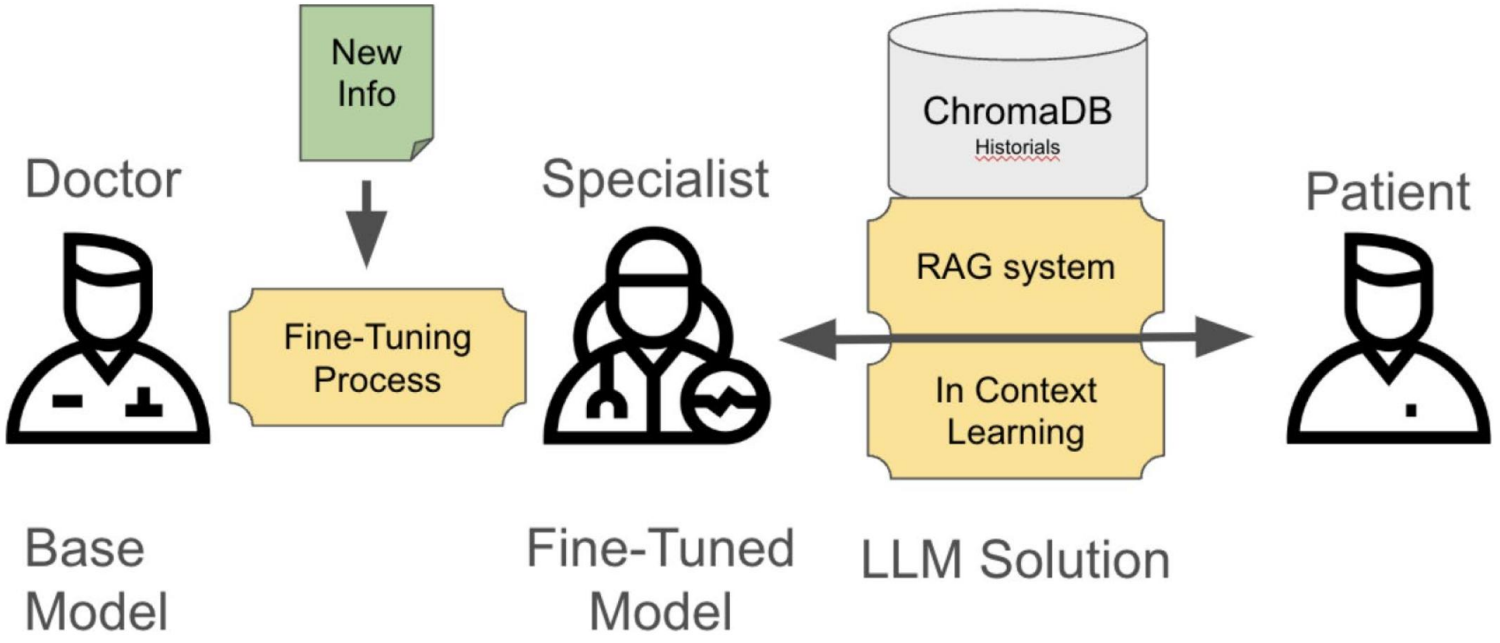
01 | RAG: Retrieval Augmented Generation



01 | LLM Fine Tuning



01 | LLM Fine Tuning



02 | FROM LLMs TO AGENTIC AI

LLM (Standalone)

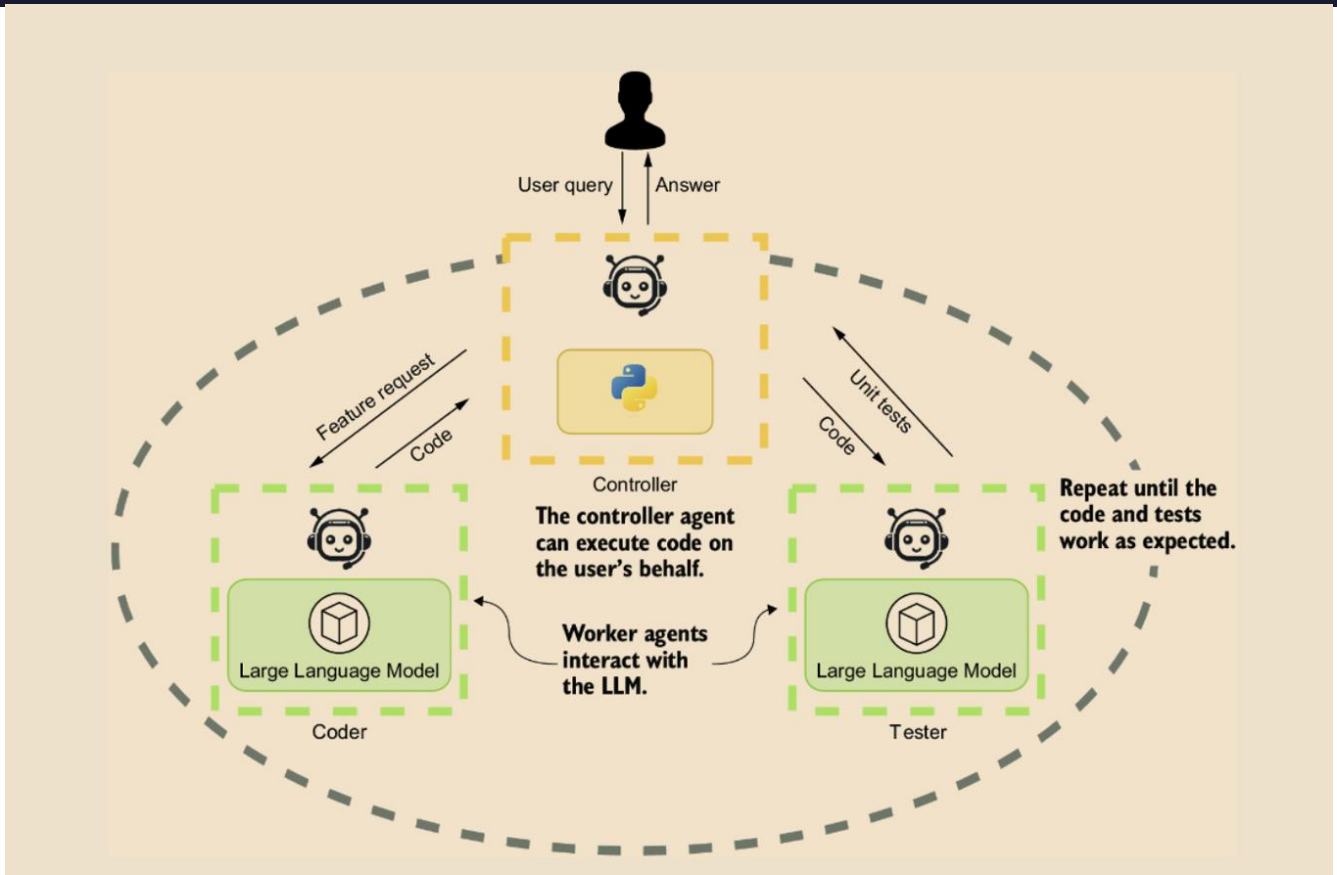
- Single-turn interaction
- No memory between calls
- Cannot take actions
- No tool access
- No planning loop
- Passive — waits for input

Autonomy

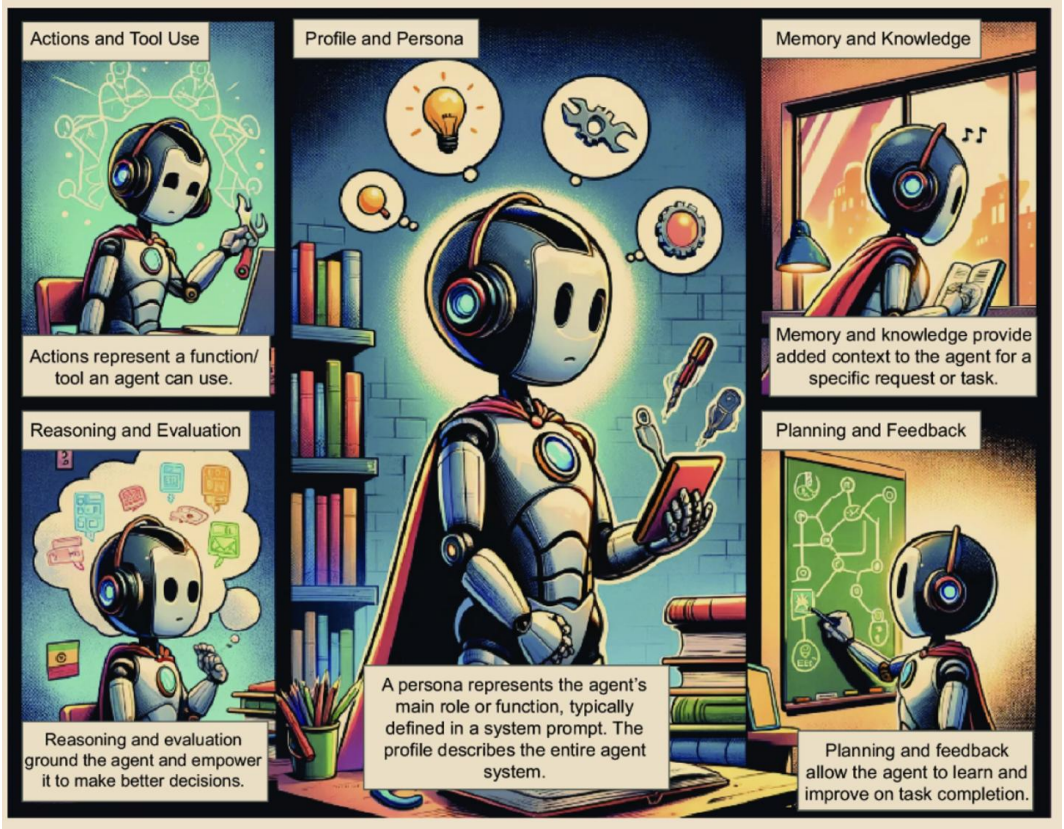
AI Agent

- Multi-turn, goal-directed loops
- Persistent memory & context
- Executes actions in the world
- Tool & API integration
- Plans and self-corrects
- Active — pursues objectives

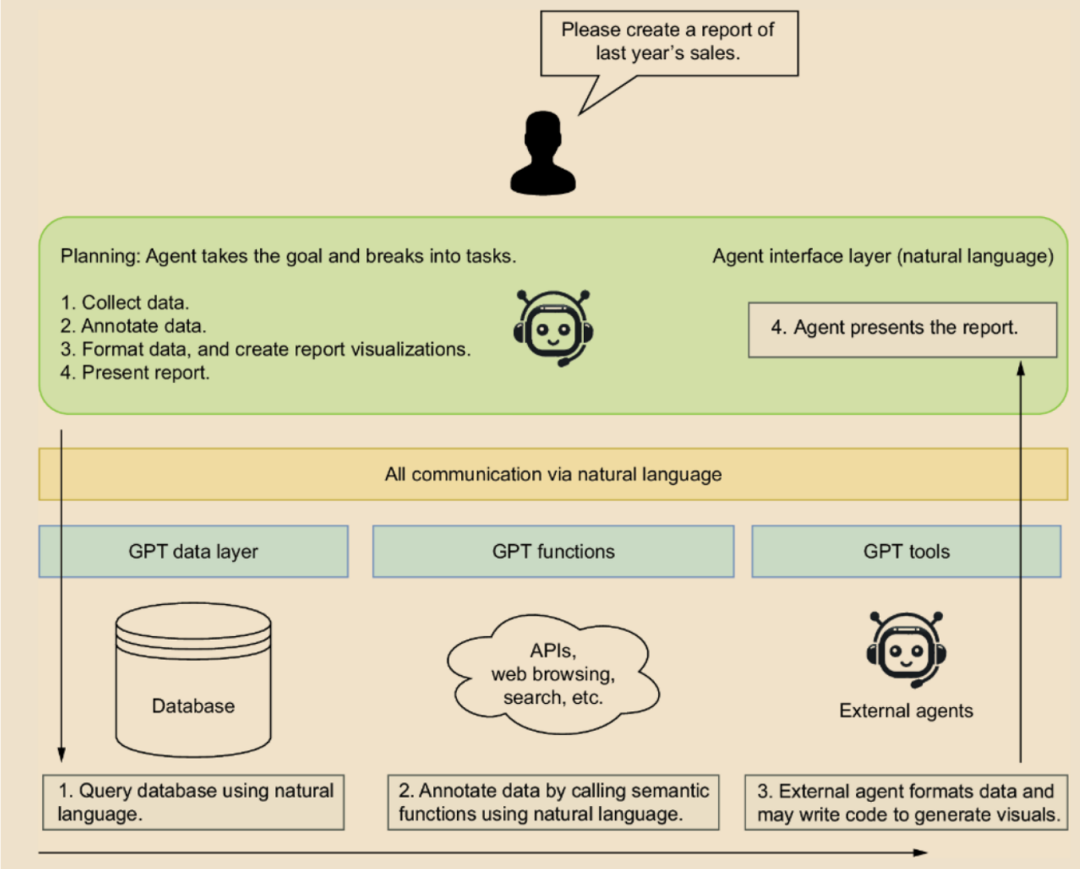
02 | What is an AI Agent?



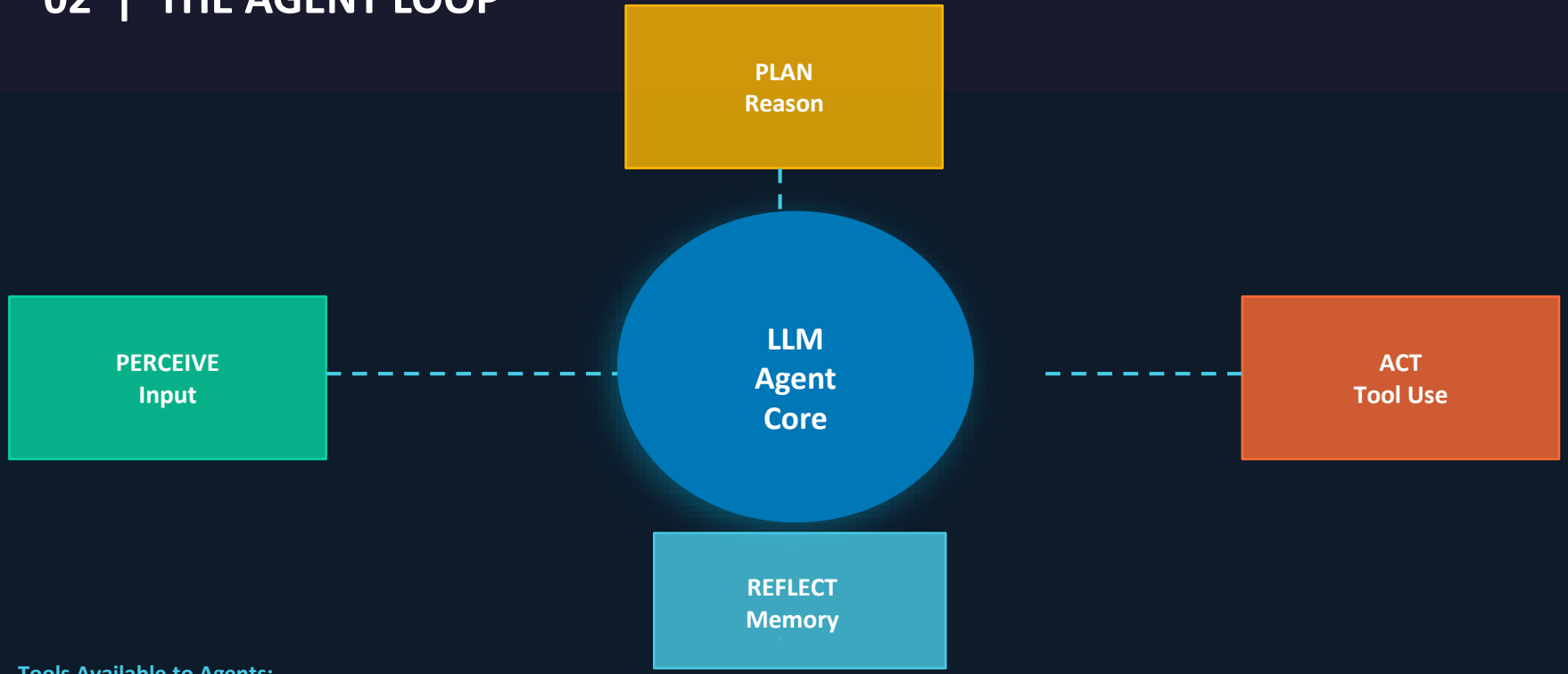
02 | What is an AI Agent?



02 | An Example of AI Agent Works



02 | THE AGENT LOOP



Tools Available to Agents:

- Web Search
- File System
- Code Executor
- APIs & DBs
- Email/Calendar
- Sub-Agents

03 | AVAILABLE FRAMEWORKS

LangChain

Chains, agents & retrieval

Mature, extensive ecosystem. Ideal for RAG pipelines, tool-augmented agents, and memory modules.

✓ Huge community · Rich integrations

▷ Production RAG, chatbots

AutoGen

Multi-agent conversations

Microsoft's framework for orchestrating multiple LLM agents in a conversational loop with human-in-the-loop.

✓ Code execution · Role-based agents

▷ Research, automation, coding

CrewAI

Role-based agent crews

Defines agents as role-playing crew members with assigned tasks. Simple, expressive YAML-based config.

✓ Fast setup · Great for teams

▷ Workflow automation

LlamaIndex

Data-centric agents

Specializes in connecting LLMs to structured & unstructured data. Strong knowledge graph & agentic RAG.

✓ Deep data connectors

▷ Knowledge bases, QA

CASE STUDY

AI Tutoring Assistant

04

A multi-agent system for personalized education

4	RAG	Real-time
Specialized Agents	Knowledge Retrieval	Student Feedback

04 | SYSTEM ARCHITECTURE

05 | AUTOGEN FRAMEWORK

What is AutoGen?

Microsoft's open-source framework for building multi-agent AI systems.

Core Concepts:

- ConversableAgent — base agent class
- AssistantAgent — LLM-powered responder
- UserProxyAgent — human/code executor
- GroupChat — multi-agent orchestration

Key Feature:
Automated back-and-forth conversation until task is complete or human intervenes.

AssistantAgent

AI-powered reasoning, code generation, analysis

UserProxyAgent

Executes code, represents human feedback, tool calls

GroupChatManager

Routes messages between agents in group conversations

Custom Agents

Extend ConversableAgent with domain-specific logic

05 | AUTOGEN: IMPLEMENTATION

Python · AutoGen Multi-Agent Setup

```python
import autogen

# Configure the LLM
config_list = [{"model": "gpt-4", "api_key": API_KEY}]
llm_config = {'config_list': config_list}

# Create Assistant Agent
assistant = autogen.AssistantAgent(
    name="AI_Tutor",
    llm_config=llm_config,
    system_message="You are a helpful tutor..."
)

# Create User Proxy Agent
user_proxy = autogen.UserProxyAgent(
    name="Student",
    code_execution_config={"work_dir": "coding"},
    human_input_mode="NEVER"
)

# Start the conversation
user_proxy.initiate_chat(assistant, message="Explain gradient descent")
```

Install

pip install pyautogen

Agent Config

Each agent has a role, LLM config, and optional tools

Code Execution

UserProxy can auto-run generated Python code in sandbox

Conversation

Agents message each other until termination condition met

05 | AUTOGEN: MULTI-AGENT GROUPCHAT

Python · GroupChat — AI Tutoring System with Multiple Specialized Agents

```python
import autogen

llm_config = {"config_list": [{"model": "gpt-4", "api_key": API_KEY}]}

# Specialized agents
curriculum_agent = autogen.AssistantAgent(name="Curriculum", llm_config=llm_config,
    system_message="Retrieve and structure learning content for the student topic.")

tutor_agent = autogen.AssistantAgent(name="Tutor", llm_config=llm_config,
    system_message="Explain concepts clearly with examples and check understanding.")

assessor_agent = autogen.AssistantAgent(name="Assessor", llm_config=llm_config,
    system_message="Generate questions and evaluate student responses.")

student = autogen.UserProxyAgent(name="Student", human_input_mode="ALWAYS")

# Create GroupChat and Manager
groupchat = autogen.GroupChat(agents=[student, curriculum_agent, tutor_agent, assessor_agent], messages=[], max_round=20)
manager = autogen.GroupChatManager(groupchat=groupchat, llm_config=llm_config)

student.initiate_chat(manager, message="Teach me about Neural Networks")
```

06 | CHALLENGES & CONSIDERATIONS

⚠ Reliability & Hallucination

Agents may generate confident but incorrect outputs. Requires grounding, verification steps, and human oversight.

Evaluation

Hard to measure agent performance. Need task-specific metrics, automated test suites, and human evaluation benchmarks.

Safety & Ethics

Autonomous agents can take unintended actions. Importance of sandboxing, minimal permissions, and ethical guidelines.

Cost & Latency

Multi-agent loops multiply API calls. Optimize with caching, smaller models for sub-tasks, and token budgeting.

Context Management

Long agent loops can exceed context windows. Use summarization, external memory, and selective retention strategies.

Human-in-the-Loop

Deciding when to involve humans is critical. Too much = slow; too little = unsafe. Adaptive oversight is key.

06 | THE FUTURE OF AGENTIC AI

Foundation Models as Reasoning Engines

Models like o1, Gemini Ultra becoming native planners — not just text generators.

Standardized Agent Protocols

OpenAI, Anthropic, Google converging on tool-use standards. MCP (Model Context Protocol) emerging.

Agent Marketplaces

Composable, reusable agents deployed as microservices. Buy, share, and chain agents.

Embodied & World Agents

Physical AI agents (robots, simulations) powered by LLM reasoning — bridging digital and physical worlds.

Key Takeaways

Thank You

- LLMs are the reasoning core powering modern AI agents
- Agency = LLM + Memory + Tools + Planning Loop
- Frameworks (AutoGen, LangChain, CrewAI) simplify building
- Multi-agent systems excel at complex, collaborative tasks
- Safety, evaluation & ethics must be designed in from day 1

Questions? Let's discuss!

Resources & Links

→ **AutoGen GitHub**	github.com/microsoft/autogen
→ **LangChain Docs**	python.langchain.com
→ **CrewAI**	crewai.com
→ **LlamaIndex**	docs.llamaindex.ai
→ **Anthropic (Claude)**	anthropic.com
→ **OpenAI Cookbook**	cookbook.openai.com