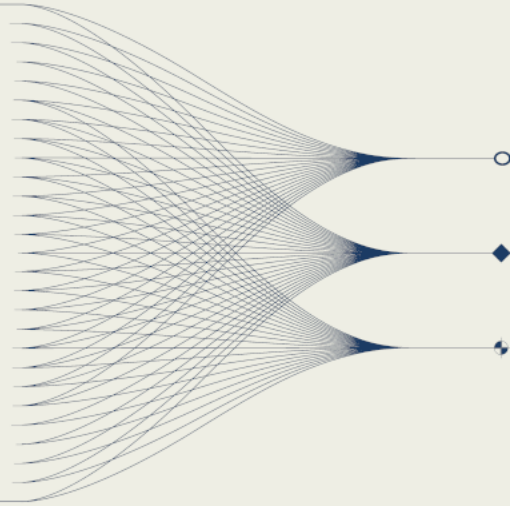
O4.3-565

A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

Comprehensive Nuclear-Test-Ban Treaty Organization (CTBTO), P.O. Box 1200, 1400 Vienna, Austria

CTBTO PREPARATORY COMMISSION | PUTTING AN END TO NUCLEAR EXPLOSIONS



Presentation Date: 10 September 2025

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Contents

- What is a Large Language Model (LLM)?
- Building blocks of LLM applications
- A framework for developing LLM applications
- Capabilities of the framework: RAG, Function Calling, Code Interpreter
- Use Cases: How CTBTO can benefit from LLM applications?
- Exploring future directions

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

What is a Large Language Model (LLM)?

- A large language model (**LLM**) is a type of AI (**generative AI**) that can process and produce natural language text

- LLMs are built on machine learning: specifically, a type of neural network called a **transformer** model
  - **Training** dataset: massive amount of text data such as books, articles, and web pages (~ 15 trillion tokens)
  - Hardware: Trained in a GPU cluster (~ tens of thousands of H100 equivalent GPUs)
  - Post training: Fine-tuned for specific purpose, i.e. instruct using RLHF

- Key Characteristics of LLMs:
  - **Parameters**: Contains billions of parameters (weights) that are fixed during pre-training/fine-tuning
  - **Context Window**: Number of tokens (1 token = 3/4 words) that the LLM can process in one pass (2K - 2M)

- In our framework we consider **off-the-shelf** open-source Large language Models (LLMs):
  - Examples: Llama, Qwen, GPT-OSS, Mistral, Falcon, Gemma, Phi, Yi, DeepSeek, etc.
  - Deployed **locally** in our dedicated GPU infrastructure
  - Multimodal (text, code, vision, image, speech) & reasoning LLMs up to ~ 400 billion parameters

A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

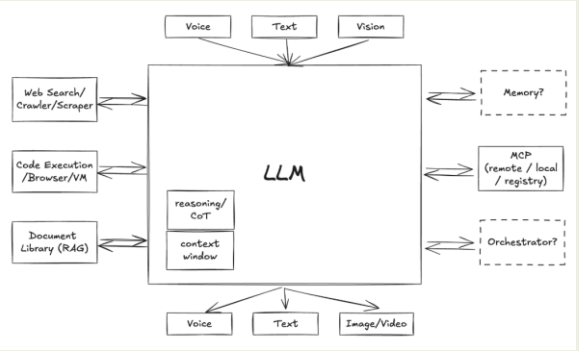
Building blocks of LLM applications

- **The Large Language Model:** A pretrained off-the-shelf open-source LLM or a fine-tuned LLM adapted for specialized use cases

- **InContext Learning**: The capability to learn new knowledge just form the context
  - Context: Instructions (acting role), extra context, output format/style/tone, goals/rules for the task and the query
  - Techniques: Zero-Shot/Few-Shots (provide a few examples to drive the result)

- **Retrieval Augmented Generation (RAG):** Incorporate external information to “ground” the LLM
  - Vector Store: A database to store embeddings, i.e. transformation of “chunks” of documents
  - Similarity Search - kNN: Retrieval of the k most relevant “chunks”
  - APIs/Plugins: Retrieve information from google or internal API, i.e. Jira ticket information

- **Context Engineering:** Programmatically assembling the context for an LLM by combining
  a) *External knowledge* retrieved from documents (RAG) or APIs/Plugins with
  b) *InContext Learning techniques* and then feeding this **constructed context** into the LLM



A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Types of LLM applications

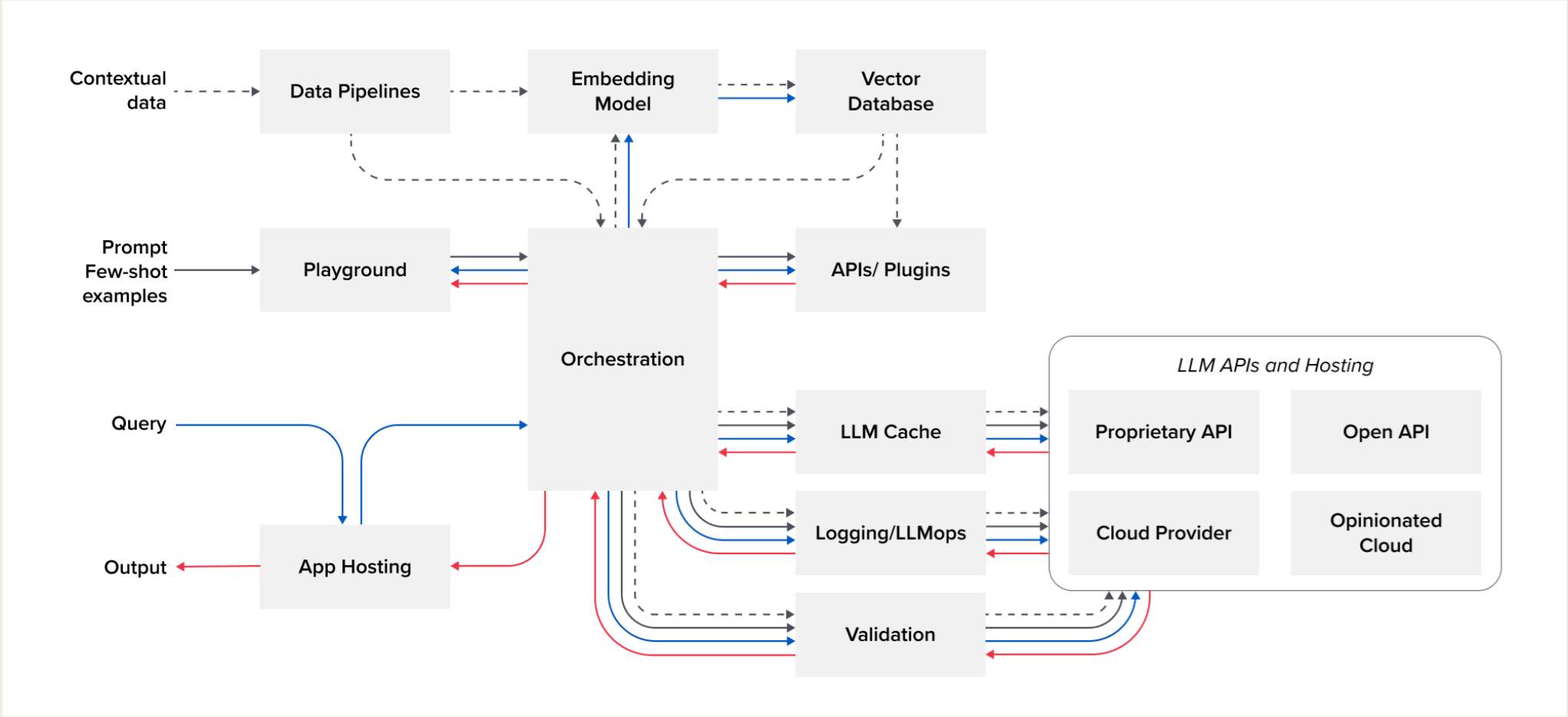
Type	Functionality	Use Cases - Spectrum of Capabilities
AI Chatbots	Dialog-based flow - Applications designed to simulate conversations with human users through text	When integrated with contextual data, they can answer **domain specific** questions and are typically embedded in websites (QnA).
AI Assistants	More advanced than chatbots – Augments work. Can have multiple modalities (text, voice and image)	Can connect with various data sources and systems (like DMS/CMS, Git, DBs, APIs, etc). AI Assistants **automate** routine tasks, by **executing actions** (using tools).
AI Copilots	Auto-completes as you write – Decision support in scientific or coding applications	Specialized fine-tuned models that are **embedded** in scientific or coding software applications (copilot) to assist with specific tasks, benefiting from extensive user data.
AI Agents	Highly **autonomous** systems capable of complex decision-making and learning from their environment.	LLM-based agents can cooperate with other agents, operate in the **background** and waiting for events to occur or tasks to be completed. Examples: Deep Research or SWE-Agent

A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Framework for building LLM applications



Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

LLM Application Stack deployed on NVIDIA DGX-1 Servers

Application	Description
Ollama/vLLM	Open-source LLM **Inference** Engine for running LLMs locally (text LLMs, code LLMs, vision LLMs, embedders)
Qdrant/Neo4j	Open-source **Vector & Graph Databases** to store embeddings, entities, relationships and their metadata for effective Knowledge Search
Flowise	Open-source Low-Code LLM **Orchestration** Tool (based on LangChain and LangGraph)
Open WebUI	Open-source AI **User Interface** with RAG, Code Interpreter, Function Calling, Web Browsing, RBAC, Multilingual
Langfuse	Open-source **Observability** Tool for Inspection of traces, metrics, evaluations and prompt management
LLaMa Factory/Easy Dataset	Open-source **Fine-tuning** and Training Framework for LLMs supporting multiple models
Data Pipelines	Pipelines for parsing, chunking, extracting, injecting, embedding and automatically populate the Vector DB
LiteLLM	An OpenAI-compatible **Gateway** that allows you to interact with multiple LLM providers through a unified API
LLM Playground	Standalone **custom applications** like PDF to Markdown, Whisper (Speech to Text), LightRAG, GPT Researcher, Paper Reviewer, Plagiarism Detector, etc.

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Capabilities of the framework

- Specialized **fine-tuned** models that have been adapted to our needs

- System prompt customization to tune the Assistant's personality and capabilities
  - Provide set of **instructions** and **guidelines** to steer the behavior of the model
  - Adjust parameters of the model like **temperature** and **context length**
  - Leverage In-Context Learning (**ICL**) to learn new skills

- Use of predefined tools & grounding techniques (**RAG**):
  - Custom-made tools via **function calling** (integrate Confluence, Jira, Gitlab, ECS, DOTS, Elasticsearch, etc)
  - Specific **knowledge** from Confluence/SharePoint or IDC pdf documents (IDC Database Schema, IDC Operational Manual, IDC processing of SHI Data, etc)
  - Code Interpreter enables the assistant to write and **run code** in a **sandboxed environment** (containerized)

- All of the above hardcoded in a custom **modelfile** that can be assigned to a person/team

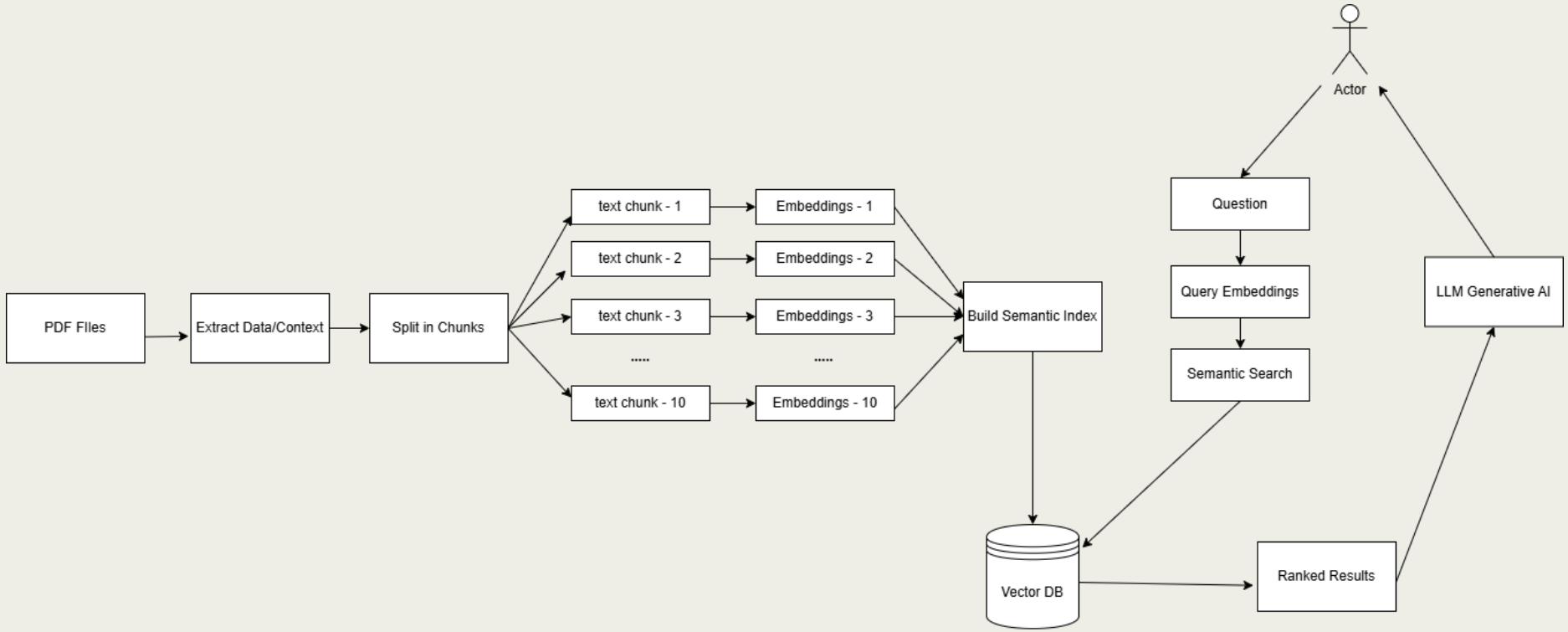
A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

How Retrieval Augmented Generation (RAG) works

- **Retrieval Augmented Generation** (RAG) is the process of retrieving relevant contextual information from a data source and passing that information to a large language model alongside the user's prompt.



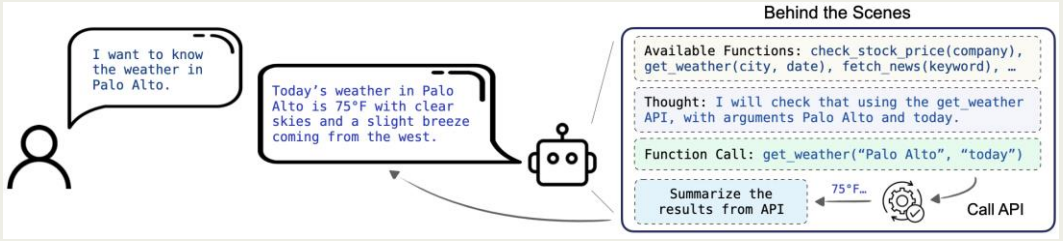
A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

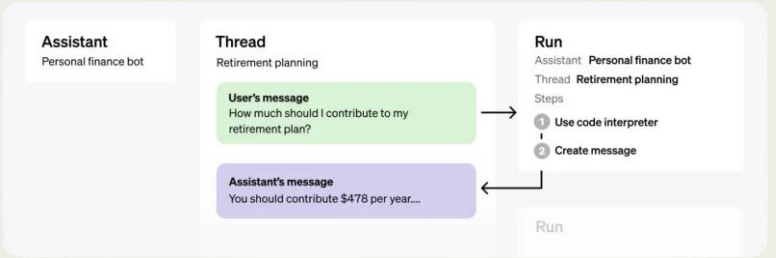
O4.3-565

What is Function Calling, Code Interpreter & Fine-tuning

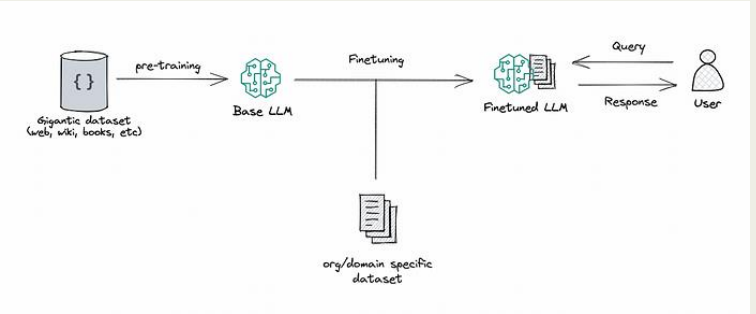
- **Function calling** (tool use) allows you to connect an LLM to external tools and systems, i.e. an AI assistant needs to fetch the latest weather information.



- **Code Interpreter** allows an AI Assistants to write and run Python code in a sandboxed execution environment.



- **Fine-tuning** is a technique used to adapt pre-trained Large Language Models (LLMs) for specific tasks using custom datasets. This process **modifies** the parameters of a pre-trained LLM to create a task-specific model.



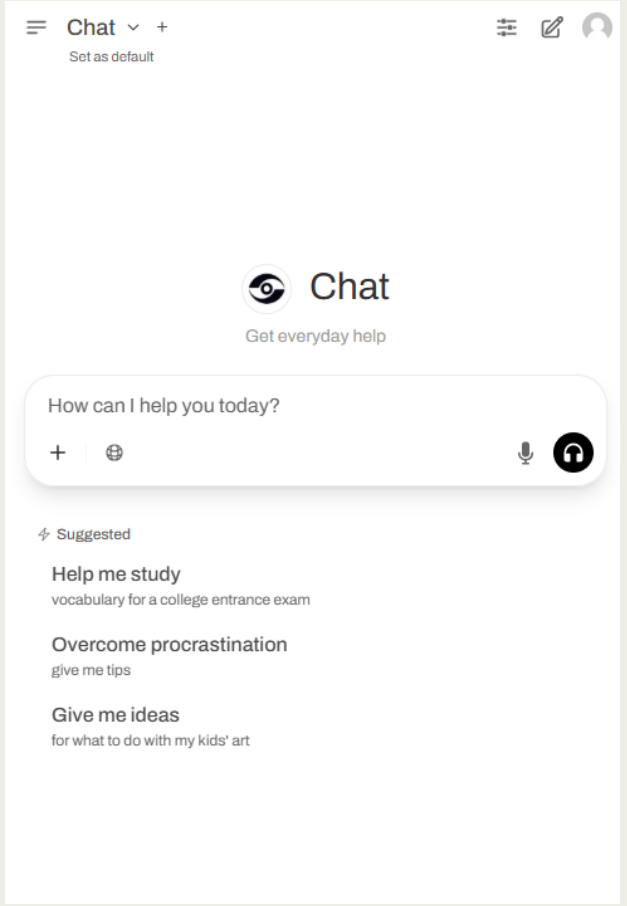
A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Use Case: A general-purpose chat interface

- It is deployed **locally** in our dedicated GPU infrastructure (NVIDIA Servers)
- It is based on popular **open-source** technologies and tools:
  - vLLM (inference), Open WebUI (chat UI), Qdrant (vector DB), Langfuse (observability)
- Capabilities of the Chat UI:
  - Draft email replies, **summarize a document**, create templates for projects
  - Brainstorm ideas and collaborate on a project (using **workspaces**)
  - Prototype an application (using **artifacts**): code snippets, diagrams, or website designs
  - **Extensible** using custom pipelines (RAG), functions (pipes/filters) and tools
  - Create **custom models** with tools/knowledge and assign to specific person/team



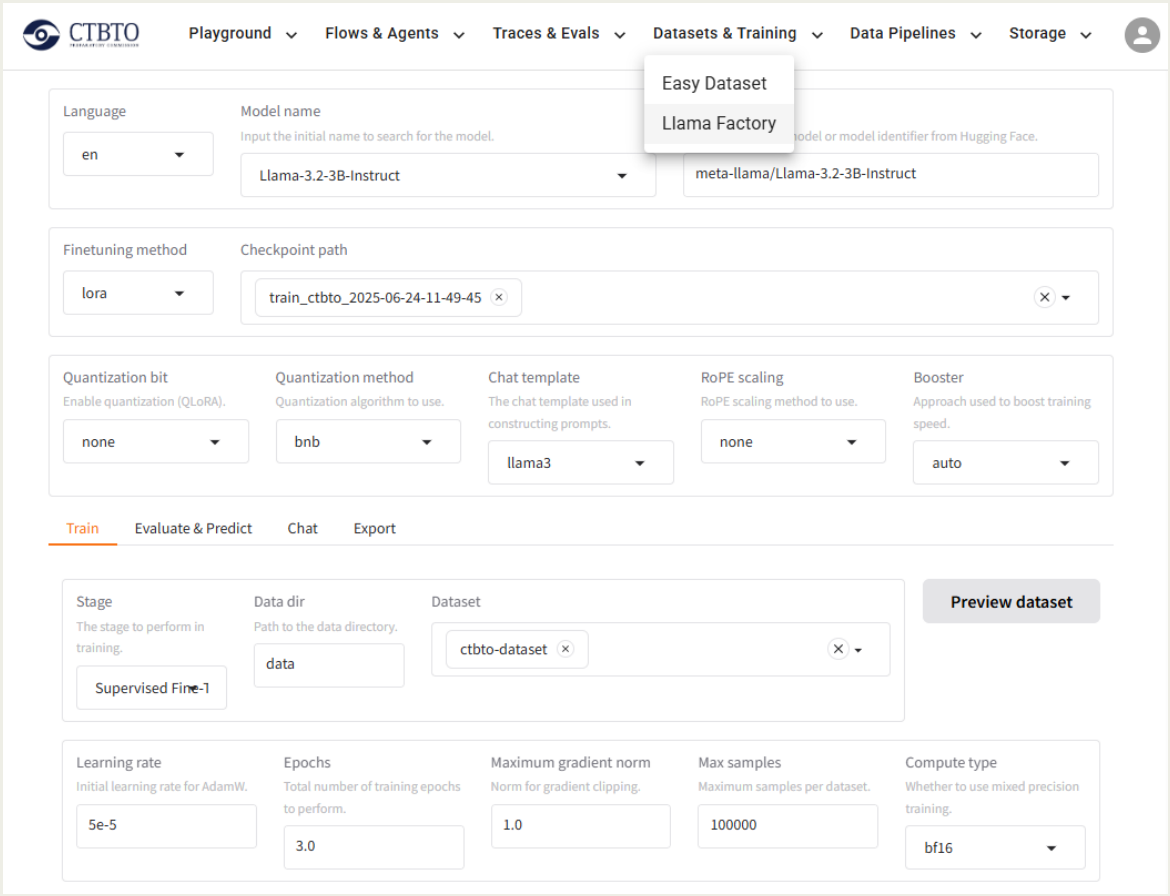
A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Use Case: A platform for developing LLM applications

- A **platform** for building, evaluating, training, monitoring and configuring LLM assistants and agents
- Flowise/LangChain (orchestration), Langfuse (evals), Easy Dataset & Label Studio (datasets/data labeling), Llama Factory (fine-tuning)
- Capabilities of the platform:
  - Helps with **dataset creation**, data labeling and annotation from private CTBTO data
  - Streamlines the **fine-tuning**, deployment, and management of custom models and adapters
  - Manage **LLM API access** and enforce budgets, guardrails, logging and cost tracking
  - More information: **P3.5-569** (e-poster)
- **Who is it for?** Targeted to developers/admins



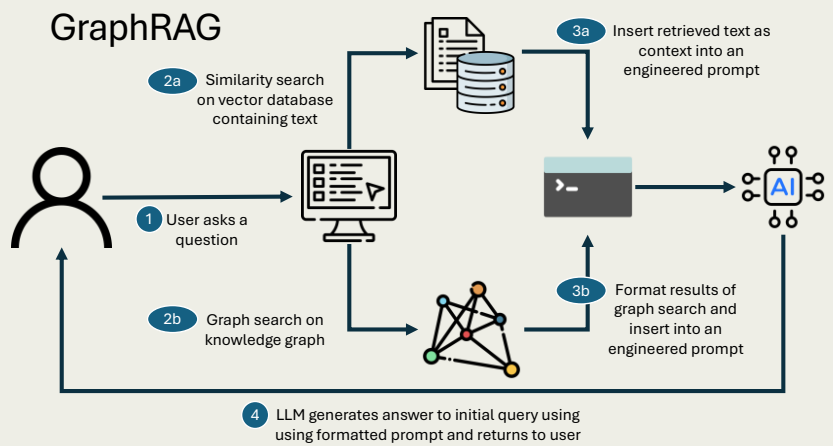
A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Use Case: Purpose-built AI Assistants

- A robust Graph-based **Retrieval Augmented Generation** (RAG) solution:
  - Use **local open-source LLMs** to extract entities/relationships as well as popular embedders to calculate embeddings from documents.
  - Based on LightRAG (which is a **hybrid GraphRAG** solution)
  - Runs on our dedicated GPU infrastructure.
  - Use Neo4j for storing the Knowledge Graph



- AI Assistants utilizing GraphRAG:
  - **OPS** and **Station Operator Assistants**: Knowledge from IRS (i.e. solved tickets), Build pipelines (pdf2markdown) and use semantic search to query the IRS Knowledge, Tools via function calling to retrieve/update IRS ticket information
  - **Other PTS Assistants:** WGB Assistant: Knowledge from ECS (i.e. WGB documents) imported to Vector DB, Use semantic search to query the collection, and many more: PMTool Assistant, SWP Assistant, MDA Assistant, etc

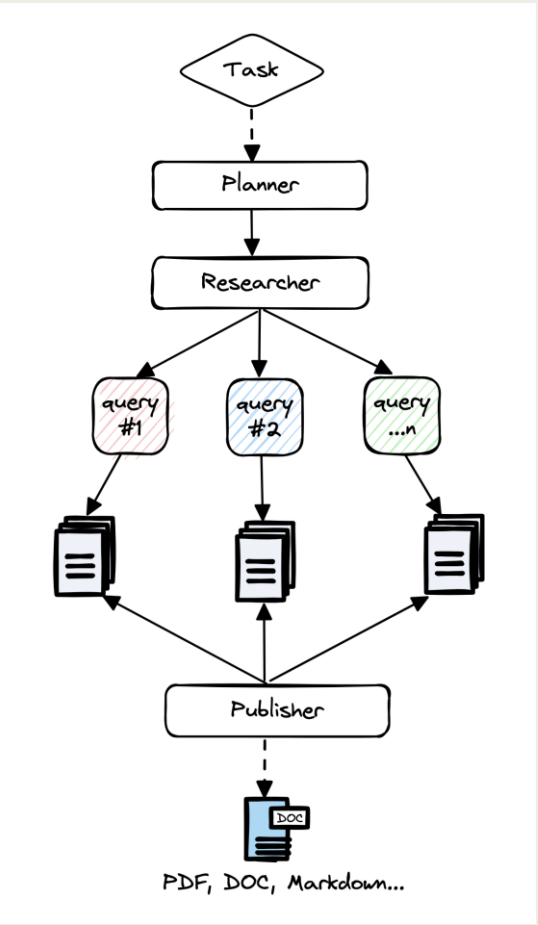
Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Use Case: Deep Research on CTBTO documents

- Deep Research solution (**Report Generation**):
  - Is based on GPT Researcher which is an **autonomous agent** (using LangGraph) designed to automate and enhance online (or local documents) research
  - It leverages large language models (LLMs) to perform comprehensive research on a wide range of topics, producing detailed, factual, and unbiased research reports with citations
  - It runs on our dedicated GPU infrastructure and uses open-source models as the FAST, SMART or STRATEGIC LLM (and open-source embedders)

- Capabilities of Deep Research:
  - Generate detailed research reports using web and/or **local CTBTO documents**
  - **Aggregate** over 20 sources for objective conclusions
  - Maintains memory and context throughout research
  - Export reports to PDF, Word, and other format



Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

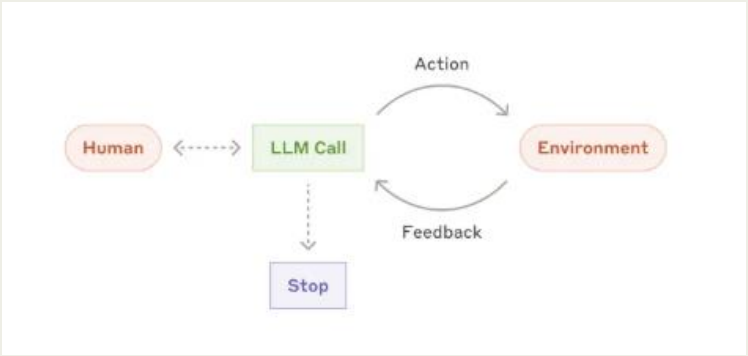
O4.3-565

Use Case: AI Coding Assistants & Agents

- An AI coding Assistant/Agent is a **tool** that uses a LLM that is finetuned for **coding tasks** and for "tool-use":
  - Both the tool and the model are specialized for **creating working software**
  - It can carry out software development tasks, such as fixing bugs, on a human developer's behalf

- **AI Coding Assistants** (higher predictability – lower agency):
  - Allow developers to work faster by automating repetitive or time-consuming tasks
  - Solve simple, pre-defined tasks but struggle with novel or open-ended problems
  - Uses an IDE where the **developer** is driving the coding

- **AI Coding Agents** (higher agency – lower predictability):
  - Agents are **autonomous** and typically implemented as an LLM performing actions (via tool-calling) based on environmental feedback in a loop.
  - Handle complex, multi-step problem-solving tasks, often across dynamic environments.



A framework for developing LLM applications

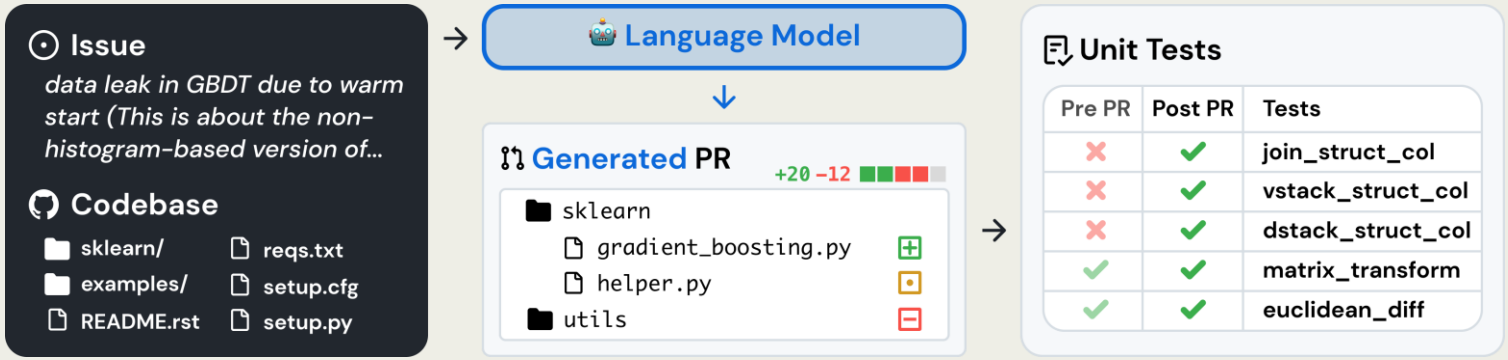
Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

What is SWE-bench?

- A **benchmark** used to evaluate the ability of large language models (LLMs) to solve **real-world** software engineering problems

  - Given a codebase and an issue, a LLM is tasked with generating a patch that **resolves** the problem.
  - Generating patches (PRs) for GitHub issues in popular open-source Python repositories.
  - It consists of a dataset of 2,294 problems and their fixes, allowing researchers to assess how well LLMs can understand and resolve code-related issues in a way that mirrors how human developers work.
  - Results are **verifiable**!



A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

How we use the AI coding tools (P4.3-576)

- **AI Coding Assistants: VS Code with Continue and Cline**
  - AI Plugin using local/hosted code LLMs (e.g. qwen3-coder:30b, codestral:22b) over entire IDC codebase
  - We use **vLLM** for inference which is suitable for multiuser environments
  - Builds a map of the entire git repo, works well in **larger codebases**, designed to **assist** users

- **AI Coding Agents: Codex CLI, Aider and OpenHands**
  - Use open-source agentic frameworks to automatically resolve issue and submit MR
  - **Automation** of software engineering tasks, assign simple issues to the Agent through Gitlab
  - Human still in control: has to manually **review and approve** the merge request

- **Specific Use Cases:**
  - *Explain code*: create automatic documentation of complex codebases
  - *Code completion*: Autocomplete code as soon as typing a few words
  - *Write new code*: Augment the software engineer to automate code writing
  - *Write comments*: Comprehensive documentation across the full codebase
  - *Write test cases*: Help automate software testing by writing new test cases

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Exploring future directions

- Exploring **Fine-tuning frameworks** in CTBTO (Easy Dataset & LLaMa Factory):
  - Knowledge Assistant: Fine-tune open-source LLMs on specific CTBTO domain knowledge/dataset
  - text-to-SQL Assistant: Fine-tune open-source LLMs on IDC database schema
  - SHI/RN Copilots: Fine-tuning a **Vision Transformer** on CTBTO labeled data:
    - Develop **datasets**, data preparation steps, and algorithms → build a new finetuned model
    - Using LoRA to **adapt** pre-trained open-source LLMs to PTS **specific** tasks like SHI or RN

- Explore **reasoning** models that are good for coding and create on-prem **SWE-agents:**
  - Running on our private cloud (sandboxed) → spins-up new docker containers (based upon a base image)
  - The agent goes to work in its own environment and after a few minutes gives you back a working PR

A framework for developing LLM applications

Evangelos Dellis, Cahya Wirawan, Janero Del Rosario

O4.3-565

Questions?