What is ChatDev?

What is ChatDev?

ChatDev is an open source agentic framework that implements multiagent collaboration through an organized team of specialized intelligent agents that are powered by large language models (LLMs). Each AI agent is orchestrated to collaborate on tasks within the core phases of the software development lifecycle to autonomously generate and produce a software application.

ChatDev simulates a virtual software company that operates through various intelligent agents holding different roles within the organization's environment. These agents form a multiagent organizational structure where they can collaborate by participating in functional seminars that aid in streamlining the software development process. ChatDev applies AI to the widely adopted "waterfall" model, a software development lifecycle model that breaks down the development process into phases: designing, coding, testing and documenting. Specialized agents work collaboratively to complete each phase.

ChatDev's primary objectives are to offer a customizable and scalable agentic LLM orchestration framework and to serve as an ideal scenario for studying collective intelligence. Collective intelligence is effective collaboration, the phenomenon where a group can outperform an individual in a complex task. The widely accepted approach to collective intelligence defines individuals as entities that can act with purpose and reason and adapt to an environment.[1] In agentic frameworks like ChatDev, these intelligent entities are known as agents.

Collective intelligence in multiagent systems (MAS)

Collective intelligence applications in AI and machine learning include collective intelligence systems or frameworks called multiagent systems (MAS). A MAS can be described as an interconnected network of problem-solving entities, or agents, that collaborate to solve problems that exceed the capabilities or knowledge of a single agent.[2]

AI tools like multiagent systems are an approach to collective intelligence (COIN) design. COIN is an extensive multiagent system that is characterized by minimal to no centralized communication or control. It features a world utility function that evaluates the potential histories of the entire system.[3] The field of MAS is concerned with the interactions among agents within the system as well as the makeup of each agent themselves. Although there are several ways to design a MAS, the core goal remains the same: to achieve a global task through the actions of its components.

Your subscription will be delivered in English. You will find an unsubscribe link in every newsletter. You can manage your subscriptions or unsubscribe here. Refer to our IBM Privacy Statement for more information.

Cooperative multiagent interactions

One of the most difficult steps in the design of a functional MAS is the coordination of the agents in a way that helps ensure that they cooperate on the global task.[4] Effective collaboration requires both individual task-solving abilities and well-coordinated interactions among participating individuals.[5] Planning a MAS environment involves the following considerations:

1. The constraints that the other agents' activities place on an agent's choice of actions
2. The constraints that an agent's commitments to others place on an agent's choice of actions
3. The unpredictable evolution of the world that is caused by other, unmodeled agents[6]

One way to approach this design challenge is to model teamwork explicitly through multiagent collaboration.

The multi-collaborative design pattern breaks down a complex prompt and delegates the abstracted tasks to agents who execute them based on their specialized roles.[7] For example, ChatDev uses a dual-agent communication design that is called ChatChain to facilitate cooperative communication.

How does ChatDev work?

Frameworks like ChatDev expand the functionality of LLMs beyond their exceptional natural language processing capabilities for text generation. ChatDev positions LLMs as versatile agents through its orchestration techniques that involve prompting and evaluating multiple collaborative agents with various social roles (for example software engineer, chief technology officer, chief executive officer, designer, tester).

Each agent uses their specialized role to work collaboratively on the core phases of the software development lifecycle. This process involves extensive communication among agents in their specialized roles to comprehend and analyze requirements through natural language, and development and debugging using programming languages. Phases in the ChatChain have varying levels of engagement in both natural and programming languages depending on the agents' role and prompt.

How ChatDev prompts its autonomous AI Agents

ChatDev's agents use role-playing and inception prompting as guides to complete their tasks. Inception prompting is a conversational LLM auto-prompting method that enables agents to prompt each other to solve tasks through role-playing.[8]

ChatDev applies prompt engineering at the start of each subtask phase. Although ChatChain does well at defining the task-solving process, agents simply exchanging responses without any additional prompting or guardrails is not effective for multi-round, task-oriented communication. To advance the progression of productive communications and avoid challenges like role flipping, instruction repeating and fake replies, ChatDev employs an inception prompting mechanism.

The inception prompting mechanism works by hypnotizing the LLMs with the instructor and the assistant system prompt as they are instantiated. LLM hypnosis can have a negative connotation (it can be used maliciously to manipulate a model) but in this case, it is a way to effectively prompt role-playing agents. In the prompt, tasks and roles are specified to help ensure that the LLM is instructing agents on how to fulfill their collaborative role and follow instructions.

These initial prompts for each role guide agents to give appropriate responses without role-flipping. The prompts for both roles in the system are nearly identical, addressing the overview and goals of the current subtask, the specialized roles, available external tools, communication protocols, termination conditions and constraints or requirements to prevent undesirable behaviors.[9] This mechanism functions like an LLM guardrail, enhancing the quality of responses and reducing the need for human intervention.

How AI agents interact in ChatDev

To guide proper cooperative communication between agents, ChatDev introduces an agentic workflow that is called ChatChain to further break down each phase into more manageable subtasks. This process guides multi-turn communications between different roles to propose and validate solutions for each subtask. Agents are governed with a mechanism called a communicative dehallucination, a communicative pattern to alleviate unexpected hallucinations. With this mechanism, agents request more detailed information before responding directly and then continuing the next phase of communication based on these details.[10]

ChatDev provides a browser-based visualizer to study the interactions of each individual agent acting within its role and environment. This interface provides users with the ability to investigate the multiagent system design itself. Users can view the artificial cooperation structure that allows the agents to work together to complete a global task beyond their individual capabilities.

ChatChain

ChatChain follows a dual-agent communication design that simplifies the task-solving process along the agentic workflow. The agentic design pattern begins by segmenting the software development process into three sequential phases: design, coding and testing. The coding and testing phases are further broken up into subtasks: code writing and completion, and code review (static testing) and system testing (dynamic testing).

Each communication phase within a ChatChain is made of one instructor agent and one assistant agent. In each subtask, two agents with distinct specialized roles (such as a reviewer and a GUI design programmer) fulfill the functions of an instructor and an assistant. The instructor agent instructs the discourse toward the completion of a task while the assistant agent receives and follows these instructions and responds with appropriate solutions.[11] The agents engage in multi-turn dialog until the task is complete or consensus is reached. Solutions extracted from the communication phase range from text (for example a description of the application's wanted functionality) to code (for example the initial source code).

This agentic workflow simplifies communications by avoiding complicated multiagent topologies and effectively streamlining the consensus process.[12] This approach provides a smooth transition between subtasks, as solutions from previous tasks serve as bridges to the next phase. This workflow continues until all subtasks are completed, guiding agents along the way. The workflow's chain-style structure links natural language and programming-language subtasks, effectively guiding agents on what to communicate. ChatChain is one solution to the most challenging problem of MAS design, agentic cooperation.

Using the default settings, the ChatChain produces software in the following order:

- **Demand analysis:** Decide the overall method procedure process for the application.
- **Language selection:** Decide what programming language to use to build and run the software.
- **Coding:** Agents write the code to build the application.
- **CodeCompleteAll**: Complete the code including missing functions/classes.
- **CodeReview:** Review and modify the code for functionality.
- **Test**: Run the software and modify the code based on the test report.
- **EnvironmentDoc:** Document the environment.
- **Manual:** Document and write a manual for the application.

Communicative Dehallucination

LLM hallucinations occur when models produce inaccurate or nonsensical output. Coding hallucinations are a type of hallucination that affects the outputs of models performing programming-related tasks. ChatDev's researchers have found that coding hallucinations frequently appear when the agent performing the assistant functionality struggles to follow precise instructions.[13] Assistants struggle to comply with instructions typically due to vagueness that requires multiple follow-up interactions and adjustments. To avoid undesirable outputs, ChatDev introduces communicative dehallucination.

Communicative dehallucination prompts the assistant to actively request more detailed suggestions from the instructor before providing a formal response.[14] Assistants take on an instructor-like role and proactively seek more information (for example the precise names of external dependencies, or which GitHub repository to commit changes to), taking on a deliberate "role reversal" before delivering a response. The assistant performs precise optimization after the instructor responds with modifications. The communication pattern instructs agents on how to communicate, enabling fine-grained information exchange while also reducing coding hallucinations.

Visualizer

ChatDev's Visualizer is a Python app that runs a local web demo where users can view real-time logs, replayed logs and ChatChain. The app contains three separate log viewers for each.

- **Log Visualizer:** A real-time log that prints the agent's dialog information, environment changes and other system information useful for debugging.
- **Replay Visualizer:** The replay log prints the dialog information of the agent.
- **ChatChain Visualizer:** Users can upload any ChatChain configuration file and view the agentic workflow chain for any current or previous software projects. This feature offers a transparent view of the entire software development process for users to examine each subtask solution and identify any possible inefficiencies.

The Visualizer also contains a Chat Replay page that shows the dialogs in natural languages between agents for a given generated software application.

ChatDev's LLM Integration

The concept of ChatDev's agentic design pattern is like the mixture of experts machine learning technique except its AI agents that are specialized experts handling complex tasks. Orchestration systems are needed to coordinate collaboration among a mixture of specialized agents.

The foundation of such orchestrations is pre-trained transformer models like OpenAI's GPT models.[15] Foundation models like IBM's granite series can also be used to ground agents with robust language capabilities and grant them specialized functionality. For instance, Granite Code is a family of models ranging from 3B and 34B parameter sizes and is trained on 116 programming languages. This offering is beneficial for agentic frameworks planning on using communicative agents for software development. At the time of writing, ChatDev supports OpenAI's GPT-3.5-turbo and GPT-4 models to power their intelligent agents.

How ChatDev scales

ChatDev is implementing a way to scale LLM-based multiagent collaboration with multiagent collaboration networks (MacNet). MacNet is inspired by the neural scaling law, which suggests that increasing neurons leads to emergent abilities.[16] ChatDev proposes a similar principle that applies to increasing agents in multiagent collaboration.

MacNet uses acyclic graphs to structure agents and enhance their interactive reasoning through topological ordering. Solutions are derived from the agents' interactions. This process consistently outperforms baseline models, facilitating efficient collaboration among agents across different network topologies and accommodating cooperation between more than a thousand agents. Through this application, ChatDev identified a collaborative scaling law that shows that the quality of solutions improves in a logistic growth pattern as the number of agents increases.

Multiagent systems for software development

Rather than trying to encompass all capabilities within a single model, multiagent systems divvy up tasks among several specialized agents. Agents can use the same language model or different ones, providing them the potential to accomplish a wide range of tasks.

ChatDev did an evaluation in a popular research paper that compares performances of itself and two other LLM-based software development orchestration frameworks, GPT-Engineer and MetaGPT. The metrics were based on completeness, executability, consistency and quality. ChatDev and MetaGPT outperformed GPT-Engineer, a single-agent orchestration approach, demonstrating that complex tasks are more challenging to solve in a single-step solution. These performance results suggest that breaking down complex tasks into more manageable subtasks enhances the effectiveness of task completion.

ChatDev outperformed MetaGPT significantly in the quality metric due to the agents' cooperative communication methods that use both natural and programming languages. Agents that are capable of efficient communication were able to guide each subtask to completion, overcoming the restrictions that are typically linked to manually established optimization rules.[17] This outcome suggests that multiagent frameworks that use communicative and collaborative agents offer more versatile and adaptable functions.

Other multiagent frameworks

GPT-Engineer – A single agent orchestration framework for building software applications. The user prompts the AI to build an application with the wanted software functionality and can communicate back and forth with the system to make improvements or updates.[18] Users are able to benchmark their own agent implementations against popular datasets through ‘bench,’ a binary installed in the system that contains a simple benchmarking interface.

MetaGPT – A multiagent framework that operates as a collaborative software entity that assigns different role-playing agents to GPT models to complete complex tasks. Like ChatDev, agents maintain various roles within a virtual environment akin to a software company. MetaGPT also uses carefully orchestrated SOPs (Standard Operating Procedures) to generate software.

crewAI – crewAI is an open-source multiagent framework that operates on LangChain. It organizes autonomous AI agents into teams that handle workflows and tasks that are related to LLM applications. crewAI integrates with any LLM.

AutoGen – Microsoft’s open-source multiagent conversation framework provides an elevated abstraction of foundation models. AutoGen is an agent-based framework that employs multiple agents to interact and address tasks. Its key features include customizable AI agents that participate in multiagent dialogs with adaptable patterns, enabling the creation of diverse LLM applications.