What is ReWOO?

ReWOO (short for “reasoning without observation”) is a reasoning framework that makes large language models (LLMs) more cost-effective and accurate in some complex reasoning applications. Models with ReWOO engage in a reasoning process around a problem before trying to solve it, leading to much greater efficiency, accuracy and robustness under tool failure.

The first LLMs (like OpenAI’s GPT-1 and GPT-2 models) provided answers directly; the subsequent wave of chain-of-thought models that debuted in 2022 added an element of externalized reasoning, with models essentially “thinking out loud” as they arrived at a response, improving accuracy and explainability.

Next came a generation of augmented language models (“ALM systems”) and AI agents, which added tool-calling capabilities on top of this reasoning. The early ALM frameworks—like ReAct—employ a thought-action-observation pattern, where the system will observe what it generates before beginning to think again. While generally effective, frameworks like ReAct can require heavy token consumption, since each subsequent tool call must include all the conversation history that precedes it —a cost that compounds with each step.

ReWOO breaks away from the think-act-observe pattern by decoupling reasoning from external observations, allowing the model to plan its chain of reasoning internally before selectively invoking tools or retrieving information. This separation reduces unnecessary back-and-forth and allows the model to maintain a plan throughout the task.

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.

How ReWOO works

ReWOO employs three distinct modules, which divide and conquer complex tasks. First, a Planner module maps out a blueprint for how the model will behave based on the user’s prompt. Second, a Worker module executes the plan, calling external tools (without repeating costly LLM API calls for "thinking," as in ReAct). Finally, a Solver module takes the plans and evidence, synthesizing the final response.

Though the difference in approach may seem minor, the results are dramatic: ReWOO performs as well (or slightly better) than ReAct against some benchmarks—all while using about 80% fewer tokens. (A token is a unit of semantic meaning for an AI model; the more tokens, the higher the cost of operation.) For instance, on the HotpotQA dataset (one battery of questions used to evaluate AI systems), ReWOO achieves 42.4% accuracy using 2,000 tokens, while ReAct achieves 40.8% accuracy using 10,000 tokens.

Crucially, this optimization of token efficiency makes reasoning models economically viable at scale.

ReAct vs. ReWOO: a real-world example

To illustrate the difference between these two common generative AI frameworks, let's examine a specific use case. Consider the different ways a ReAct vs. ReWOO system would address a user query asking for help packing for a trip that involves a flight between New York and Chicago tomorrow, followed by a drive to Milwaukee a day later.

A ReAct system would decompose the problem into a sequence of three thought-action-observations cycles before giving its final answer. In the first cycle, it would think, "I need to check tomorrow's weather in New York," using retrieval-augmented generation (RAG) to search for that weather (an action), finally observing the result. That result then serves as the input towards another three-step think-act-observe cycle for Chicago's weather. Third, it would do the same for Milwaukee's weather. Finally, it would collate its findings into an output (e.g. "Pack layers, because it gets colder in each location").

A ReWOO-style system, by contrast, would gain efficiencies by doing all the planning upfront. First, it would plan, "I need to get New York's weather tomorrow, Chicago's weather tomorrow, and Milwaukee's weather the day after." Next, it would work, by calling weather APIs in one tight sequence (or potentially in parallel), without doing any costly "thinking" at this workhorse step. Finally, it would solve, collating the evidence and outputting a final answer.

Benefits and downsides of ReWOO

In addition to token efficiency, ReWOO demonstrates an additional benefit: robustness under tool-failure. If a tool fails under ReAct, for instance, the system can get caught in an infinite loop (as the LLM repeatedly queries a broken database for the weather in Chicago, for example).

ReWOO is nimbler. Even if a tool fails to return a given piece of evidence, the initial overarching plan is still in place: The Worker module can progress, and the Solver module will be able to deliver at least a partial answer. In the weather example, instead of getting caught in an infinite or excessive loop querying a database for Chicago's weather, the Solver module would at least return an answer informing the user of New York's and Milwaukee's weather (assuming the Worker module was able to retrieve those bits of evidence), which might ultimately be sufficiently helpful for the user's planning needs.

Despite ReWOO's benefits, it is not a universally superior framework; it is simply better for certain kinds of jobs, particularly when the types and quantities of evidence needed are regular and predictable. Where ReWOO falls short, however, are with less predictable or structured problems that may require creativity, exploration or improvisation. With known unknowns, ReWOO excels, but with unknown unknowns, it flounders.

For instance, ReWOO would not be an optimal for debugging Python code, an exploratory and iterative process where each fix might yield new errors and clues, with the best-laid plans quickly becoming obviated. A more adaptable framework like ReAct, while less token-efficient in the abstract, would ultimately be a better match for such a problem.

How to implement ReWOO

As with most AI systems and frameworks, various approaches are available for the implementation of ReWOO workflows. An "official" implementation of the framework, which was first described by researcher Binfeng Xu (along with his colleagues, in 2023[1]), is available via Github. GenAI frameworks like LangGraph (which calls its modules "nodes") and the related LangChain are also popular. And a ReWOO-style multi-step reasoning methodology is also available using IBM's Granite.

One can get started with ReWOO at a conceptual level in any LLM environment with a well-crafted prompt that simply encourages the AI to craft a step-by-step plan to answer subsequent questions before progressing to any tool input.

The paper first describing ReWOO, for instance, includes example prompts, including one that begins: "For the following task, make plans that can solve the problem step by step. For each plan, indicate which external tool together with tool input to retrieve evidence." The study authors add, however, that "ReWOO is a general paradigm and prompts are not necessarily fixed. We encourage readers and users to adjust the prompts tailored to their own needs."[1]