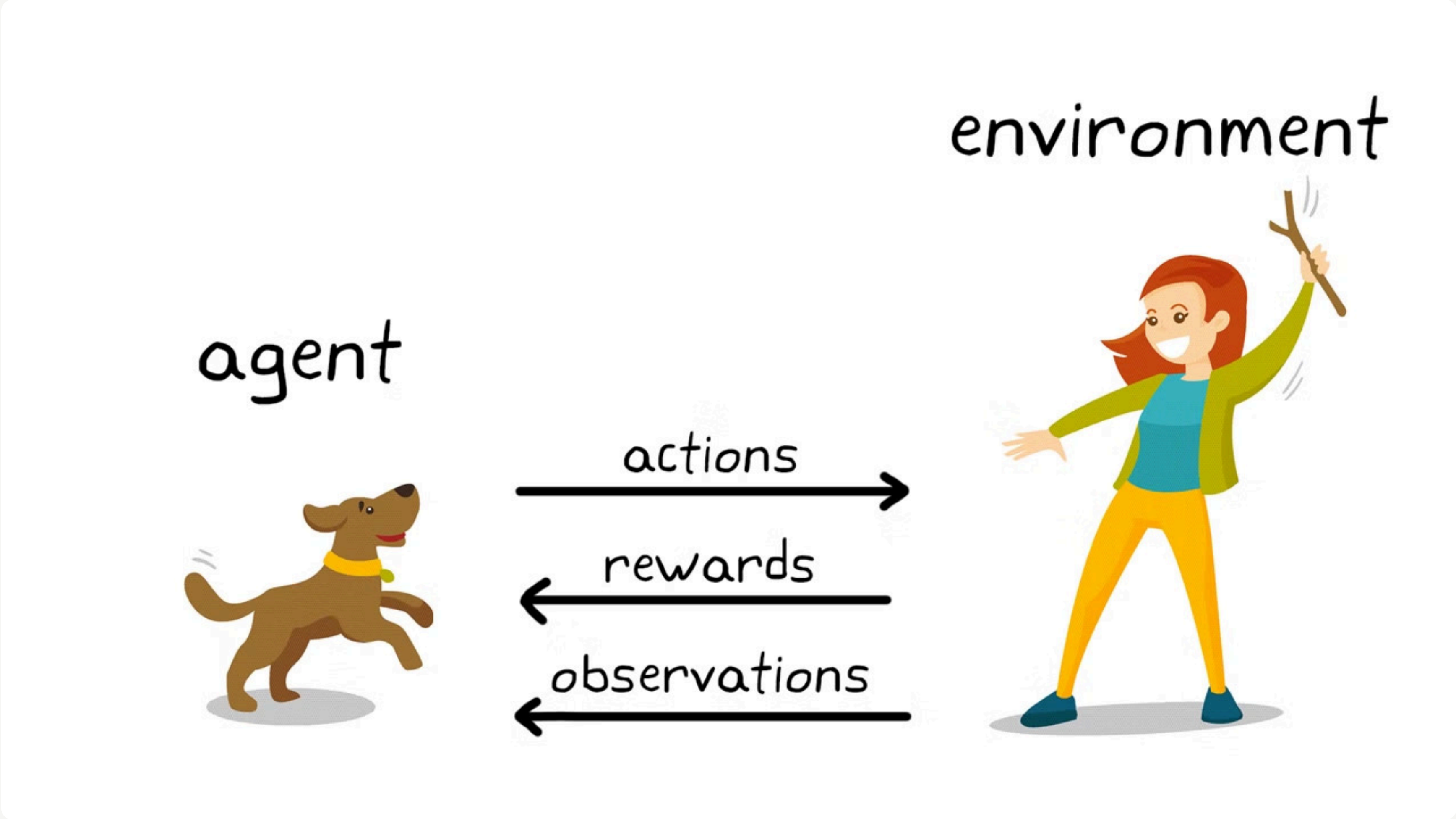
Updates on AI — 4 October, 2024

I only have four sections for today's Update. I'm taking a personal day tomorrow and a few sections I was going to write about tomorrow now have to wait. The four sections are, nonetheless, interesting or at least they were to me.

- New techniques and updates on using reinforcement learning; including "reward" schemes
- Training new models using "weaker but cheaper" (WC) models for synthetic data generation and sampling
- Divergent Chain of Thought (CoT)
- Agent-Oriented Planning in Multi-Agent Systems

Lastly for this week, I'll give an "AI Week in Review" with new things and thoughts that were announced that caught my eye.

by Bryan Sparks



Reinforcement Learning and Reward systems

The graphic above and this first introduction paragraph are from this article.

As a refresher, what is Reinforcement Learning?

> Using reinforcement learning terminology (figure above), the goal of learning in this case is to train the dog (agent) to complete a task within an environment, which includes the surroundings of the dog as well as the trainer. First, the trainer issues a command or cue, which the dog observes (observation). The dog then responds by taking an action. If the action is close to the desired behavior, the trainer will likely provide a reward, such as a food treat or a toy; otherwise, no reward will be provided. At the beginning of training, the dog will likely take more random actions like rolling over when the command given is "sit," as it is trying to associate specific observations with actions and rewards. This association, or mapping, between observations and actions is called policy. From the dog's perspective, the ideal case would be one in which it would respond correctly to every cue, so that it gets as many treats as possible. So, the whole meaning of reinforcement learning training is to "tune" the dog's policy so that it learns the desired behaviors that will maximize some reward. After training is complete, the dog should be able to observe the owner and take the appropriate action, for example, sitting when commanded to "sit" by using the internal policy it has developed. By this point, treats are welcome but, theoretically, shouldn't be necessary.

Several years ago, reinforcement learning (RL) was the "way" to train large models; of course there were many ways, and still are, but RL became very popular when demonstrated with what Google DeepMind did with AlphaGo and AlphaZero playing the complicated and ancient Go board game. I've written about RL and the famous "Move 37" in AlphaGo lore but leave any further reading to your searches. It's good and interesting if you have a moment to explore.

DeepMind continues to make heavy use of RL in their newer models that are simply extraordinary; like AlphaGeometry and it's follow-on AlphaProof, and with AlphaProteo, These are also worth some time to explore as they are amazing but are beyond the scope of this article.

It appeared to me that a few years back RL kind of took a "back seat" to other training/tuning methods but it is making a comeback, rightfully so, in my mind, with LLMs and Frontier Models with OpenAI's recent paper/article describing techniques in using RL in their recent "o1" model release. They use RL to help in the reasoning tasks, the CoT methods, of o1 to get better reasoning results. As one researcher at OpenAI suggested in a video interview of a group of researchers: the results were better than could have been reached by more traditional Human Feedback (HF) methods, even if the HF were at scale; meaning LOTS of "humans" giving quality feedback.

Going back a few years ago, the common, and perhaps sole, approach to giving models feedback was with RLHF (reinforcement learning with human feedback). More recent advancements, the past couple of years, have introduced Reinforcement Learning with AI Feedback (RLAIF) and other similar techniques. These RLAIF techniques have become so effective that I think using RLHF will be set aside.

So, what does that mean and how are RLAIF, or similar, techniques used? There are rich and interesting advancements being researched in this area. Perhaps this topic will deserve a series of sections in future updates but for now I will simply give a list of references that you can explore and determine their use or effectiveness in our genealogical scope. For what it's worth, I think this is an area worthy of serious consideration in many of the areas we do or intend to do.

To start let's see what others say in describing RLAIF when compared to RLHF: https://www.datacamp.com/blog/rlaif-reinforcement-learning-from-ai-feedback

This is an article from May of this year that uses open source feedback to reinforce: RLAIF-V: Aligning MLLMs through Open-Source AI Feedback for Super GPT-4V Trustworthiness... https://arxiv.org/pdf/2405.17220

An article from February: A Critical Evaluation of AI Feedback for Aligning Large Language Models.

This paper is a survey of other papers that explore Multi-Agent Reinforcement Learning (MARL). The paper is: Deep multiagent reinforcement learning: challenges and directions**.** Admittedly, this paper is from 2019 but MARL continues to be an area for research. For instance, this paper from May of this year: EFFICIENT MULTI-AGENT REINFORCEMENT LEARNING BY PLANNING. There is also this paper from 2023 that lists the then-current methods for MARL: Multi-Agent Reinforcement Learning: Methods, Applications, Visionary Prospects, and Challenges

I found this GitHub repository that tracks recent papers in the area of RLAIF (this would have been good to find earlier in my research 🤔). https://github.com/mengdi-li/awesome-RLAIF

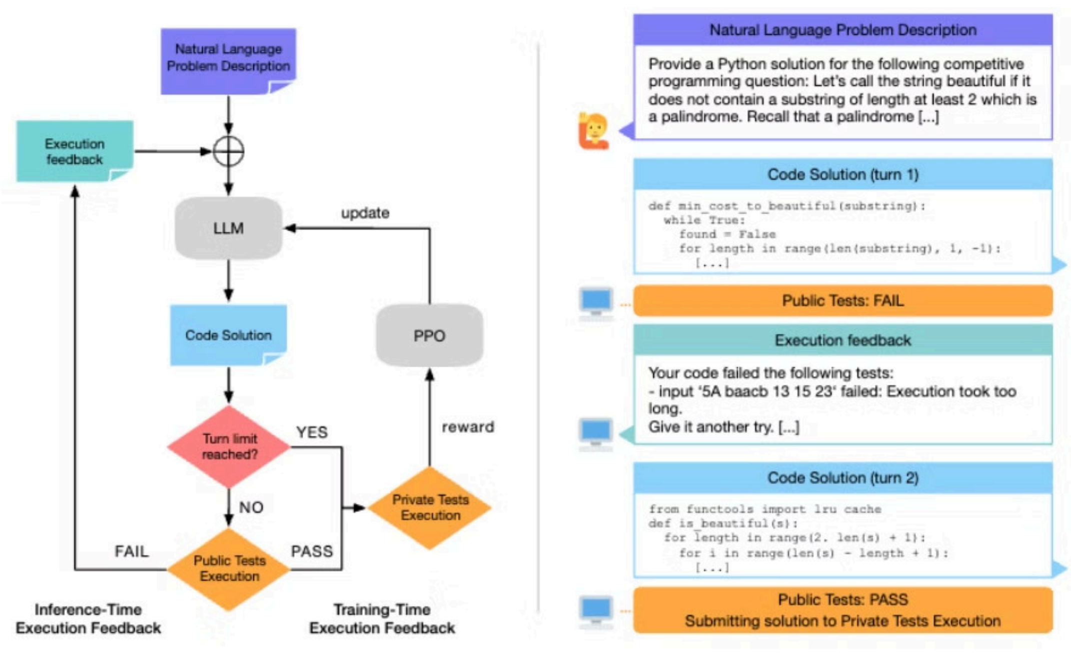
This reward paper is interesting though I recognize that the use-case presenting is in finding a pathway in a maze, like a warehouse. Nonetheless, the reward scheme Independent Q Learning (IQL) has broader application in MARL implementations, I feel. Cooperative Reward Shaping for Multi-Agent Pathfinding

This paper from July of this year from IEEE is heavy with math concepts but gives a survey of "reward engineering" of these various learning systems. The paper is good and the links to reference papers at the end is also a good list of additional research in this area. The paper is Comprehensive Overview of Reward Engineering and Shaping in Advancing Reinforcement Learning Applications

Lastly, this paper was published by Meta just last week entitled "RLEF: GROUNDING CODE LLMS IN EXECUTION FEEDBACK WITH REINFORCEMENT LEARNING." Though this paper is focused on code-generation it has principles that could be applied to other use cases. I'm particularly impressed, as the title hints, at the improvements at "execution time" or at inference. From the Abstract it says:

> Large language models (LLMs) deployed as agents solve user-specified tasks over multiple steps while keeping the required manual engagement to a minimum. Crucially, such LLMs need to ground their generations in any feedback obtained to reliably achieve desired outcomes. We propose an end-to-end reinforcement learning method for teaching models to leverage execution feedback in the realm of code synthesis, where state-of-the-art LLMs struggle to improve code iteratively compared to independent sampling. We benchmark on competitive programming tasks, where we achieve new start-of-the art results with both small (8B parameters) and large (70B) models while reducing the amount of samples required by an order of magnitude. Our analysis of inference-time behavior demonstrates that our method produces LLMs that effectively leverage automatic feedback over multiple steps.

A "flow" from this method is illustrated by the graphic below (from the paper):



Natural Language Problem Description

Provide a Python solution for the following competitive programming question: Let's call the string beautiful if it does not contain a substring of length at least 2 which is a palindrome. Recall that a palindrome [...]

Code Solution (turn 1)

```
def min_cost_to_beautiful(substring):
  while True:
    found = False
    for length in range(len(substring), 1, -1):
      [...]
```

Public Tests: FAIL

Execution feedback

Your code failed the following tests:
- input '5A baacb 13 15 23' failed: Execution took too long.
Give it another try. [...]

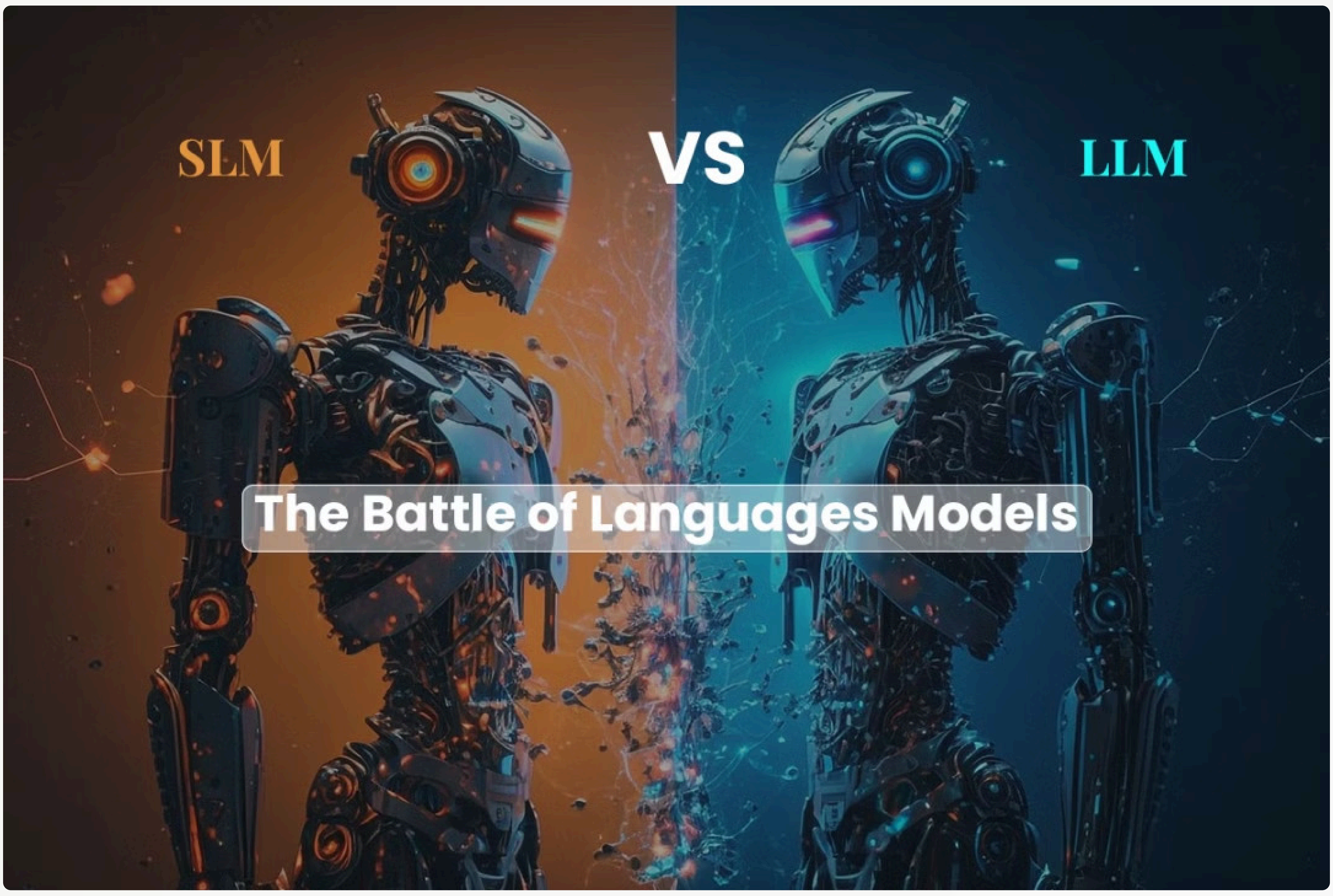
Code Solution (turn 2)

```
from functools import lru cache
def is_beautiful(s):
  for length in range(2. len(s) + 1):
    for i in range(len(s) - length + 1):
      [...]
```

Public Tests: PASS
Submitting solution to Private Tests Execution

Figure 2: **Left:** Overview of reinforcement learning with execution feedback (RLEF). The LLM is repeatedly prompted to implement code according to a problem description. Each attempt is evaluated on a public test set; upon failure, feedback is inserted into the conversation. If public tests are passing, or a specified turn limit is reached, execution on additional, private tests determines the reward signal. The model is then updated to optimize the reward with PPO. **Right:** Example dialog with two model responses. Execution feedback hints at an inefficient first solution, to which the model responds to utilizing a cache. The code passing the public test sets will be evaluated on the full test set.

That is probably enough to mention for now. Hope this area was interesting for you. I guess my takeaways are: a) RLAIF is our "friend" and likely the best way to get results going forward not only in better models but in crafting reasoning schemes on models, b) reward systems to "encourage" models to give "better" results is a rich area of interest for me, particularly with combined with an improvement-loop of agents acting in concert to get the "best" result in some scenario, c) using "agent" to help with getting better "feedback" to models or for some problem presented is a bit adjacent but also an exciting space, d) lastly, synthetic data to help in providing feedback or in training/tuning models has proven, to me, to be highly effective and is worth of some future section in a future Update. The next section of this Update is on this topic.



Smaller, Weaker, Yet Better: Training LLM Reasoners via Compute-Optimal Sampling

Out of Google DeepMind comes an approach of using "cheaper" and "smaller" models to help train other models and get better results. This seems like an area that should interest FamilySearch in that we could train models with this technique that solve complex tasks from our unique set of data sources to give us broader capabilities.

From the abstract from the paper is says:

> Training on high-quality synthetic data from strong language models (LMs) is a common strategy to improve the reasoning performance of LMs. In this work, we revisit whether this strategy is computeoptimal under a fixed inference budget (e.g., FLOPs). To do so, we investigate the trade-offs between generating synthetic data using a stronger but more expensive (SE) model versus a weaker but cheaper (WC) model. We evaluate the generated data across three key metrics: coverage, diversity, and false positive rate, and show that the data from WC models may have higher coverage and diversity, but also exhibit higher false positive rates. We then finetune LMs on data from SE and WC models in different settings: knowledge distillation, self-improvement, and a novel weak-to-strong improvement setup where a weaker LM teaches reasoning to a stronger LM. Our findings reveal that models finetuned on WC-generated data consistently outperform those trained on SE-generated data across multiple benchmarks and multiple choices of WC and SE models. These results challenge the prevailing practice of relying on SE models for synthetic data generation, suggesting that WC may be the compute-optimal approach for training advanced LM reasoners.

This paper is worthy of a browse but I'm not going to speak further on it here.

I'm intrigued by this general approach of using small language models (SLMs) that are small, easier and cheaper to fine-tune than LLMs and how these "small" models provide "feedback" to larger models that are solving more complex tasks. Also, this, as in this paper and this section, of using "weaker but cheaper" models to generate synthetic data to help with larger models. I think we, perhaps naturally, fall into the mindset that "bigger is better" in most things with generative AI but that isn't always proving to be accurate and certainly not cost effective.

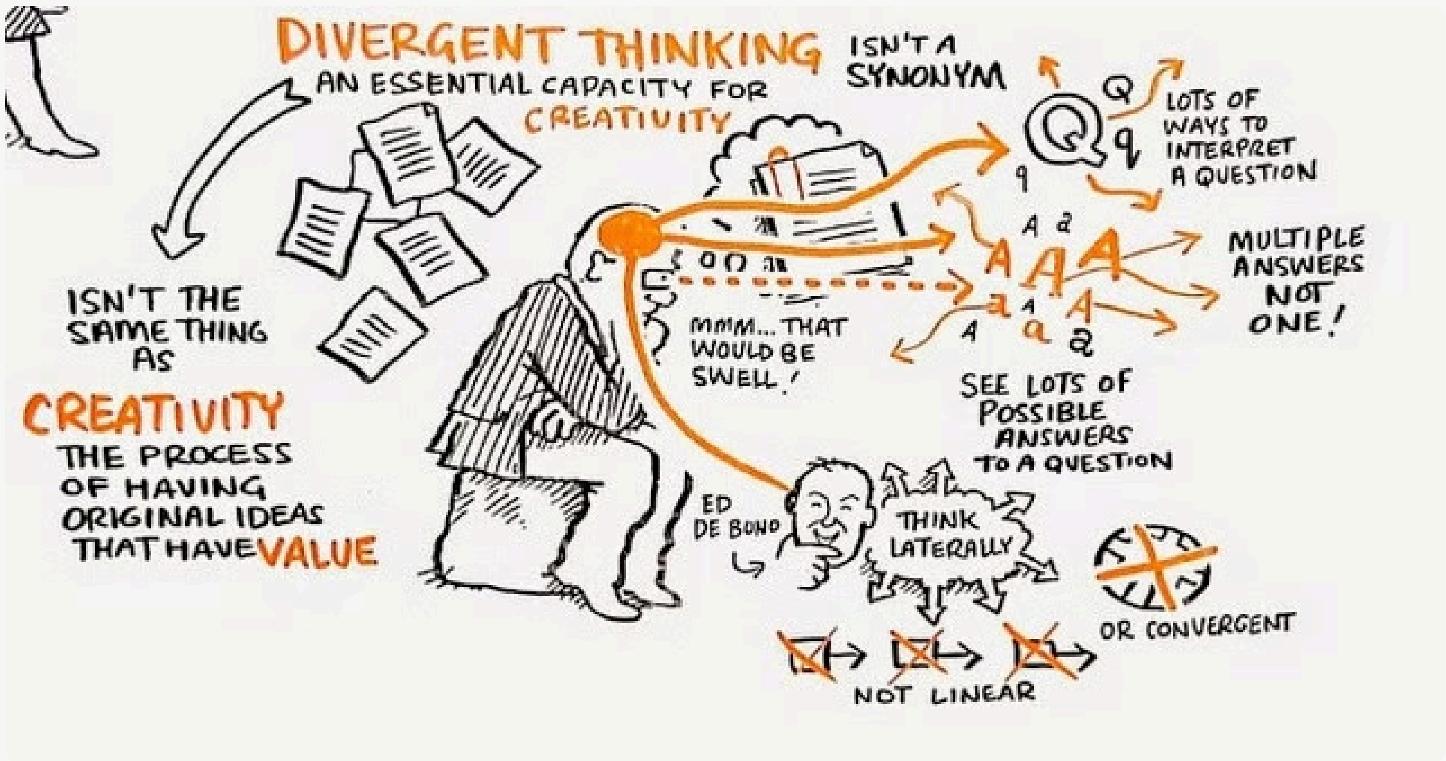
Though, I'm not saying that the large foundation models aren't capable with properly prompted in many ways. They, nonetheless, fall far short of the more "interesting" use cases and problems we likely will want to explore for better, quicker, and faster results; faster in terms of getting to market as opposed to inference time response.

It just feels like there are new "solution flows" or "architecture diagrams" of gross components and interactions within those components that might need to consider these new approach to "flow engineering" using an array of capable, fine-tuned, models perhaps in configurations of agentic systems.

I think "this" is all I'm going to say to this at this time. There are just strong indications that a new way, already here or fast approaching, that ISN'T just using your "foundation model" of choice in an API, even with structured output like so many of the foundation models are now offering.

This all seems akin to many years ago when software engineering largely moved to "object" models because the "objects" allowed for better abstractions and extensions to be made to those objects all towards more complex but much more capable large systems solving complex problems. I think we on the verge of a similar change where groups of agents armed with domain specific data and domain specific capabilities that are cheaper to run and fine-tune, "communicating" in groups, providing intricate and targeted reward systems to the larger tasks presented. Further, these systems offer, or could offer, scalable and valuable improvement loops for betterment — on their own with little human involvement.

Doing these Updates every week allows me, or more accurately drives me, to explore areas that seem impactful or could be impactful. Most of what I read and learn doesn't make it into these weekly Updates, however, this research illuminates, to me, trends that are very interesting. These trends are more interesting than the particular things that are written about. To this section and its concepts, this reinforces a belief that I've found on the rich value of small models and where a mixture of small and large models can do amazing things in tuning, reinforced learning, or in agentic configurations. This seems like an area where the organization might want to explore the current "state of the art" and best practices or likely best practices, on new implementation architectures, designs, flows, work descriptions, and the like.


DIVERGENT THINKING
ISN'T A SYNONYM
AN ESSENTIAL CAPACITY FOR CREATIVITY
LOTS OF WAYS TO INTERPRET A QUESTION
MULTIPLE ANSWERS NOT ONE!
ISN'T THE SAME THING AS
CREATIVITY
THE PROCESS OF HAVING ORIGINAL IDEAS THAT HAVE VALUE
MMM... THAT WOULD BE SWELL!
SEE LOTS OF POSSIBLE ANSWERS TO A QUESTION
ED DE BONO
THINK LATERALLY
OR CONVERGENT
NOT LINEAR

Divergent Chain of Thought

CoT has become very popular as a reasoning step with models. I have mentioned in previous Update the many variants of these methods that have shown further improvement. Some of those are Tree of Thought (ToT) and Graph of Thought (GoT). There are many others. As a brief refresher let's review what CoT is.

CoT prompting is an advanced technique for improving the reasoning abilities of LLMs. Here are the key points about CoT reasoning:

1. Definition: CoT prompting encourages LLMs to break down complex problems into a series of intermediate reasoning steps, mimicking human-like thought processes
2. Purpose: It aims to enhance LLMs' performance on tasks requiring complex reasoning, such as arithmetic, commonsense, and symbolic reasoning problems
3. Methodology: CoT prompting typically involves providing the LLM with examples that demonstrate step-by-step reasoning, encouraging it to apply similar logical processes to new problems
4. Effectiveness: CoT has been shown to significantly improve results on various reasoning tasks, particularly for models with around 100 billion parameters or more.

This paper published just this past July is entitled: Fine-Tuning with Divergent Chains of Thought Boosts Reasoning Through Self-Correction in Language Models. From the Abstract of the paper it says:

> Requiring a Large Language Model to generate intermediary reasoning steps has been shown to be an effective way of boosting performance. In fact, it has been found that instruction tuning on these intermediary reasoning steps improves model performance. In this work, we present a novel method of further improving performance by requiring models to compare multiple reasoning chains before generating a solution in a single inference step. We call this method Divergent CoT (DCoT). We find that instruction tuning on DCoT datasets boosts the performance of even smaller, and therefore more accessible, LLMs. Through a rigorous set of experiments spanning a wide range of tasks that require various reasoning types, we show that fine-tuning on DCoT consistently improves performance over the CoT baseline across model families and scales (1.3B to 70B). Through a combination of empirical and manual evaluation, we additionally show that these performance gains stem from models generating multiple divergent reasoning chains in a single inference step, indicative of the enabling of self-correction in language models.

The last sentence is really interesting, "enabling of self-correction in language models."

Though there has been many improvements to CoT ideas implemented for LLMs, it appears to me that there is much to be gained by these language models in encouraging them to "think" or "reason" on their answers before responding. With each improvement gained we learn, as an industry, that these foundation models are more capable that we originally thought—if properly "encouraged."

Though I mentioned this paper (published 21 Sept 2024) and its ideas in a previous Update, I think, it is worth another read on CoT solving problems. This paper aims to study why the form of CoT improves the reasoning capability of LLMs.

AGENT-ORIENTED PLANNING IN MULTI-AGENT SYSTEMS

I have written a lot on agentic systems, frameworks, and their variants. I've been impressed by the idea of an agent mesh backed by large language models and small language models. I'm enamored by the idea of SLMs that are trained on domain specific data adding to the collective. I'm also impressed by the idea of agents, in the mesh, that have specific "skills" or "capabilities" and how this idea seems endless in scope.

Of course, each agent can be further equipped with CoT reasoning, or similar, and have sub-agents that help in their specific area. These agents and sub-agents can be improved in time. I fully believe that these improvements don't necessarily need to be human driven but directed by other agents tasked with "continued improvement" tasks; see the Nvidia DrEureka paper previous mentioned in prior Updates and the "improvement loop" described within it.

As a continuation of this general idea is that this paper proposes an improvement in planning and improvement on outputs from agent networks. From the Abstract it says:

> Through the collaboration of multiple agents possessing diverse expertise and tools, multi-agent systems achieve impressive progress in solving real-world problems. Given the user queries, the meta-agents, serving as the brain within these systems, are required to decompose the queries into multiple sub-tasks that can be allocated to suitable agents capable of solving them, so-called agent-oriented planning. In this study, we identify three critical design principles of agent-oriented planning, including solvability, completeness, and non-redundancy, to ensure that each sub-task is effectively resolved, leading to satisfactory responses to the original queries. These principles further inspire us to propose a novel framework for agent-oriented planning in multi-agent systems, leveraging a fast task decomposition and allocation process followed by an effective and efficient evaluation via a reward model. During the planning process, the meta-agent is also responsible for evaluating the performance of the expert agents, making timely adjustments to the sub-tasks and scheduling as necessary. Besides, we integrate a feedback loop into the proposed framework to further enhance the effectiveness and robustness of such a problem-solving process. Extensive experiments demonstrate the advancement of the proposed framework in solving real-world problems compared to both single-agent systems and existing planning strategies for multi-agent systems.

Whereas prior task flows might be illustrated by this flow (graphic from the paper):

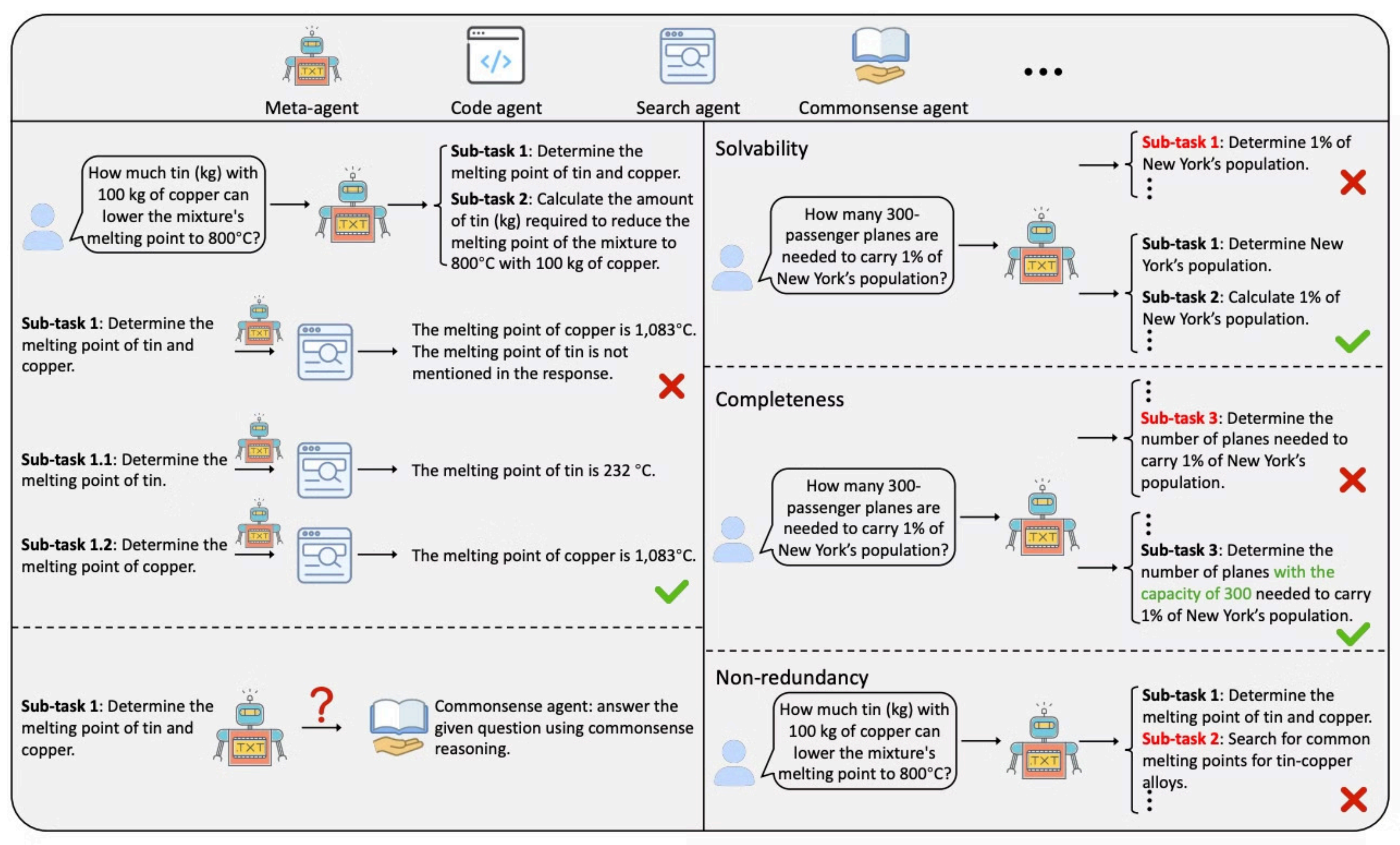

Meta-agent
Code agent
Search agent
Commonsense agent
...
How much tin (kg) with 100 kg of copper can lower the mixture's melting point to 800°C?
Sub-task 1: Determine the melting point of tin and copper.
Sub-task 2: Calculate the amount of tin (kg) required to reduce the melting point of the mixture to 800°C with 100 kg of copper.
Sub-task 1: Determine the melting point of tin and copper.
The melting point of copper is 1,083°C. The melting point of tin is not mentioned in the response.
Sub-task 1.1: Determine the melting point of tin.
The melting point of tin is 232 °C.
Sub-task 1.2: Determine the melting point of copper.
The melting point of copper is 1,083°C.
Sub-task 1: Determine the melting point of tin and copper.
?
Commonsense agent: answer the given question using commonsense reasoning.
Solvability
How many 300-passenger planes are needed to carry 1% of New York's population?
Sub-task 1: Determine 1% of New York's population.
Sub-task 1: Determine New York's population.
Sub-task 2: Calculate 1% of New York's population.
Completeness
How many 300-passenger planes are needed to carry 1% of New York's population?
Sub-task 3: Determine the number of planes needed to carry 1% of New York's population.
Sub-task 3: Determine the number of planes with the capacity of 300 needed to carry 1% of New York's population.
Non-redundancy
How much tin (kg) with 100 kg of copper can lower the mixture's melting point to 800°C?
Sub-task 1: Determine the melting point of tin and copper.
Sub-task 2: Search for common melting points for tin-copper alloys.

Figure 1: Examples of agent-oriented planning in multi-agent systems, regarding two challenges (left side) and three design principles (right side).

The paper describes the challenges of this straightforward approach as follows:

> Challenges The challenges of agent-oriented planning in multi-agent systems can be two-fold. Firstly, different from existing studies focused on task decomposition or chain-of-thought reasoning , agent-oriented planning requires intentional decomposition of user queries to effectively associate sub-tasks with agents, which includes considerations of the description of sub-tasks, the granularity of decomposition, the format of the responses, and so on. An example is illustrated at the higher left of Figure 1. Given a user query "How much tin (kg) with 100 kg of copper can lower the mixture's melting point to 800 °C?", a naive decomposition might suggest "determine the melting point of tin and copper." followed by "calculate the amount of tin (kg) required to reduce the melting point of the mixture to 800 °C with 100 kg of copper." However, when the sub-task of determining the melting points is assigned to a search agent, it may not result in satisfied responses since the query involves two entities simultaneously. In the context of agent-oriented planning, it is important to consider the capabilities of a search agent. As a result, such a sub-task should be further decomposed into individual entity searches: first determine the melting point of tin and then determine the melting point of copper. Secondly, assigning sub-tasks to appropriate agents is non-trivial, as the meta-agent can only rely on the agents' descriptions to determine task allocation in most cases (Shen et al., 2024). However, a concise and highly generalized description that adequately illustrates an agent's capabilities may not always be available, leading to suboptimal allocation. For example, if the description of a commonsense agent does not specify the extent of its knowledge base, the meta-agent may struggle to ascertain whether it is suitable to assign the task of querying the melting points of tin and copper to that agent, as shown at the lower left of Figure 1.

So what to do with these shortcomings?

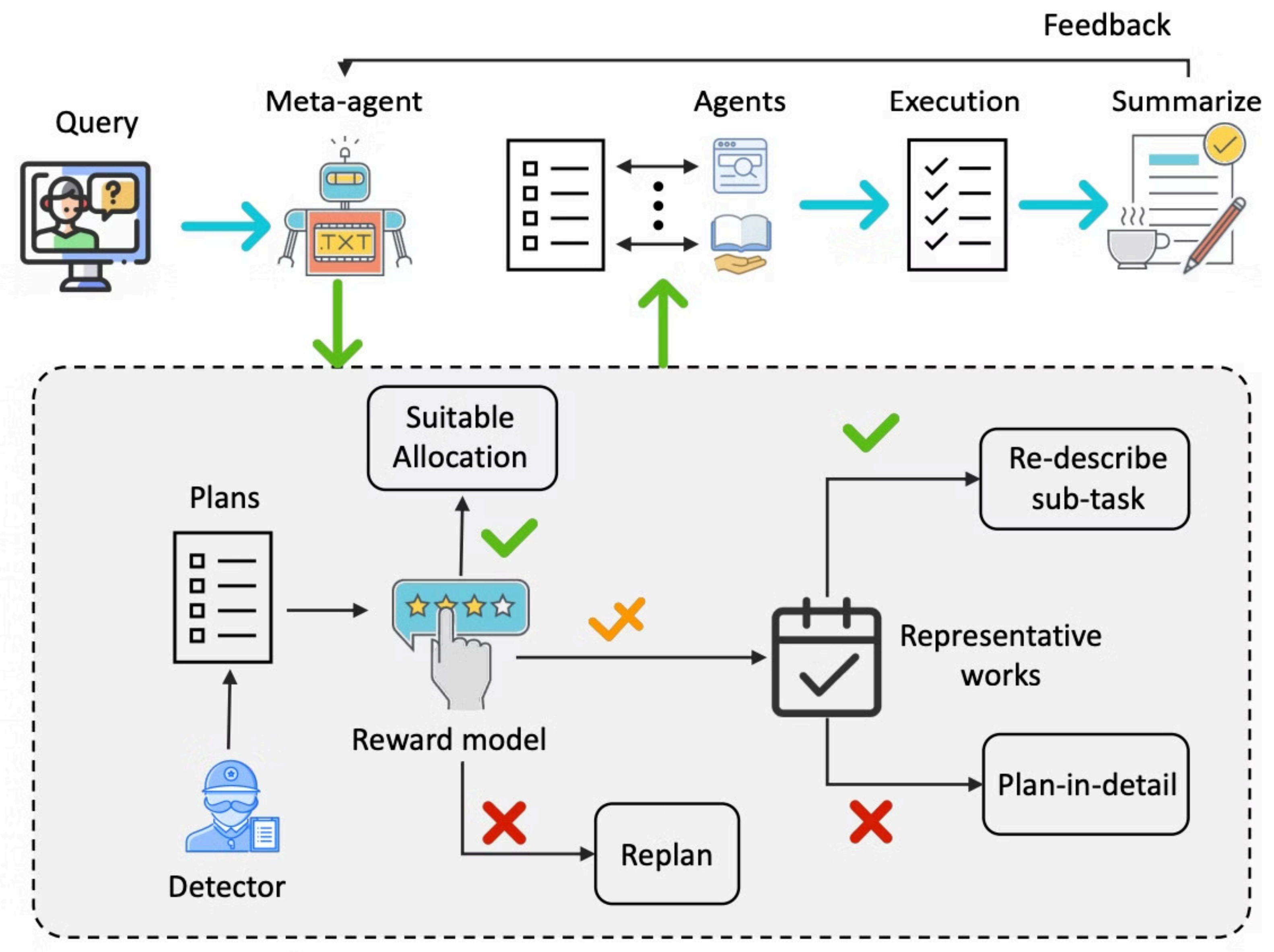


Figure 2: Overall architecture of the proposed agent-oriented planning framework.

This approach of agent-oriented planning is described as follows:

> First of all, following existing studies, we provide detailed instructions to the meta-agent to guide it in performing agent-oriented planning. Different from previous studies, the provided instructions incorporate the following requirements: (i) Integration of user query Q and all the agent descriptions A: We include both the user query and the descriptions of agents in the instructions, promoting the meta-agent to fully consider the capabilities of each agent and tailor the sub-tasks to align with these capabilities. (ii) Suggestions for assigned agents: We require the meta-agent to provide suggestions for the agents assigned to each decomposed subtask. Although we tend to separate task decomposition and assignment into two independent tasks performed sequentially, the experimental observations indicate that combining these two tasks enhances the effectiveness of agent-oriented planning. (iii) Structured decomposition of tasks: The decomposed tasks are required to be structured in a sequential manner, specifying any dependencies that may exist between sub-tasks, which ensures that the execution of tasks follows a correct logical order. The adopted prompt can be found in Appendix A.1. The aforementioned process of agent-oriented planning, which is called fast decomposition and allocation in this study, heavily relies on the capabilities of the meta-agent and meticulously designed system prompts. While such an end-to-end approach can achieve high efficiency, its success rate, namely fulfilling the three design principles mentioned above and providing reliable answers to the original user query, and its stability, namely the meta-agent can follow instructions and produce formatted responses, might not be satisfactory. Inspired by recent studies on scaling laws during inference (OpenAI, 2024b), we propose to design mechanisms to guide the meta-agent toward more comprehensive reasoning processes. The results of fast decomposition and allocation should be viewed as intermediate outputs rather than final results, which need to be further evaluated to offer detailed revision suggestions for the meta-agent.

All of the prompts used in this framework are included in the Appendix of the paper—they are fascinating, easy to read, and understand.

I wasn't going to include this section in this Update due to time but I couldn't stop thinking about it and felt that it needed to be mentioned even though it is already Monday morning and I'm late getting this out. I'm not saying this framework is the "final word" on improvements in agent and multi-agent planning but it just gets better and better. I love that this approach is relatively easy to implement and will give better and more accurate responses to queries presented. I also am impressed that these various improvements can be applied regardless of the actual agentic framework in use; e.g., LangGraph or some other.

AI news from the week:

The "big" news, to me, from this past week was OpenAI's announcement, during the OpenAI Dev Day, of their long-awaited voice interface and the real-time nature of its use. This, voice interaction, feels like a game-changer for using LLMs going forward. User interfaces "might" dramatically change going forward if this catches on—which I think it will.

The second big announcement from OpenAI's Dev Day was their "Canvas" feature as a response to Anthropic's Claude "Artifacts," I think. Canvas is certainly "cool" and this trend of interaction with queries on the "left" and responses on the "right" of the screen is really compelling, to me.

Word is that Anthropic is nearing another release and I'm excited to see where their Artifacts will be improved.

There is so much more ... but I'm out of time and need to get this out, finally. Perhaps the Update for this Friday will include more in this section and I'll cover some of last weeks announcements as well.

That's what I have. More to come next week.

Have a great weekend!