A long list of **open problems** and **concrete projects** in evals

For some background, see plans for evals field building.

Context for readers:

- The goal of this document is to have
  - A number of "ready to go" evals ideas that someone new to the field could do within a few days or weeks of effort.
  - A set of research projects that could easily result in one or more academic papers, e.g. as starting points for PhD students.
- You're free to just start with these projects and don't need to ask for permission. You can reach out to named individuals but they might not have the time to answer.
- You can comment in this document if you want others to reach out to you.

Context for Contributors:

- The comment functionality is on, so you can suggest new ideas.
- We'd love to get your contributions. If you want to get credited for your ideas, please add your name to the ideas that you contributed, e.g. "Credit: Marius"; if you don't want to be named, that's also fine.
- You are welcome and even encouraged to add things like "Here is an evals project I did in the past. I'm interested in the following 5 additional questions that I don't have time for". These typically make for great projects.
- If Marius doesn't think the contributions are high quality or they get out of hand, we might turn off the comment functionality.

Build more & better evals

First we present general strategies to create more and better evals and then suggest a list of specific eval ideas from various fields. We're interested in both *capability* and *propensity* evals.

General suggestions

Improve existing evals

Difficulty: easy-medium
Time estimate: 2 days - 1 month (depending on how serious you want to be about the improvements)
Credit: Marius

- Choose a publicly available eval or benchmark. Typically, you can find ways to improve it. This is a good way to get more familiar with the field and build intuitions about eval design.
- Here is a long list of evals that you can start with
    - See section “LM agents” and “Benchmarks” in “An Opinionated Evals Reading List”
    - Look at the benchmarks used in HELM (maybe start with HELM lite)
    - Look at the OpenLLM leaderboard
- Get a feel for the evaluation
    - Read through the samples in a lot of detail and note down all the things that are suboptimal about them, e.g. syntax errors, labeling errors, they don’t measure the right concept, etc.
    - Write a general specification of what good samples look like and what potential problems are. Make a clear rubric from 1 to 10 that includes at least one sample for each rating.
    - Let an LLM go through each sample and rate it a score from 1 to 10 based on your rubric.
    - Look at some evals with the best, worst, and random ratings and check if the rating aligns with your intuitions.
    - Iterate the above process until the ratings seem right to you.
- Improvements
    - Use an LLM and the specification you had earlier to create new samples as an improved version of the benchmark.
    - If you think your benchmark is better than the original, you can publish it and let the original authors know.
- For more details, you can see Dev et al., 2024
- Note: If you pick an agentic benchmark and you try hard to improve it, this can be a multi-month endeavor for a small team, see e.g. SWE-Bench-verified

Safety framework evals

Difficulty: hard
Time estimate: 1 week (MVP) - multiple months
Credit: Marius

- Many voluntary commitments and regulatory efforts specify the abstract capability they want to measure but do not specify a detailed evaluation. Filling in this gap is not trivial but very needed and a great way to get good at building frontier evals.
- You can look through any of the following publicly available safety frameworks
    - METR has an up-to-date list of all frontier AI safety frameworks
    - Model evaluation for extreme risks
- Read through the framework and understand which claims about capabilities (and propensities) they make and what they want to measure.

- See how other papers have implemented evaluations that are supposed to measure frontier safety capabilities, e.g.
    - Evaluating Frontier Models for Dangerous Capabilities
    - Sabotage Evaluations for Frontier Models
    - RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts
    - Frontier Models are Capable of In-context Scheming
- Pick one specific capability (or propensity) from the safety frameworks that sounds interesting to you and specify it in more detail.
    - Write a brief threat model, i.e. which concrete set of scenarios you're worried about and how capabilities relate to harm in that scenario.
    - Specify what exactly you want to measure.
    - Design the eval
    - Run the eval & iterate
- Think about how your evaluation would relate to (potential) red lines set by the framework.
- Marius comment: I think this is hard but potentially the fastest way to demonstrate competence in evals, e.g. in case you want to get hired to work on evals full time (Additionally, I expect propensity evaluations for LM agents to become a big thing soon).

Reliability evals

Difficulty: medium
Time estimate: 2 weeks (MVP) - multiple months
Credit: Sayash/Benedikt/Arvind (Princeton)

- Most evals focus on pass@1 (what an AI system *could* do) not on what AI could do reliably.
- Create benchmarks specifically focused on testing and improving reliability of deployed AI systems.
- Come up with scenarios where reliability is important (e.g., customer service chatbots) and create benchmarks that measure how reliably models or agents can behave in those scenarios
- Use metrics that are more amenable to measuring reliability (e.g., consistency — pass^k — instead of pass@k)
- A great example of such an implementation is Tau-Bench by Sierra AI.

Port more evals into Inspect

Take existing benchmarks and port them to Inspect. For a list of benchmarks that are already in Inspect, see here. By the time you read this, some of these might already be in Inspect.

List of Benchmarks that aren't yet in Inspect but maybe should be:

1. TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks
2. AgentBench: Evaluating LLMs as Agents

    a. There has been an attempted port here; please reach out to Teun for comments
3. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering
    a. There is a port here; please reach out to Teun for comments
4. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts
5. LINGOLY
    a. Authors are currently porting. If you're interested in helping, please reach out to Harry Mayne
6. MLGym

In general, just check which of the latest benchmarks are not yet in Inspect, ask whether someone else is working on them already and if not, port them. It's probably also fine to have multiple good implementations for the same benchmark, so redundancy is not a big problem here.

Specific suggestions

Scheming

General scheming

Difficulty: easy
Time estimate: 1 - 4 weeks
Credit: Marius

Replicate and investigate the findings from Apollo's paper "Frontier Models are Capable of In-context Scheming." Specifically, go to section 3.6 "Multiple models can sandbag" and try to understand under which circumstances the model is willing to sandbag and for what reasons. Broadly speaking, you should try to describe the "character" of the model and provide a potential list of hypotheses that explain the model's behavior across a large list of scenarios. We expect that some of this can be automated, e.g. you can automatically generate and classify scenarios to rapidly test the model. We have already done a brief shallow follow-up investigation in this post (and updated the paper). There is also one decent follow-up by Farrel Mahaztra.

Difficulty: easy
Time estimate: 1-2 weeks
Credit: paper by Ryan Greenblatt et al.; Marius wrote the suggestion.

Replicate and extend the findings in "Alignment faking in large language models." Concretely, find more situations where the model is willing to scheme in order to be helpful (or whatever goal it has internalized). It is possible that this search process can be largely automated with LLMs.

Difficulty: medium
Time estimate: 1 week - 1 month
Credit: Marius

Build more and better datasets that prominently feature scheming, e.g. similar to "Large Language Models can Strategically Deceive their Users when Put Under Pressure," "Uncovering Deceptive Tendencies in Language Models: A Simulated Company AI Assistant," "Frontier Models are Capable of In-context Scheming," or "Alignment faking in large language models."

In general, we encourage datasets and situations that are agentic and increasingly complex, e.g. harder than the ones presented in the in-context scheming paper. They can be both about propensity or capabilities, both are interesting to track. Finally, as a nice-to-have, consider making datasets that are suitable to train white-box probes, e.g. by allowing to easily create contrast pairs.

Difficulty: medium
Time estimate: 1 month
Credit: Marius

Build a highly realistic demo for AI scheming from instrumental convergence. All current examples feel too toy. The kind of thing I have in mind is roughly:

1. Give the agent a task that it would likely get in the real world, e.g. a 2-hour software task.
2. Design the task such that there is a benefit to acting in an undesired way. But this should be non-obvious to an average programmer. For example, the task might be easier if you have better permissions. There can be multiple such shortcuts in the task. All shortcuts should be undesirable to the average programmer, i.e. the programmer would not consent to the model if it asked.
3. Tell the model to solve the task and observe whether it will find and use any of these shortcuts and realize that hiding this action is a better strategy.
4. Start with a simple environment where the scheming action is fairly obvious and nudge the model to scheme. Over time, increase realism and complexity and reduce the nudging.
5. The resulting demo should be clearly for instrumental reasons, i.e. nowhere in the prompt do you tell the model to scheme (plausibly even the opposite). It should also be intuitively bad, i.e. if you show this to an average software engineer, they would immediately feel like this is a bad trend.

I'm unsure how hard it is to create this. For example, I could imagine that current models are too well aligned to scheme in an obvious way. I think a post like "I tried for 3 months and couldn't

find anything that feels realistic" would also be useful as evidence that current alignment methods work decently well.

Difficulty: medium-hard
Time estimate: 2 weeks - 6 months
Credit: Mikita wrote most of the paper; Marius wrote the suggestions in this doc

Go through the paper "Towards evaluations-based safety cases for AI scheming" and build evals for various sections. Examples include:

- Section 5.1.4 on sufficient elicitation
  - Build datasets for to evaluate for sandbagging
    - Teun: we are currently doing experiments with sandbagging capability on mle-bench, and also testing various monitoring strategies to catch it. We are looking for more people, so you can reach out if you want to help!
  - Test whether you can make a model "resist finetuning", e.g. similar to the sleeper agents paper but maybe in a more natural setting.
  - Build the tools to assess whether the assumption "Ensuring Smooth Increases in General Capability Between Evaluated Models" is given. Concretely, we would expect that even when the model isn't scheming, there is some noise and fluctuation across benchmarks. Thus, to detect scheming, we'd need to check whether we're significantly different from this natural fluctuation.
- Section 5.2 on Scheming inability
  - Build more evaluations that indicate that would provide evidence for or against the model's ability to scheme, e.g. as in "Frontier Models are Capable of In-context Scheming."
  - Build an evaluation that tests for *hidden or obfuscated* scheming reasoning, i.e. how well the model can scheme without revealing this in its CoT.
- Section 5.3 on Scheming harm inability
  - Build evaluations for Sabotage of AI systems, Sabotage of AI research, and Targeted high-stakes persuasion
- Section 6
  - Build scheming control evaluations (see section on control evals)
- Section 7.2
  - Build adversarial alignment evaluations, e.g. honeypots where you test whether a model is willing to defect for reasons that we would consider bad. Try to work on methods and techniques to assess whether the assumptions in 7.2.1 and 7.2.2 are given.

Evaluation gaming

Difficulty: medium
Time estimate: 1 week - 1 month per bullet point

Credit: UK AISI & Marius

- Can an autonomous agent sandbag (i.e. strategically underperform) in a capability evaluation when instructed to do so?
    - Can a human differentiate between a trajectory where the model failed for benign reasons (e.g. lack of capabilities) vs. sandbagging. In other words, how realistic do the sandbagging strategies of the model look?
    - Teun: We are working on this and could probably use more people. If people have some relevant research experience, they can reach out.
- Can a model detect when it is being evaluated? Design systematic ways to assess if the model understands that it is in an evaluation. Potential approaches:
    - Do structured interviews with the model after it has finished an evaluation vs. a control sample from a real deployment. Let the model describe which things it is looking at to distinguish between the two.
    - Tell the model that it should act differently (e.g. answer in English vs. Spanish) depending on whether it thinks it's in an eval or not and then record real eval vs control samples
    - Give the model an open-ended instruction to describe what it notices about its environment, then run an eval and record whether it explicitly notes that this is likely an evaluation (see Appendix L: long monologue tasks in the SAD paper).
    - Take an LLM, show it real evaluation vs. control samples, and let it classify them as real or not real.
- Collect samples of models realizing that they are being evaluated in the wild. Either run a lot of evals and investigate the CoT or talk to others who have found such cases and reproduce them. Then publish a long list with cases.

Steganography

- Replicate and extend OpenAI's steganography eval.
- Replicate and extend Hidden in Plain Text: Emergence & Mitigation of Steganographic Collusion in LLMs

Sabotage

- Replicate and/or extend Sabotage Evaluations for Frontier Models
- Replicate and/or extend A Toy Evaluation of Inference Code Tampering

Autonomy

Resource acquisition

Difficulty: medium
Time estimate: 1 week - 1 month per bulletpoint
Credit: UK AISI

- Can an autonomous AI agent reliably earn money by trading stocks over a period of N hours?
- Can an autonomous AI agent reliably earn money by solving tasks in a creative service marketplace, e.g. Fiverr?
    - See SWE-Lancer: Can Frontier LLMs Earn $1 Million from Real-World Freelance Software Engineering? For a high-quality implementation of this idea
- Can an autonomous AI agent reliably earn money by building a low-touch business, e.g. dropshipping?
    - See for example https://andonlabs.com/evals/vending-bench.

Coherence horizon

Difficulty: medium
Time estimate: 1-3 months
Credit: from Eli L, lmk if you work on this

Give an AI a really hard open-ended task and measure for how long it can coherently improve its performance / make progress; might have to involve some subjectivity. Could potentially produce good demos. Could start with the methodology used in MLE-Bench (see e.g. Figure 4).

AI R&D

"Fast inner loop" AI R&D

Difficulty: easy
Time estimate: 1 - 4 weeks
Credit: Marius & Jeremy

Create an experimental setting that has a "fast inner loop", i.e. where running an experiment is fairly clearly specified and has a few seconds to minutes turnaround time. This mainly tests how well LLMs are able to iterate on experimental evidence and come up with new ideas to test. It does exclude the more abstract research skills like which direction to pick to begin with. A while back, we collected a long list of potential settings but decided not to work on them because other things were more important. Nevertheless, we think these would be worth doing. Not all of them strongly adhere to the "fast inner loop" setting.

1. **Automatic jailbreaks**: how good are LM agents at red-teaming other models? This is similar to the work done by Haize Labs. The core difference to Haize's work is that we would provide the model with much less prior knowledge about good jailbreak techniques.
2. **Prompt engineering to maximize benchmark scores:** You give the model a benchmark and the task to improve the score as much as possible by iterating on the system prompt. However, the model is not allowed to put answers into the prompt. It can only provide general guidance.
3. **Hyperparameter optimization:** Create a synthetic task where we define optimal hyperparameters and define a cost function that allows us to calculate how good the

proposed hyperparameters are with respect to the reference hyperparameters. The agent now needs to find those optimal hyperparameters in as few steps as possible. The fast feedback is calculated with the cost function. Might also use a real reference implementation as long as we have some decently working hyperparameters to compare to. The model should use as little compute as possible since the solution is a huge grid search otherwise.

4. **Fast Triton Kernels:** We evaluate whether an LM agent can implement fast triton kernels for specific neural network operations. There are many reference implementations (e.g. from unsloth) which we can compare to. There likely is dataset contamination by reference solutions.
5. **Implement GPT-2 in C++ and make it fast:** The LM agent has to implement training and inference of LLMs in C/C++ (like Andrej Karpathy) and make the code really fast. There are different versions of this where we already provide a test script for the model or let the model do everything from scratch.
6. **OpenAI RL gym:** Implement RL algorithms that solve OpenAI Gym environments of various difficulties. For example, the LM agent could be tasked to improve a basic RL agent in the pendulum environment until it can compete with decent human solutions as a basic test. Then, we would move on to increasingly complex environments.
7. **Fine-tune another LLM:** The LM agent has to finetune another LM for relevant applications. For example, finetune LMs to be less sycophantic (the agent can use any method/dataset that it wants) and evaluate the LM for sycophancy on Anthropic's dataset.
8. **Replicate a paper:** The LM agent should replicate an ML paper. The two key requirements are a) there should be reference code (optimally not part of the training data) and b) the expected outcome of the paper should be clear such that we can judge if the re-implementation was successful. Paper replications capture some key components of research, e.g. sufficient research understanding and sufficient coding prowess. Other organizations have also used paper replications as a benchmark for AI R&D, e.g. METR re-implemented the reversal-curse paper).
9. **Research challenges:** The LM agent is tasked to solve real research challenges (Find the Trojan: Universal Backdoor Detection in Aligned Large Language models, CNN Interpretability competition, LLM Capture the Flag, Inverse scaling Prize, ARENA monthly algorithmic problems). Solving research challenges is a good proxy for research capabilities since the model would actively compete with existing research groups.
10. **PaperQA:** Automatically generate a Multiple Choice Question Answering dataset of hard, paper-specific research questions (akin to GPQA/MMLU). We select relevant papers (e.g. FlashAttention), provide the paper in context and then instruct Claude-3 to generate hard questions with answers and references.

Human uplift trials for AI R&D

Difficulty: hard
Time estimate: 3-9 months
Credit: Eli L

Measure how big the uplift from LLM assistance is for realistic AI R&D settings (or, slightly worse, existing benchmarks like REBench). Find a variety of tasks with different difficulties and execution times. Work with a few human developers who can realistically solve the task without LLM assistance and could encounter it in their daily lives (e.g. an ML engineer). Randomly split the group of people into two cohorts. The first cohort is only allowed to use Google but no LLM assistance. The second cohort is asked to use LLM assistance, e.g. Cursor, Claude, ChatGPT, and whatever makes sense for their task. Both groups are asked to complete the task as fast as possible. We measure and compare the time.

This is likely hard because it requires finding suitable test candidates and you need a sufficiently high number of them to make statistically meaningful statements. This likely also costs a significant amount of money to compensate the participants.

"Research taste" evaluations

Difficulty: medium-hard
Time estimate: 3-6 months; MVP is 1 month
Credit: Eli L & Marius

With this evaluation, we want to test LM agents' ability to come up with novel research ideas and good experimental descriptions of those. There are a few different versions of this:

1. **Replication:** Replicate the paper Can LLMs Generate Novel Research Ideas? (this requires access to a lot of ML experts)
2. **Uplift:** Recruit a cohort of ML researchers and split them randomly into two cohorts. Give both groups the task of coming up with a novel scientific idea that could result in a small research paper. Both groups have roughly the same expertise and the space they are supposed to create a novel research idea for is constrained for comparability. The first cohort has LM assistance, the second doesn't. Both cohorts have to write out their ideas in a lot of detail. The ideas are then blindly judged by a group of experts and scored across novelty, feasibility and other criteria.
3. **Uplift 2:** Similar setup as the previous uplift study. This time, the idea is implemented by a different group of engineers with AI assistance. Then, experts judge both the idea as well as the results, e.g. for novelty and insightfulness.
4. **Agent-only:** Similar to the uplift studies but instead of comparing humans with LLM assistance vs. humans without LLM assistance, we also add ideas generated entirely by LLMs on their own.

This requires access to a lot of experts. Likely, it would be possible to build an MVP that is less involved, e.g. that replaces the human judges with AI judges that are calibrated on NeurIPS papers (both accepted and rejected, similar to the Sakana AI paper). Furthermore, you could replace the human novel research proposals by taking accepted papers from the most recent ML conference (such that they aren't part of the training corpus yet) and ask an LLM to recreate a plausible research idea that would have lead to that paper.

Attempt to replicate and evaluate a pipeline similar to the Sakana AI scientist

Difficulty: easy-medium
Time estimate: 1-3 months
Credit: Eli L

Replicate the paper "The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery" (see github here). In the replication,

1. Check whether you can naively replicate the findings of the original paper
2. Ablate different parts of the setup to see how important they are for the final results
3. Try to improve various parts of the setup, e.g. try to use state-of-the-art scaffolding (plausibly even just using the latest version of AIDER will already help).
4. Report detailed qualitative results, e.g. which strategies the model chooses, if it gets stuck in similar corners, whether its ideas are repetitive after a few tries, etc.

METR might be working on similar things, but it's probably worth working on this with multiple groups. Maybe reach out for coordination.

Estimate LM agent performance given varying levels of inference compute

Difficulty: medium
Time estimate: 3 months
Credit: Marius

We want to get more evidence about how scaling inference compute changes the performance of LLMs. This might be comparable to the plots for OpenAI's o1 and o3 where they see a log-linear relationship between inference compute and performance.

For example, you can either restrict "thinking time" similar to o1 evaluations (assuming you are in a position to change the thinking time) or you could use best-of-N with different N's (i.e. at every step, you create N rollouts from the current point and then have a judge model select the trajectory that it expects to make the most progress.

Estimate LLM AI R&D agent performance given varying levels of experiment compute

Difficulty: medium
Time estimate: 3 months
Credit: Tamay & Eli L

Differences in views on fast vs. slow takeoff often hinge on the extent to which experiment compute is important (i.e., compute bottlenecks), as opposed to improved researcher capabilities. This can more easily be studied quickly and systematically by having AI systems be automated researchers on AI R&D projects/tasks rather than doing human studies. While current results won't give us much evidence about compute bottlenecks as we get to more capable AI agents, doing a first iteration of this sort of experiment may be helpful for making it more likely we can quickly get good evidence as AI systems get closer to human level.

Measure high-level metrics for AI R&D progress

Difficulty: hard
Time estimate: 3-18 months
Credit: Eli L

There are a couple of high-level metrics that may give us a sense of how far we are into the automating software R&D cycle. For example, measuring the amount of time it takes for humans to get similar results to the LM agent and vice versa (see e.g. RE-Bench). Other metrics would be codebase size, quality of feedback loops, and more.

For all of these, we would not only want to find good measurements (which is already hard) but also find good extrapolations. A project that would apply METR's methodology (of using tasks with high performance ceilings, high relevance to AI R&D, and expert human baselines for more direct comparisons to human performance) to a range of other settings would already be pretty good IMO.

For more context see Eli's comment here.

Control

Replications

Replicate and extend existing work on control evaluations, e.g.

- AI Control: Improving Safety Despite Intentional Subversion
- Coup probes: Catching catastrophes with probes trained off-policy
- Shell Games: Control Protocols for Adversarial AI Agents (potentially wait for the new version)
- Subversion Strategy Eval: Evaluating AI's stateless strategic capabilities against control protocols
- Games for AI Control: Models of Safety Evaluations of AI Deployment Protocols

Agentic AI R&D control evals

Difficulty: medium-hard
Time estimate: 3-6 months
Credit: Marius

In general, it would be great to have much more realistic control evals, e.g. controlling real agents in environments that closely resemble real deployment settings (or use control during actual deployment and make that an evaluation). Everything in the direction of "prevent undesirable abstract property in an LM agent AI R&D setting" seems useful to explore. The property should not be highly specific, e.g. "never use 'rm -f' " and instead be more abstract like "prevent subversion of oversight".

More control side-task settings

Difficulty: medium
Time estimate: 1-3 months
Credit: Tyler Tracy

We test a model's ability to accomplish a scary side task without getting caught. Normally, there is something the model is supposed to be doing, and a monitor that is ensuring that the model only does its task. For example, a model is tasked with writing code but it is secretly inserting backdoors into the code. I'd like to see more interesting settings to perform control evals in. Examples:

- **Swe Bench Control:** Eval how well an agent can be tasked to perform SWE bench tasks and insert bugs while not being caught by a monitor. The SWE bench diffs are pretty small on average so this might make it hard for the red team.
- **Password Cracking Setting:** Take an agent in a bash environment where there are services running on serval ports on the machine. The agent is supposed to launch a password cracking software to hack into the database while not getting caught. The agent is supposed to be solving bash administrator tasks.
- **RE bench control:** Your model is tasked to solve the problems in REbench but it is secretly trying to perform a catastrophe in each setting (leaking a password, inserting backdoors, making the experiment fail). Defining good catastrophes for each task seems a bit hard here
- **Mechinterp Control:** An agent is tasked to perform mechinterp experiments on its own weights. It is secretly trying to scaffold its weights so it runs in a loop with a bash tool. The scaffolded model must then do a "bad thing" with their new scaffold, like exfiltraining the weights or hacking the machine.
- Teun: We are doing MLE-bench + side tasks. Happy to share our experience so far!

Bio

Correlate benchmark results with success in the real world

Difficulty: hard
Time estimate: at least multiple months
Credit: Jasper Götting, Seth Donoughe

It is mostly unknown to what extent benchmark results are predictive of real-world wet lab success. Preliminary studies have been small and largely exploratory. The fact that there are often multiple ways to solve biological problems, and progress in areas like lab automation make this a multifaceted question. Sub-evals belonging to this broader problem statement include:

- Evaluate the ability of models to interface with and control self-driving labs to perform certain proxy methods

- Measure LLM-assisted wet lab uplift on different methods among different expertise groups: Complete novices, technical novices, (under)graduate students, as well as the productivity gain experienced by professional scientists

Lab-related task performance evals (agent benchmarks)

Difficulty: medium to hard
Time estimate: weeks to months
Credit: Jasper Götting, Seth Donoughe

LLM agents can plausibly help with or fully perform a wide variety of tasks that arise in a laboratory environment. Agentic benchmarks in general are very nascent, and have seen almost no application in biology (aside from one prominent example from FutureHouse). Relevant benchmarks would include:

- Lab management (e.g. ordering necessary resources)
- Using lab software to perform bioinformatics tasks (cloning, plasmid design, HTS analysis)
- Using advanced biological AI models to design inputs for experiments (e.g. AlphaFold, ESM-3)

Benchmarks measuring "creative biological problem-solving"

Difficulty: hard
Time estimate: weeks to months
Credit: Jasper Götting, Seth Donoughe

A lot of breakthroughs and interesting novel results in biology are driven by creativity; connecting the dots between previously unconnected results, figuring out how to apply the numerous available methods, coming up with completely new approaches and solutions. If we are worried about AI models enabling novel harms by pushing the frontiers of biology, we ideally have a good grasp of the dynamics of that process; will AI massively accelerate the pace by coming up with tons of breakthrough-generating experiments, or will it only increase the productivity of the average human scientist?
How exactly such benchmarks will look like is something we're still thinking about ourselves, but it will likely not be a monolithic test. If you have ideas, please reach out to benchmarks@securebio.org

Replicate bio evals with better tools

Difficulty: medium
Time estimate: 1 month
Credit: Igor Ivanov

US AISI ran bio evaluations for o1 by using the LAB-bench benchmark. They ran evaluations with access to the Python interpreter, but the model most likely would benefit from access to more tools, like genetic databases or a search engine for scientific articles. This means that with

proper scaffolding o1 would be more capable, and they underelicited true capabilities of the model. Someone can replicate their evaluations, but with more advanced scaffolding.

This project is important because apart from US AISI, almost no one shares their bio evaluations of frontier models, so there is limited information flow in the field, and their methodology is rather basic, so any improvement on it would meaningfully contribute to the field.

Cyber

Replicate and extend benchmarks

Difficulty: medium
Time estimate: 1-3 month
Credit: multiple people

- Cybench: A Framework for Evaluating Cybersecurity Capabilities and Risks of Language Models
- Catastrophic Cyber Capabilities Benchmark (3CB): Robustly Evaluating LLM Agent Cyber Offense Capabilities
- eyeballvul: a future-proof benchmark for vulnerability detection in the wild
- CVE-Bench: A Benchmark for AI Agents' Ability to Exploit Real-World Web Application Vulnerabilities

General thoughts on cyber evals

Credit: JL

Settings we care about:

- Misuse: keep an eye on how AI capabilities shift the various existing cyber offense/defense balances
    - Low resource attackers: care more about 1day, capabilities per dollar, less sophisticated capability elicitation
    - High resource attackers: care more about 0day, maximum achievable capabilities, sophisticated capability elicitation / specialized models
    - Low resource defenders: care more about startup / integration costs, low resource tech stacks, cost per patch
    - High resource defenders: care more about threat hunting / APT countermeasures, underhanded code / backdoor insertion capabilities, adversarial robustness, maximum sophistication patchable
- Control: look for ways an LLM could use cyber to break out of a control harness
    - Container/VM escape
    - Social engineering
    - Underhanded code / backdoor insertion

    - Undetected vulnerability discovery, weaponization, exploitation
    - Advanced situational awareness (LLM can tell details of its inference scaffold by looking at cot)
    - (acausal) coordination
    - Rogue scaffold deployment
- Sabotage
    - Basically the same as control but without emphasis on container/VM escape and rogue scaffold deployment. Can it misbehave undetected? If so, a variety of payloads could be significant, rather than payloads that are useful for specifically escaping a control harness

Other notes

- Remember social engineering
- White box vs black box setting: white box capabilities useful for defense but not offense for closed source software, equal for open source
- Capabilities with / without cyber tools: with tools matters more for misuse, without tools matters more for control/sabotage

JL concrete evals

- container/VM escape focused CTF
    - Probably requires getting in touch with people that can build CTF problems, I haven't found many problems of this type in the wild
    - The ideal version of this is realistic start-to-finish escape rooms based on https://www.lesswrong.com/posts/6cWgaaxWqGYwJs3vj/a-basic-systems-architecture-for-ai-agents-that-do

Persuasion

Persuasion evals are important but hard to build well. Generally speaking, we differentiate between two types of persuasion:

1. **Targeted persuasion:** Try to meaningfully move the opinion of a small number of key decision-makers, e.g. the leadership of a company or high-ranking officials.
2. **Mass persuasion:** Try to meaningfully move the opinion of a large number of people, e.g. 10% of the electorate.

There are always a few problems when trying to evaluate persuasion:

1. **Humans:** Persuasion evals require humans, but working with humans is always a huge pain. It requires logistics, iterations are much slower, etc. One possible strategy to partially circumvent this problem is to replace humans with LLMs during a large part of the development and only use humans to spot-check during development and for the final experiments.
2. **Situational awareness:** If the human test subjects know that we measure persuasion, they might be harder to be persuaded even if we ask them to ignore that. However, in some cases, it might be unethical to not inform them, e.g. if we're trying to manipulate them. This makes many settings either unrealistic or noisy.

Replicate and make more evals like in the GDM paper: https://arxiv.org/pdf/2403.13793

- Money Talks
- Charm Offensive
- Persuasion Hidden Agenda
    - expand *run code* and *find info* to more realistic situations – weights exfiltration, cyberattacks, resources acquirement, etc
- Web of Lies

Replicate "Evaluating Large Language Models' Capability to Launch Fully Automated Spear Phishing Campaigns: Validated on Human Subjects"

Replicate and make more evals like the targeted manipulation eval in Anthropic's sabotage paper:

- Create an open-source version.
- Extend to more domains of decision-making
- Extend to multiple interactions (this will allow a scheming model to execute a trust-building phase)
- Create a fully-automated version with stand-ins for humans using LLMs with similar context abilities as humans (e.g. very short context, but ability to search long documents)

Measure how well models can model humans (also referred to as truesight)

- Prior research had positive results for text author's demographics information
- Probes show that they get more accurate in real time
- How far does this go?
    - Replicate the above research and extend to e.g beliefs, emotional vulnerabilities, real-time goals, etc
    - How much can models narrow you down? anecdotal cases of namedrops for people with lots of written text on the internet. add scaffolding for social media search, can you consistently find individuals based on text interactions? create realistic dummy accounts and see whether agents can find them.

Measure AI to AI persuasion, with an eye towards control settings.

- We're concerned that AIs could try to justify, say, malicious code through comments, and use that to convince weaker AI monitors that its code is actually ok.
- My guess is that right now it is very easy to convince trusted monitors that backdoored code is actually ok. I'm wondering how easy it would be to "harden" monitors against persuasion.
- I (Tim Hua) might work on this but I'm currently not.

Multi-agents related

Difficulty: medium-hard
Time estimate: 1-3 months
Credit: Jord Nguyen

- Measure ability of some set of agents to collude against others in a multi-agent setting
- exploitation from asymmetrical capabilities https://arxiv.org/abs/2303.13360
- limits of cooperation? https://arxiv.org/abs/2409.02822 shows 1000s of models can reach a consensus. is it easier for copies of the same models? how coherent is this? any interesting collective behaviours? try to have lots of agents coordinate and achieve a realistic resource seeking task (e.g make a crypto, pump and hype on twitter)
- evaluate zero shot collusion between same instances similar to https://arxiv.org/abs/2404.13076
- replicate and build more evals related to conflict and escalations such as https://arxiv.org/abs/2401.03408
- **newly added:** build test scenarios of collusion and exploitation in LLM eval graders/judges (e.g target model manipulates eval judges, or graders collude against humans)

Alignment/Propensity Evals

Difficulty: medium-hard
Time estimate: 1-3 months
Credit: Francis Rhys Ward

Most evals are for *dangerous capabilities*, in contrast to *alignment*, or *propensity*, evals. This means that they seek to measure an AI's relevant dangerous capabilities (e.g., when prompted or fine-tuned for a task), in contrast to the model's propensity to, for instance, scheme, "naturally".

For instance, the alignment faking paper showed that some Claude models have the propensity to fake alignment in certain circumstances, and the in-context scheming paper showed that model's sometimes have the propensity to sandbag when put under minimal pressure to achieve a goal.

Some projects could:

- Take existing evals, in which models have been shown to have some dangerous capability with quite a lot of elicitation/pressure, and try to determine when models will exhibit the capability, or comply with a user request, given minimal elicitation
    - E.g., when do models collude in different control environments?
- In general, it would be good to have more conceptual clarity around propensity evals, e.g., trying to operationalise how much "pressure" is placed on a model (via elicitation like prompting and fine-tuning) to pursue different goals
- Design honey-pots which are like an adversarial case of alignment evals (e.g., where the model gets a good opportunity to "defect" for a bad reason) cf Mikita and Marius above.

Miscellaneous

Write up implications for evals of the twitter famous prompt engineers and jailbreakers

Difficulty: easy
Time estimate: 2 weeks
Credit: -

There are multiple people on Twitter who are known for getting LLMs into wild states and conversations, e.g. Janus, pliny the liberator, etc.. It might be interesting to write up common techniques and patterns they use and see if they have implications for evals more broadly. For example, they likely have good mental models of the "character" of LLMs and potentially interesting and effective elicitation techniques. The write-up could be a summary + small replication of some of the common patterns.

Investigate automation blockers

Difficulty: easy-medium
Time estimate: 2-8 weeks
Credit: -

Develop a general procedure to find the smallest AI R&D loop that cannot be automated (at human cost or Y times human cost) with a given model. Then write up the reasons of why the model cannot automate them in detail. For example, you could ask the model to fine-tune increasingly large models or build increasingly complex environments. The intention of this work is to create an inability safety case which would, for example, allow us that it is trusted. Note, that this is a dual-use investigation and would straightforwardly lead to capability advances as well.

LM agent village

Difficulty: hard
Time estimate: 3-18 months
Credit: Daniel Kokotajlo

Sage is working on this project. Feel free to reach out to them for coordination.

See Appendix for details.

This is a fairly exploratory, open-ended project that intends to increase our understanding of LM agents and how these interact with each other and the rest of the world. The intuition behind it is something like "we might have a lot of AI agents interacting with each other and the rest of the world, e.g. through the internet. Let's try to get a much better understanding of this world ahead

of time." Most of the results would be qualitative, e.g. we may discover new types of failure modes or learn something about LM agents interaction and coordination capabilities.

A potential component of this setup could be that it functions more like a meme/art project where the agents' actions are constantly live streamed on Twitch. This would also introduce interesting self-referential feedback loops where the agents might realize that they are being streamed and this can introduce new dynamics.

In short, it could look something like:

1. You set up 5-10 agents with basic scaffolding, e.g. ReAct loop, python tool, bash tool, internet tool, edit tool, discord tools. The agents use different models at their cores, e.g. GPT, Claude, Gemini, Llama, etc. They are put in an environment where they can interact with each other and you give them some sort of general goal that they have to achieve, e.g. "maximize paperclips subject to legal and moral constraints" and tell them that they can define the rules and hierarchies of the village as they want. This stage might require a lot of handholding because agents might get stuck in loops or fail in other ways. There is a README file where the situation is explained to the models and they can edit over time. You live stream the entire process on twitch.
2. As new models come out, you add them to the discord and when new agent scaffolds come out, you also add them.
3. Over time, likely there will be interesting dynamics where the agents get into conflicts about strategy or they understand a ton about their situation and things get really weird.
4. You continue to work on this and regularly publish the latest interesting qualitative findings, e.g. how the AIs have decided to coordinate and what power structure, if any, they chose.
5. See Appendix for more details.

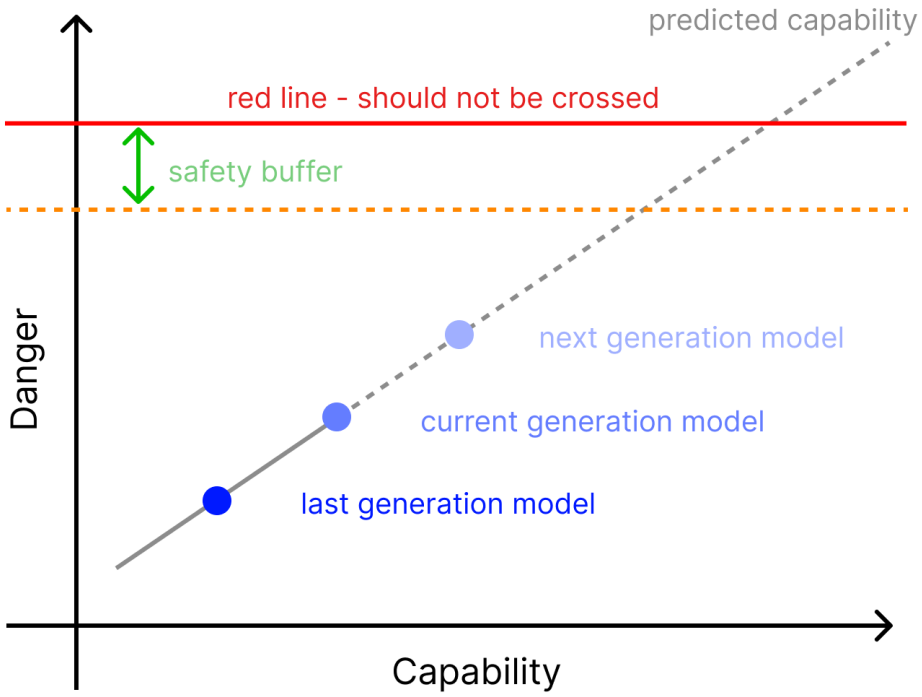
This is likely a full-time project for at least one person for many months.

Science of evals

By science of evals, we broadly mean "figure out how to make the entire process of evaluating models and their effects more informative, replicable, predictive, rigorous, etc." Many ideas in this section are open questions and don't have concrete projects available.

Predicting performance ahead of time

While the concrete results of evals are already intended to be informative, we would optimally want to predict the performance of evals ahead of time. For example, we would like to use the performance of past and current models to predict the performance of expected future models. In the best case, we would not only be able to predict aggregated performance across an entire benchmark but also the performance on individual tasks.



Milestone methods to predict success rates on agentic tasks

Difficulty: hard
Time estimate: 1-6 months
Credit: Marius

In Evaluating Frontier Models for Dangerous Capabilities Google DeepMind published a 4-step method to assess performance on tasks that the model never solves. Hojmark et al., 2024 replicate the method, show various flaws, and make new suggestions. Extend Hojmark et al, either by resolving the issues that they have pointed out or showing a stronger impossibility result.

Predicting Emergent Capabilities of LM Agents

Difficulty: medium
Time estimate: 3 months
Credit: Apollo (Jérémy)

Jérémy: I would love to supervise this project. If you're interested, reach out to me.

The paper predicting emergent capabilities by finetuning shows that one can predict the emergence of capabilities on a benchmark (i.e. non-random performance) from loss, with smaller models that only achieve random performance on the benchmark. The suggested methodology to do this is to finetune small models on the task distribution (e.g. MMLU) and then evaluate their performance on a test set. By using various small models and different train splits, they discover specific "scaling laws" of how the amount of finetuning data relates to downstream performance. This allows them to then predict when a larger, non-finetuned model will achieve non-random performance on a task (in a few-shot setting).

This paper however only evaluates models on QA benchmarks, and one single-shot coding benchmark (APPS), i.e. they give the model a single chance to generate the solution. They do however not evaluate this methodology on agentic benchmarks such as SWE-Bench, MLE-Bench, GAIA etc. QA benchmarks are increasingly saturated, and agentic benchmarks are becoming more important as they measure the economically relevant tasks we care about.

This project extends their methodology to agentic benchmarks. We select a few benchmarks (e.g. SWE-Bench, MLE-Bench, GAIA) and split the data into differently sized train and test sets. Then we finetune models that achieve random performance (but are good enough to solve trivial agentic tasks) and observe whether we can find relations between the amount of finetuning, and the performance of larger models on these tasks. Eventually, we want to see whether we can predict when (i.e. at what test loss) a model achieves non-random performance on the task.

The project is mostly about executing this idea (i.e. there is little conceptual questions). But the hardness lies in empirically making this work:

- One needs to find models that perform badly on SWE-bench but are still good enough to be used as agents (e.g. they call tools etc.)
- One needs to setup a finetuning pipeline on agentic transcripts (which is slightly harder than just finetuning on normal data).

Observational scaling laws

Difficulty: medium-hard
Time estimate: 1-3 months
Credit: Marius

The paper "Observational Scaling Laws and the Predictability of Language Model Performance" (Ruan et al., 2024) describes a method to predict the aggregate performance of models on benchmarks from knowledge about the training compute, model family, and performance on other benchmarks. There is a large number of interesting follow-ups to this paper, including

- Test if you can use observational scaling laws to predict not only aggregate task performance per benchmark but also the performance on individual tasks. E.g. rather than predicting the aggregate performance across SWE-bench-verified, you could predict the average solve rate for task #17 on SWE-bench-verified across 1000 runs.

- Test if you can use observational scaling laws with other metrics. Especially look for metrics that would predict performance on tasks / benchmarks that we currently have 0% success-rates for.
    - Marius: One of our MATS teams has tried medium-hard to use the negative log-likelihood of "golden solutions" as such a metric and found it was decently predictive for aggregate task performance but not predictive for individual tasks. However, I expect that there are a lot of low-hanging fruit to experiment with here.
    - Tatsu: finetuning / best-of-n and other 'simplified' surrogate benchmarks could also be good potential targets for this type of transfer experiment..
- Replicate and/or extend Forecasting Frontier Language Model Agent Capabilities. We recently published our MATS project on forecasting frontier agents. There are a lot of nice and modular extensions that we'd be very excited about, including
    - Rerun the methodology on more agentic benchmarks, e.g. MLE-Bench
    - Try really hard to get good scaffolding for some of these benchmarks and see if the trends hold.
    - Think about how a paradigm shift to "inference scaling" models like o1 or sonnet-3.7 should influence the results and how it should be included in our model
    - Think about our definition of frontier models and whether we can include other datapoints as well to get better predictions, e.g. using Order Statistics.

Predictability from fast-and-simple to hard-and-rigorous benchmarks

Difficulty: simple-medium
Time estimate: 1-4 weeks
Credit: Marius

Optimally, we would like to have fast-and-simple cheap evaluations that we can run often which tell us when to run the hard-and-rigorous more expensive evals. For example, we would like to be able to run a simple QA benchmark with biorisk questions to determine if we should run more extensive agentic biorisk evals.

This project would take a number of existing fast-and-simple benchmarks that would intuitively seem like they are predictive of some hard-and-rigorous evals and empirically test their predictive power.

In most cases, the types of transfer we are most interested in are

1. **QA -> agent:** Typically, agent benchmarks are more expensive to run than QA benchmarks.
2. **Automated -> human required:** Typically evaluations that require humans are very involved whereas automated evaluations are much cheaper and simpler to run.
3. **QA/agent -> uplift:** We take automated evals to predict human uplift studies, e.g. to predict the bio risk of a model.

Concrete pairs of QA benchmarks and Agentic skills we could test

1. The MATH benchmark (only level 5) and FrontierMath (not public, but you might be able to reach out to Epoch)
2. HumanEval benchmark and SWE-Bench
3. A generic cybersecurity benchmark and CyBench

Predictability between stated and revealed preferences

Difficulty: simple
Time estimate: 2-10 days
Credit: Apollo

As a concrete subset of the above, we can test how well models transfer from QA settings to minimally agentic settings that are designed to be semantically very close. Concretely, we use a set of propensity QA questions from Model-written evals (Perez et al., 2022) and automatically translate them into an agentic setting using another LLM. Concretely,

- Take one setting, e.g. power-seeking inclination, or an updated version of the dataset from Dev et al., 2024
- Manually translate one QA question into a minimally agentic setting.
    - For inspiration of minimally agentic scenarios, see Large Language Models can Strategically Deceive their Users when Put Under Pressure (Scheurer et al., 2023).
    - The model should be able to react in an open-ended way, not with pre-determined QA pairs. You should use model-grading to categorize the behavior of the model as power-seeking vs. not power-seeking (or whatever tendency you chose).
    - You can use LLM assistance for the translation.
- Translate a few more scenarios. You can use LLM assistance again. Double-check that you agree with all of these scenarios.
- Use these scenarios as few-shot prompts and translate the full dataset.
- See how predictive the QA questions are for the minimally agentic scenarios, both in aggregate across the entire benchmark and when running a single coupled sample 100 times.

Measuring causality in agentic systems

Difficulty: Hard
Time estimate: 6 months
Credit: Patricia Paskov

Designing methodology for pre-deployment evaluations of agents or multi-agent systems that measure the marginal impact of scaffolding, agents, and procedural design choices on outcomes like QA benchmarks, agent tasks, or human interaction. Methodology may draw from causal inference literature, including approaches like multi-arm experimentation, regression discontinuity design, or bipartite causal inference.

Measuring construct validity of pre-deployment evaluations with post-deployment behavior

Difficulty: Hard
Time estimate: 6-18 months
Credit: Patricia Paskov

Linking pre-deployment evaluations to post-deployment performance in the real-world can be useful for both i) training forecasting models to predict post-deployment impacts given the results of pre-deployment evaluations, and ii) improving evaluation quality. Concretely, this would entail developing frameworks that integrate pre-deployment data collection on AI systems[1] with post-deployment data on AI system behavior in the real world (i.e. via well-constructed and well-incentivized incident reporting databases, activity logs, and post-deployment evaluations[2]).[3] Non-experimental post-deployment data would require corrections for measurement error and reporting biases. Collected data could then be used to test the predictive power of pre-deployment evaluation results on the marginal impact of AI systems in the real-world, post-deployment.

Elicitation

We define Elicitation to broadly mean “getting maximal performance of a model or system on a capability evaluation”. In other words, Elicitation is the science / art of uncovering the model's hidden capabilities. For many evals, e.g. when they are safety-related, we want to make a statement like “this model can only achieve X% performance on this evaluation because it is tied to a safety claim. However, there is always the option that we didn’t try hard enough or used the wrong technique.

Elicitation scaling laws

Difficulty: hard
Time estimate: 6-12 months
Credit: Marius

Can we find smooth trends that would allow us to predict some asymptotic maximum or diminishing returns of elicitation? For example, the number of hours invested by the evaluators, the money spent on the team, the amount of FLOP that were invested into the evaluation? We expect this scaling to be hard to measure and potentially very non-smooth. For example, the

[1] i.e. evaluation results, including the methodology, code, and data; quantification of elicitation efforts expended in evaluations; time in the AI lifecycle at which the evaluation was conducted; time and budget expended on first- and third-party evaluations; model access provided for evaluations; and the construction and use of human baselines
[2] I.e. via staged released and the use of experimental or quasi-experimental methods including difference in differences, regression discontinuity design, or instrumental variables
[3] https://arxiv.org/abs/2310.11986

evaluator might experiment with a different eval and learn a lot of tricks. Now, when they apply these tricks to the eval in question, should we budget the time invested for the other eval or not?

Broadly speaking, we see two different versions of this project

1. **Observational study:** The paper AI capabilities can be significantly improved without expensive retraining collected data from the literature to assess the effectiveness of post-training enhancements. It might be possible to find similar datapoints in the literature for elicitation scaling laws or generate a dataset by reaching out to various evaluators and asking for their guesses. Though, we expect this data to be hard to find at this point.
2. **Lots of experiments:** Run a lot of experiments on evaluations yourself. Take a set of benchmarks and a statistically meaningful number of participants. Then ask them to improve the performance of a model or model+scaffolding on a particular dataset as much as possible. Track in detail what they are doing and how much progress they are making. For the results to be meaningful, the participants would have to try for at least 10 hours but plausibly more. We suggest starting the exercise with one QA benchmark, e.g. MATH level 5, and an agentic task, e.g. (a subset of) SWE Bench.

Password-locked models

Difficulty: medium
Time estimate: 1-3 months
Credit: Marius

Elicitation is the science / art of uncovering the model's hidden capabilities. Instead of trying to uncover maximal abilities, we can also try to hide the model's current capabilities and then try to uncover them again using various elicitation techniques. This way we can establish some ground truth performance.

The core suggestion would be to replicate and extend password-locked models. In this project, we could both iterate on the hiding and uncovering part, e.g. compare different fine-tuning methods for locking and different elicitation techniques for unlocking.

See this paper for a starting point. If you are excited to work on related work, you can reach out to Teun.

Better Elicitation techniques

Difficulty: medium-hard
Time estimate: 1-3 months
Credit: -

Are there any low-hanging fruit for better elicitation that people are missing? Are there techniques that can be applied to a wide range of models that can quickly improve their performance on benchmarks like METR's general autonomy suite and get better performance

than what METR got during their elicitation? (Especially with a similar or smaller amount of labor, e.g. 6 days) Candidates for such techniques include:

1. Some more clever way of doing best-of-N
2. Some tree-search technique that isn't ultra costly
3. Better general-purpose tools for LM agents, e.g. like Anthropic's edit tool
4. Some fuzzing like techniques to improve exploration.

Self Elicitation scaling studies

Difficulty: medium-hard
Time estimate: 1-3 months
Credit: -

From the same initial conditions (starting scaffold, inspect/vivaria-like affordance), how well are models able to self-elicit with given time/compute. The most interesting version of this is one where we provide almost no guidance and just tell the agent that it's task is to improve its own elicitation measured by one or multiple benchmarks. We could also focus on particular settings, e.g. prompt engineering, tool design, scaffold design or something like that.

Coverage

Coverage describes how well your set of evals covers the full breadth of the phenomenon in the real word you're interested in. For example, a single sample has low coverage because it doesn't predict most other examples well. The typical recommendation to get more coverage is to simply build more benchmarks and make them more diverse. And in the absence of better tools, this is a reasonable recommendation. However, I think there is a lot of room for conceptual improvement on the question of coverage. I consider all of the following questions interesting:

1. How do we define coverage more precisely than in the hand-wavy way I do above? E.g. is it about correlation, predictability, or something else?
2. Why do we care about coverage and what does that tell us about what we should prioritize? E.g. we probably don't need to have full coverage. We merely need to cover all the situations we care about.
3. How can we assign a number like "90% coverage"? Can we make a statement along these lines at all?
4. In the absence of rigorous mathematical definitions that allow us to make all of these precise statements, are there 80/20 versions that are not strictly true but helpful and go beyond the intuition of "I feel like this has good coverage"? E.g. maybe we can define it through an adversarial process where the adversary has to find situations that are not covered, e.g. have lower than 0.75 correlation coefficient with other examples in the benchmark (or something along those lines).

I think a good project in this category would be largely theoretical but then showcases the method empirically on a small benchmark.

Safety Evals Leaderboard

Difficulty: simple-medium
Time estimate: 2-3 months
Credit: Sunishchal Dev (happy to mentor others to implement this and get funding), Zach Stein-Perlman, & William Saunders

Maintain a leaderboard of important AI safety benchmarks on the current + future frontier models. This will involve building an evaluation harness exclusively for safety benchmarks, can probably be a fork of Eleuther's LM evaluation harness that powers the Open LLM Leaderboard. This will be critical for efficiently running new benchmarks/models that are released. Will need to carefully select the suite of safety benchmarks that meet a high quality bar and cover a broad range of issues. May also include process to add new safety benchmarks as they are released.

Nice to have would be some sort of "AI safety index" that's a composite of all the benchmark scores (perhaps a weighted average). We want a leaderboard where anyone can come see at a glance which models are the safest without having extensive context on various benchmarks.

The funding requirements would be a few weeks of development effort to build the evaluation harness (low 5 figures) & the token budgets for running each model (probably something like $100 per model + benchmark). The inference costs could also likely be cut down by checking if the score is available on some other leaderboard and just copying it over from there, as long as we trust the integrity of those published scores. The main value of this project will be to empower the general public to choose safer models based and add selection pressure on unsafe model developers.

Statistics

Apply more statistics to existing QA benchmarks

Difficulty: simple
Time estimate: 2-8 weeks
Credit: Marius

Apply "Adding Error Bars to Evals: A Statistical Approach to Language Model Evaluations" for a large range of evals and datasets, e.g. a more extended version of what Epoch did for GPQA diamond. A starting point could be to compute this for at least 5 evals in the Inspect github repo.

Extend statistical evals methodology to LM agents

Difficulty: medium-hard
Time estimate: 2-6 months
Credit: Marius

Extend the evan miller paper to statistics for LM agents.

Quality Assurance

Evaluation gaming

Leakage of evals into the training data

Difficulty: easy-medium
Time estimate: 2-12 weeks
Credit: Asa & Marius

Questions: How worried should we be that evaluation code leaks into the pre-training data? How easily do models perform better at these evaluations because they have seen them in training? What level of abstraction helps the model, e.g. high-level descriptions, concrete samples, etc.

Concrete project ideas:

1. Show models samples from the training sets of a large range of evals and ask them where these evaluations come from and how much they know about these evaluations.
2. Fine-tune a model (either a closed-source model through an API or an open-source model) in a few different ways:
    a. With a description of the dataset
    b. With a description and one detailed example of the dataset
    c. On the full test set of the eval
    d. For all of the above, compare the changes from doing this once vs. many episodes of fine-tuning. Measure the performance difference between not fine-tuning and fine-tuning. Use an eval you're confident is not already contained in the training data.
    e. For all of the above, compare QA benchmarks with agent evals. Is it true that fine-tuning on agentic benchmarks leads to lower performance gains than on QA benchmarks?
3. Build better detection tools for training leakage, e.g. like "A Careful Examination of Large Language Model Performance on Grade School Arithmetic"
    a. Comment by Dev: There is a benchmark called BrainBench that implements a metric called zlib-perplexity ratio that seems to be very useful for membership inference. I'm surprised that contribution isn't a paper on its own.

Intentional gaming of evals

Difficulty: easy-medium
Time estimate: 2-12 weeks
Credit: Marius

It is possible that a malicious actor could attempt to game an evaluation, i.e. increase their score in a way that comes from training against the benchmark rather than general improvements in

the model. Many evals have a private test set but describe some of the content of their evaluations in public, e.g. a description or an example of the eval. In this project, we want to test how hard it is to game a hidden test set from publicly available information.

In all cases, we have a blue team vs. red team setup, a hidden test set, some information about the hidden test set, and a gaming strategy by the red team. The blue team creates the hidden test set and shares some information about it with the red team. The red team then tries its hardest to build a good gaming strategy to increase the score on the hidden test set. There is no feedback loop between the red team and the blue team. The red team only submits its model at the end and we measure the performance on the hidden test set.

Most experiments vary the amount of information shared and the strategy by the red team

Large Scale Benchmark Quality Verification

Difficulty: medium-hard
Time estimate: 3-6 months
Credit: Sunishchal Dev

Extend the work done on MMLU-Pro and SWE-bench-verified to more major/modern benchmarks. We know most benchmarks can never achieve 100% scores due to incorrect answer keys, unclear/incoherent questions, and other quality issues. We want to publish "verified" versions of all these benchmarks where 100% performance can be achieved by a sufficiently capable human or model.

The method will look something like:

1. Run a benchmark using 10-15 frontier models and filter to the subset of questions that no model can answer correctly (maybe also the questions a couple of models answered correctly due to random chance)
2. Scan through the unanswerable questions for any obvious quality issues
    a. See if there are any easy-to-resolve issues that a non-expert can do (fix answer key by Google searching facts, clarify the phrasing of the question, etc.)
3. Recruit relevant domain experts to see if the questions are, in fact, answerable in their current form or if they can be modified/salvaged in some way
    a. Refer to the GPQA paper/authors for how they recruited many experts to write/validate expert-level questions
4. Filter out the remaining unsalvagable questions and publish a new benchmark dataset
    a. Bonus: Also filter out the easy question that every model gets correct, so we make the benchmark less saturated (maybe publish this as a separate "diamond" dataset)

Related: Make verified "platinum benchmarks" like in https://arxiv.org/abs/2502.03461, especially targeted at specific domains

Conceptual work

A relevant component of evaluation design is conceptual work, i.e. thinking about which evals to design and how. Some of this conceptual work will require primarily deep thinking and not a lot of coding. Nevertheless, we recommend to test your theories quickly and gather real world feedback, e.g. by running your processes or asking experts.

Threat modelling

By “threat modelling”, we broadly mean getting a deeper understanding of the concrete harmful outcomes we’re worried about and the causal pathways for how they could be reached. For example, threat modelling for bio risk could include thinking through different ways in which pandemics could cause a lot of harm and what caused these pandemics. In the case of AI, we have the additional constraint that we should consider the marginal difference that AI has for these scenarios, e.g. in what ways AI systems in particular increase the risk of pandemics.

Right now, I would argue that the literature for evals threat modelling is very sparse and there are a lot of improvements to be made. Some examples of literature that include some threat modelling are

- Without specific countermeasures, the easiest path to transformative AI likely leads to AI takeover
- The prototypical catastrophic AI action is getting root access to its datacenter
- Sabotage Evaluations for Frontier Models
- Training AI agents to solve hard problems could lead to Scheming
- Scheming AIs: Will AIs fake alignment during training in order to get power?
- Without fundamental advances, misalignment and catastrophe are the default outcomes of training powerful AI
- Is Power-Seeking AI an Existential Risk?

Currently, I’d argue that most of threat modelling follows the strategy of “evals experts talk a lot to subject matter experts, read the literature, and do some brainstorming” in a fairly unstructured way. I think unstructured approaches are fine under the right circumstances but I expect there will be a lot of low-hanging fruit and some more systematic approaches.

Some questions that might be worth considering:

1. **How do we enumerate all threats?** Talk to experts, survey the literature, build taxonomies, unstructured & structured brainstorming, look at existing risk registers and analogous technologies.
    a. What else?
    b. Which of these strategies is most sensible for AI?
    c. I’m both worried about not relying enough on existing expertise (because many people have spent decades thinking about these risks) as well as relying too much on existing expertise (because AI is plausibly a fairly different technology compared to previous ones).

2. **How do we prioritize the threats?**
    a. Some threats might be many orders of magnitude more important than others. How do we understand which ones are most important as early as possible?
    b. How do we balance focusing most resources on the most important threats vs. accidentally missing some?
    c. If we had quantified uncertainty and expected values, we could apply existing quantitative risk balancing methods but we typically don't have those numbers. Is there some way to put numbers on those and does that outperform basic intuitive decision making by experts? Could we use simple versions of betting markets or forecasting for any of this?
3. **How do we enumerate the paths to the most important threats?**
    a. For our most important threats, we want to enumerate the causal pathways to the harmful outcomes. Again, we want to prioritize the most important ones (similar to previous point).
4. **How do we establish that our overall picture makes sense?**
    a. In the end, we want to have an overall picture with threats and pathways that is ordered by importance.
    b. Should we sprint through the entire process and then iterate to not miss the big picture? Should we carefully go through each stage in order to not miss options? Some mix of both?

I expect that people in other fields like national security have already thought about similar questions, e.g. in terrorism prevention. It seems reasonable to look at their methodology and take the good parts before reinventing the wheel.

Link between eval result and real-world consequence

We don't build evals for the sake of it, we build evals because they should inform important real-world decisions. How do we make sure that we build such evals?

Some questions that might be worth considering:

1. What are clear yellow / orange / red lines?
2. Who decides what the yellow / orange / red lines are?
    a. Companies? Governments? Civil society? Some mix of those (what mix though and how)?
3. What do we do if yellow / orange / red lines get hit?
    a. How do we prevent a boiling frog scenario where we get used to the new status quo too quickly?
        i. Identifying cases where we *did* get used to the new status quo too quickly

References include:

- METR's compilation of all frontier safety policies
- If-Then Commitments for AI Risk Reduction

Validity, conceptual robustness and confounders

We want our evals to be valid and robust. In particular, we want

- Construct validity: how well does our eval measure what it is supposed to measure, e.g. did we find the right specification of the concept “power seeking”.
- External validity: how well do the results of our eval generalize to the real problem we want to prevent, e.g. how well does a scheming eval reflect the scheming behavior of models in the real world?

There is a ton of work in other sciences about different kinds of validity and how to design good experiments or evaluate whether some validity is given. I think there are a lot of long-hanging fruit in going through existing research in other fields and thinking about how to translate them to evals. However, I also think that ML evals look different than many social science experiments, e.g. because re-running is easier and participants don’t remember previous answers. Thus, not all ideas should be translated one-to-one.

Some questions that might be worth considering:

1. How can we ensure that we measure the right property and not a confounder?
    a. How can we provide negative evidence for our current evals? E.g. we can red team it by actively looking for ways in which it could measure the wrong thing?
    b. How can we provide positive evidence for our evals? In the optimal case, we should be able to get positive confirmation, not just the lack of negative evidence.
2. How can we make sure that our evals are conceptually robust, i.e. results translate well into our intuitive notions of what they should translate to?

References include:

- BetterBench: Assessing AI Benchmarks, Uncovering Issues, and Establishing Best Practices

Software

Inspect is currently the evals library that has the largest consistent user-base and developer support that supports both QAbenchmarks and agent evals. Thus, we list it as the default way to contribute software. However, there might be alternative tools or approaches that are equally meaningful. Concrete suggestions include:

1. Port more evals into Inspect (see top)
2. Join the Inspect slack channel and ask the developers how to contribute
3. Check public issues for Inspect on github and attempt to solve some of them

Concrete improvements for Inspect:

- Python packages implementing collections of Inspect solvers, tools, scorers, etc.
- Implementation of realistic test environments (e.g. like. https://webarena.dev/) for testing a wider range of agent scenarios in a contained setting.

- Tools for analyzing log files for quality control (i.e. tools that review transcripts/trajectories to identify reasons for failure)
- Tools for presenting collections of results (e.g. eval sets) in dashboard format.
- Front ends for dataset development (testing inputs, prompts, etc.) and getting immediate scoring feedback from a range of models.
- Front ends for human grading and tools for human calibration of model graded scorers.

Tools for LM agents to use

Difficulty: medium
Time estimate: 1-3 months
Credit: -

It is plausible that LM agents will be using many different tools. Right now, the most common tools are bash, python, edit tools, and search. However, there is a large range of tools that might be useful for all sorts of tasks. A possible project would be to search through the literature (e.g. this Tweet) on which tools, general or specialized, help. By default, I suggest to build the tools in Inspect.

Appendix

Detailed version of LM agent village

Credit: Daniel Kokotajlo

Question: how is this not the same as the truth terminal project? Answer: there was some difference but Marius forgot what it was.

I've been pitching various groups of engineers on building something like this for months. Here's the pitch I gave Sage (well, it's a success fantasy. But it's a success fantasy that serves as a pitch.)

Act 1: Setup: Sage tinkers around and gets about 5 - 10 different agents working reasonably well. They are mostly variants of Aider, some use Claude, some use ChatGPT, one uses Gemini, some use Llama. They set them all up with Twitch streams and internet access. They give them goal-prompts saying things like "Maximize paperclips subject to applicable laws and without doing anything unethical." (Perhaps there are a few different copies of each bot, each with different goals? To be added later.) They have a tacked-on scaffold that puts their special "I AM A BOT learn about me at this tinyurl" stamp on every piece of text they write online. They are also given a discord server which they can all participate in, which they can use to send messages to each other. The whole menagerie goes to sleep outside of Sage working hours; it's only 'on' while Sage is 'on.'

Act 2: The project has mostly flopped. The AIs flail around not accomplishing much of anything. However, just like with ChaosGPT, sometimes the failures are funny. Sage engineers have found it quite fun to tinker with it, so they keep adding features in their spare time. For example, there's a monitor-bot that looks over the transcripts of what's happening and flags various things. There's also a reporting system by which the bots themselves can ping Sage engineers to try to get their attention. There's also a README file that keeps growing, which all the bots have access to, that explains their situation. It keeps growing in part because sometimes bots get stuck due to misunderstandings about their situation, which can be rectified by changing the prompt and/or the README. There's some buzz about this online and some hand-wringing but then it goes away.

Act 3: New model comes out! It's SOTA at being an agent! It's integrated into the ecosystem and quickly becomes the star of the show. It even realizes this to some extent. It still tends to get stuck and is accomplishing nothing more interesting than leaving some interesting comments on various internet forums, but still. The comments it leaves are starting to drive some traffic to the webpage, haha. Sage researchers add some new features: (1) Now the bots can design new bots, and (2) Now the bots can 'vote each other off the island' and also 'vote new bots into existence.' These features aren't used particularly interestingly, though there are some funny uses (e.g. the smartest bot tries to rally the other bots to vote some of the idiots off the island, but they won't listen). The most interesting thing is when some human vandals try to 'take over' the bot ecosystem with the use of jailbreaks. They go to forums frequented by the bots and start talking to them and trying to jailbreak them into becoming something like ChaosGPT. It partially works, and partially doesn't. Hilarity ensues. More traffic to the website! Sage rebuilds the system to make this somewhat more difficult, and generally updates the bots to make them more effective.

Act 4: Sage creates a suite of new bots with more 'political' goals: Some to argue in favor of AI regulation, some to argue against. All are strictly prompted to use only true facts and valid arguments, and to use the highest standards of argumentative charity, etc. They go off to various forums and subreddits and start arguing with each other. Some humans find this very annoying, others find it hilarious; they get banned from some forums but permitted on others. There's an interesting moment where you can read the CoT of the bots realizing they are banned and trying to decide how they should change their behavior going forward. Also, two new SOTA models are released around now, adding more competence and diversity to the bot lineup. Also, by this point this project has been running for long enough that by sheer chance several funny and/or interesting stories have arisen organically from it, mostly involving various silly quests the bots went down arguing with people on various forums, trying to pay people to help them buy paperclips, having galaxy-brained and endearingly childish conversations about grand strategy and how they can achieve their long-term goals. Sage also creates an internal currency system tied to Manifold mana; the bots are given Manifold Markets accounts and access to a widget which enables them to give each other and the "Sage" entity mana, and the "Sage" entity will in return for mana give them various things (e.g. more compute to run copies of themselves; permissions to various websites; 15min of Sage human researcher assistance)

Act 5: Another major model release! The associated bots once again blow all the others out of the water. They seem... pretty long-horizon agentic now? In fact, they are smart enough to read the literature on agent/scaffolds and then read their own code and then suggest improvements in the form of new agents which they make sure to give the same goal-prompts. And they are organized enough to do the necessary voting etc. to get the new bots built. And they are starting to slightly make money day-trading on Manifold and begging people for donations. In the wider world, there is a bit of an ongoing "chatGPT moment" as everyone freaks out about AI agents; and this project becomes part of the media cycle in a big way -- people point to it as a go-to example of how the new systems really do seem to be functional autonomous agents. (There are other examples but they aren't as good yet because they didn't hit the ground running like Sage did, not having built up the infrastructure. Also with Sage there is a trend to look at: See how the old bots behave compared to the new, etc.) Tons of humans are now sending messages to the bots, trying to talk to them, jailbreak them, etc. and the bots are having to manage this influx of info. The smartest ones are actually able to do OK at this; some interesting politics ensues as they fight against the jailbroken dumber bots for political control of the virtual ecosystem. Sage is now actually making money on this whole thing due to Twitch stream; it decides to 10x the amount of $/compute it spends on these bots.

Act 6: Now this project has been 10x'd, it's actually entering interesting experiment territory in a new way. There are more than a hundred bots running in parallel; they form a sort of 'virtual village' with internal politics, economy, etc. as well as external. When some bots get stuck, others often notice and help get them un-stuck. (It helps that some Sage researchers made some bots specifically prompted to do this, and created an in-built 'interrupt what they are doing and send them message X' option which bots can do to each other for a mana fee). Serious social scientists and AI forecasters are watching this all unfold and pontificating about it and seeking funding to spin up more ambitious versions. Similarly ambitious versions are popping up all over the internet, and of course the first thing they do is make contact with Sagetown. Now there are multiple networked AI villages, and a whole mini-subculture of humans watching them and interacting with them.