
(a)
(b)
(c)
(d)
(e)

"Why Not Just..."

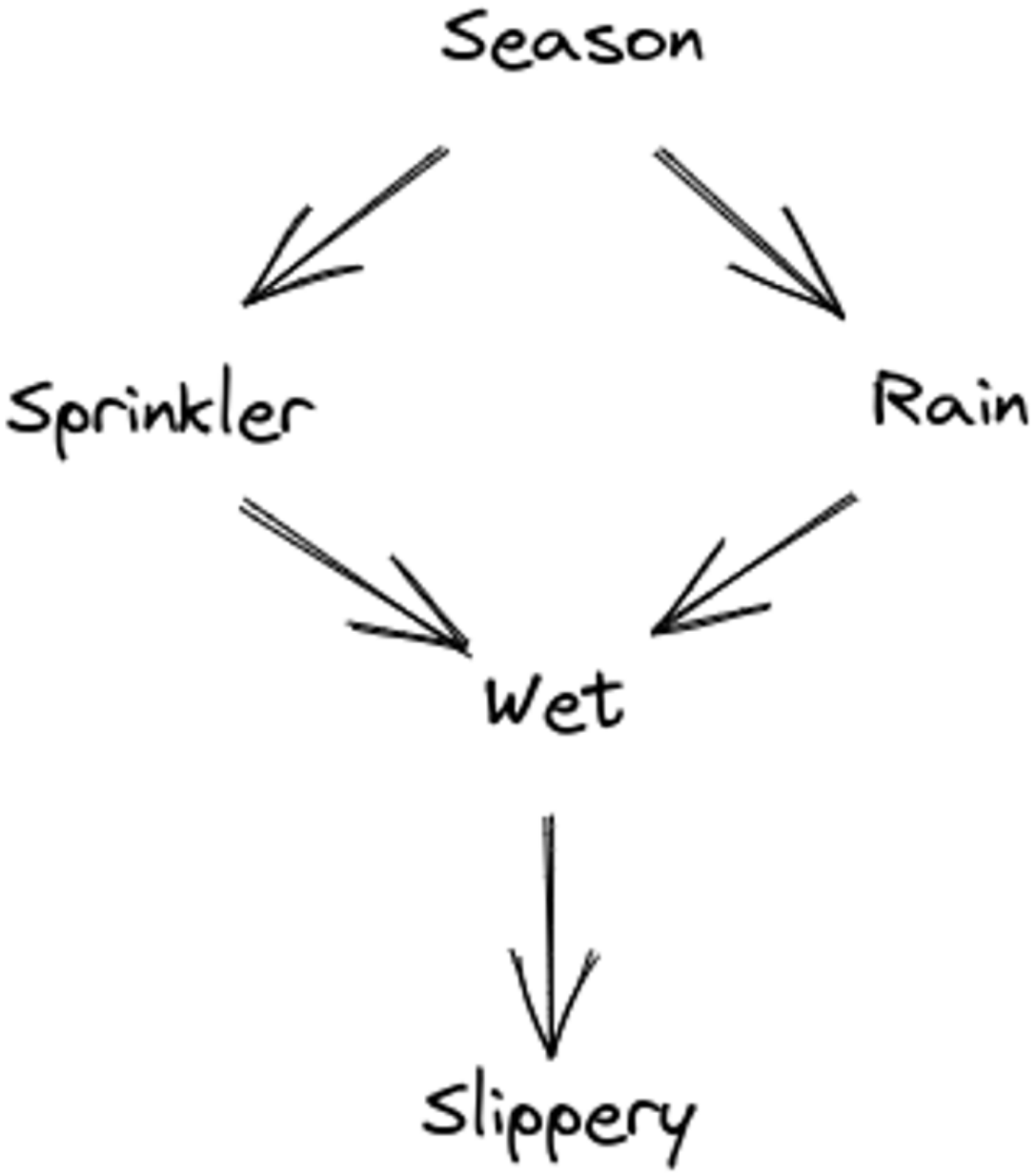
1. Deep Learning Systems Are Not Less Interpretable Than Logic/Probability/Etc
2. Godzilla Strategies
3. Rant on Problem Factorization for Alignment
4. Interpretability/Tool-ness/Alignment/Corrigibility are not Composable
5. How To Go From Interpretability To Alignment: Just Retarget The Search
6. Oversight Misses 100% of Thoughts The AI Does Not Think
7. Human Mimicry Mainly Works When We're Already Close
8. Worlds Where Iterative Design Fails

Deep Learning Systems Are Not Less Interpretable Than Logic/Probability/Etc

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

There's a common perception that various non-deep-learning ML paradigms - like logic, probability, causality, etc - are very interpretable, whereas neural nets aren't. I claim this is wrong.

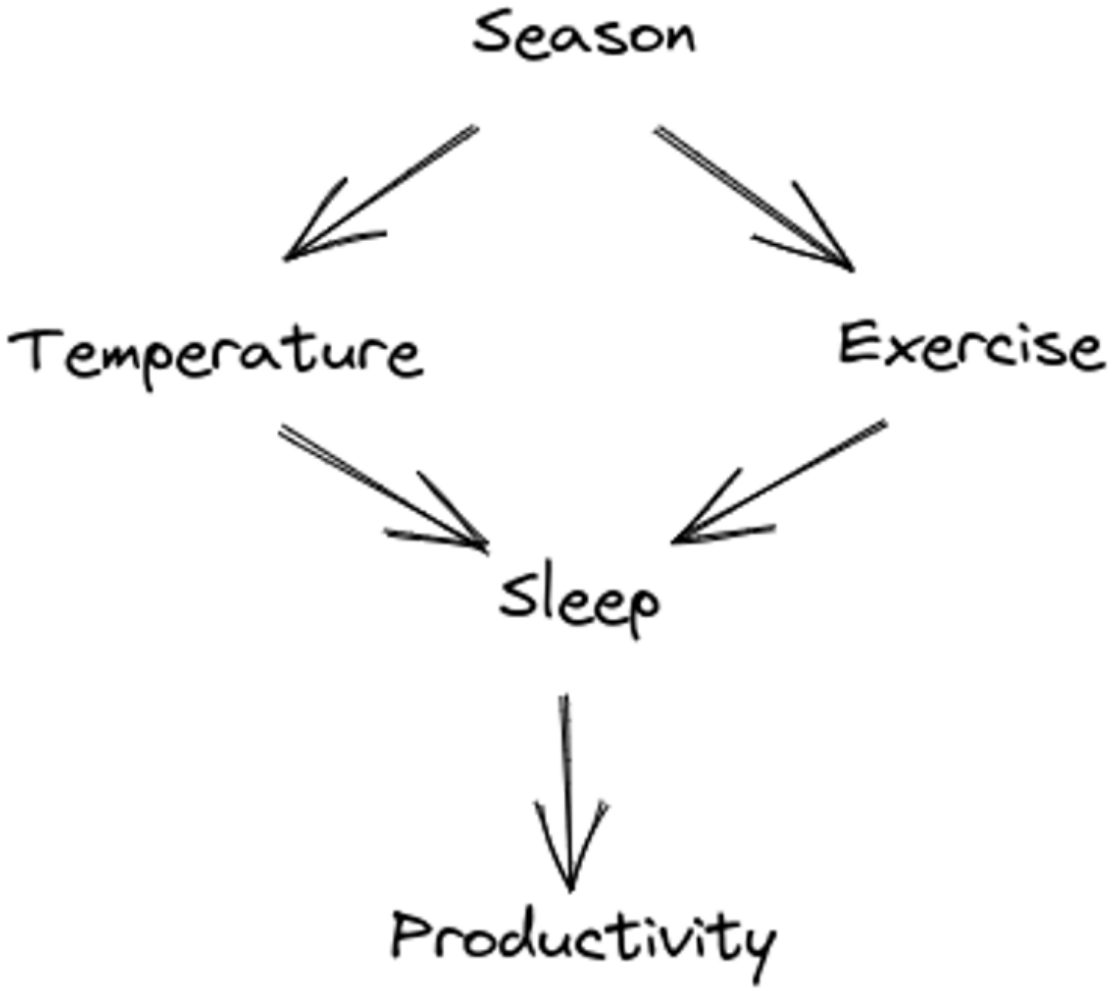
It's easy to see where the idea comes from. Look at the sort of models in, say, Judea Pearl's work. Like this:



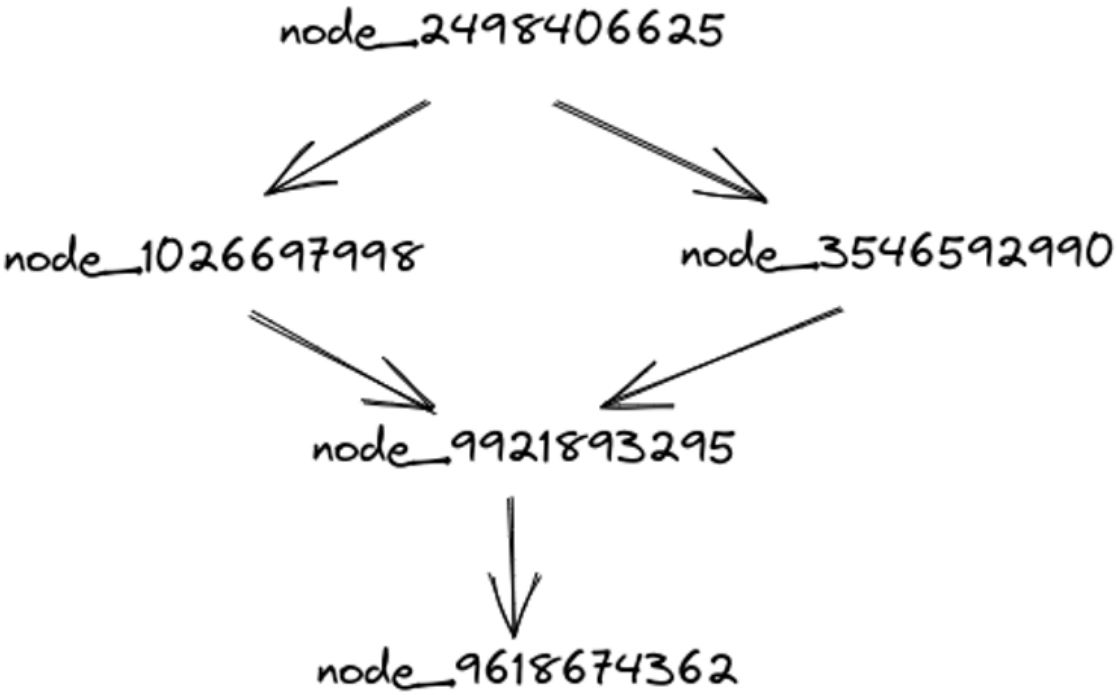
It says that either the sprinkler or the rain could cause a wet sidewalk, season is upstream of both of those (e.g. more rain in spring, more sprinkler use in summer), and sidewalk slipperiness is caused by wetness. The Pearl-style framework lets us do all sorts of probabilistic and causal reasoning on this system, and it all lines up quite neatly with our intuitions. It *looks* very interpretable.

The problem, I claim, is that a whole bunch of work is being done by the *labels*. “Season”, “sprinkler”, “rain”, etc. The *math* does not depend on those labels at all. If we *code* an ML system to use this sort of model, its behavior will also not depend on the labels at all. They’re just suggestively-named LISP tokens. We could use the exact same math/code to model some entirely different system, like my sleep quality being caused by room

temperature and exercise, with both of those downstream of season, and my productivity the next day downstream of sleep.



We could just replace all the labels with random strings, and the model would have the same content :



Now it looks a lot less interpretable.

Perhaps that seems like an unfair criticism? Like, the causal model is doing some nontrivial work, but connecting the labels to real-world objects just isn't the problem it solves?

… I think that's true, actually. But connecting the internal symbols/quantities/data structures of a model to external stuff is (I claim) exactly what interpretability is all about.

Think about interpretability for deep learning systems. A prototypical example for what successful interpretability might look like is e.g. we find a neuron which robustly lights up specifically in response to trees. It's a tree-detector! That's highly interpretable: we know what that neuron "means", what it corresponds to in the world. (Of course in practice single neurons are probably not the thing to look at, and also the word "robustly" is doing a lot of subtle work, but those points are not really relevant to this post.)

The corresponding problem for a logic/probability/causality-based model would be: take a variable or node, and figure out what thing in the world it corresponds to, ignoring the not-actually-functionally-relevant label. Take the whole system, remove the labels, and try to rederive their meanings.

… which sounds basically-identical to the corresponding problem for deep learning systems.

We are no more able to solve that problem for logic/probability/causality systems than we are for deep learning systems. We can have a node in our model labeled "tree", but we are no more (or less) able to check that it *actually robustly represents trees* than we are for a given neuron in a neural network. Similarly, if we find that it does represent trees and we want to understand how/why the tree-representation works, all those labels are a distraction.

One could argue that we're lucky deep learning is winning the capabilities race. At least this way it's *obvious* that our systems are uninterpretable, that we have no idea what's going on inside the black box, rather than our brains seeing the decorative natural-language name "sprinkler" on a variable/node and then thinking that we know what the variable/node

means. Instead, we just have unlabeled nodes - an accurate representation of our actual knowledge of the node’s “meaning”.

Godzilla Strategies

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

> Clutching a bottle of whiskey in one hand and a shotgun in the other, John scoured the research literature for ideas... He discovered several papers that described software-assisted hardware recovery. The basic idea was simple: if hardware suffers more transient failures as it gets smaller, why not allow software to detect erroneous computations and re-execute them? This idea seemed promising until John realized THAT IT WAS THE WORST IDEA EVER. Modern software barely works when the hardware is correct, so relying on software to correct hardware errors is like asking Godzilla to prevent Mega-Godzilla from terrorizing Japan. THIS DOES NOT LEAD TO RISING PROPERTY VALUES IN TOKYO. It's better to stop scaling your transistors and avoid playing with monsters in the first place, instead of devising an elaborate series of monster checks-and-balances and then hoping that the monsters don't do what monsters are always going to do because if they didn't do those things, they'd be called dandelions or puppy hugs.
>
> - James Mickens, The Slow Winter

There's a lot of AI alignment strategies which can reasonably be described as "ask Godzilla to prevent Mega-Godzilla from terrorizing Japan". Use one AI to oversee another AI. Have two AIs debate each other. Use one maybe-somewhat-aligned AI to help design another. Etc.

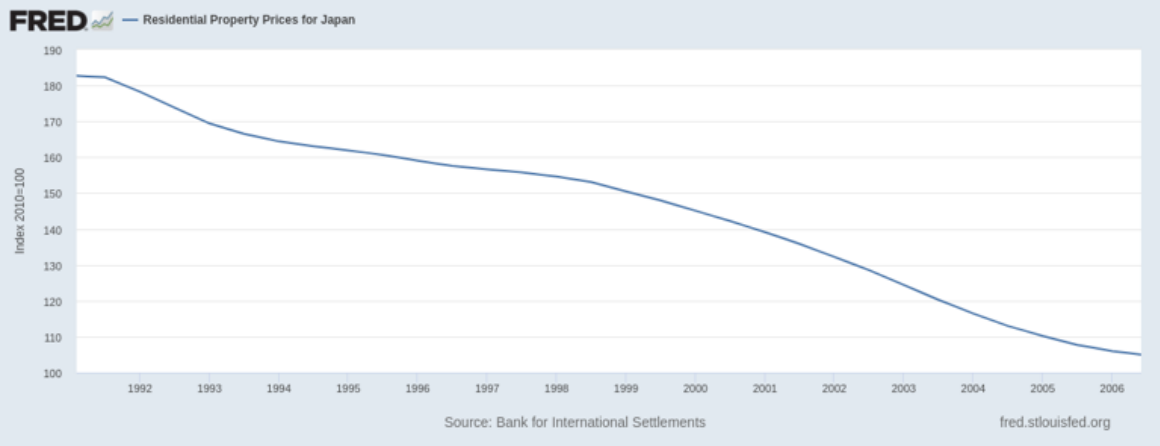
Alignment researchers discuss various failure modes of asking Godzilla to prevent Mega-Godzilla from terrorizing Japan. Maybe one of the two ends up much more powerful than the other. Maybe the two make an acausal agreement. Maybe the Nash Equilibrium between Godzilla and Mega-Godzilla just isn't very good for humans in the first place. Etc. These failure modes are useful for guiding technical research.

… but I worry that talking about the known failure modes misleads people about the strategic viability of Godzilla strategies. It makes people think (whether consciously/intentionally or not) "well, if we could handle these particular failure modes, maybe asking Godzilla to prevent Mega-Godzilla from terrorizing Japan would work".

What I like about the Godzilla analogy is that it gives a strategic intuition which much better matches the real world. When someone claims that their elaborate clever scheme will allow us to safely summon Godzilla in order to fight Mega-Godzilla, the intuitively-obviously-correct response is "THIS DOES NOT LEAD TO RISING PROPERTY VALUES IN TOKYO".

"But look!" says the clever researcher, "My clever scheme handles problems X, Y and Z!"

Response:



Oops

“Ok, but what if we had a really good implementation?” asks the clever researcher.

Response:

RAAARRRRRRR!

“Oh come on!” says the clever researcher, “You’re not even taking this seriously! At least say something about *how* it would fail.”

Don’t worry, we’re going to get to that. But before we do: let’s imagine you’re the Mayor of Tokyo evaluating a proposal to ask Godzilla to fight Mega-Godzilla. Your clever researchers have given you a whole lengthy explanation about how their elaborate and clever safeguards will ensure that this plan does not destroy Tokyo. You are unable to think of any potential problems which they did not address. Should you conclude that asking Godzilla to fight Mega-Godzilla will not result in Tokyo’s destruction?

No. Obviously not. THIS DOES NOT LEAD TO RISING PROPERTY VALUES IN TOKYO. You may not be able to articulate *why* the answer is obviously “no”, but asking Godzilla to fight Mega-Godzilla will still obviously destroy Tokyo, and your intuitions are right about that even if you are unable to articulate clever arguments.

With that said, let’s talk about why those intuitions are right and why the Godzilla analogy works well.

Brittle Plans and Unknown Unknowns

The basic problem with Godzilla plans is that they’re *brittle*. The moment anything goes wrong, the plan shatters, and then you’ve got somewhere between one and two giant monsters rampaging around downtown.

And of course, it is a fundamental Law of the universe that nothing ever goes exactly according to plan. Especially when trying to pit two giant monsters against each other. This is the sort of situation where there *will definitely* be unknown unknowns.

Unknown unknowns + brittle plan = definitely not rising property values in Tokyo.

Do we know what specifically will go wrong? No. Will something go wrong? Very confident yes. And brittleness means that whatever goes wrong, goes very wrong. Errors are not recoverable, when asking Godzilla to fight Mega-Godzilla.

If we use one AI to oversee another AI, and something goes wrong, that’s not a recoverable error; we’re using AI assistance in the first place because we can’t notice the relevant problems without it. If two AIs debate each other in hopes of generating a good plan for a human, and something goes wrong, that’s not a recoverable error; it’s the AIs themselves which we depend on to notice problems. If we use one maybe-somewhat-aligned AI to build another, and something goes wrong, that’s not a recoverable error ; if we had better ways to detect misalignment in the child we’d already have used them on the parent.

The real world will always throw some unexpected problems at our plans. When asking Godzilla to fight Mega-Godzilla, those problems are not recoverable. THIS DOES NOT LEAD TO RISING PROPERTY VALUES IN TOKYO.

Meta note: I expect this post to have a lively comment section! Before you leave the twentieth comment saying that maybe Godzilla fighting Mega-Godzilla is better than Mega-Godzilla rampaging unchallenged, maybe check whether somebody else has already written that one, so I don't need to write the same response twenty times. (But definitely do leave that comment if you're the first one, I intentionally kept this essay short on the assumption that lots of discussion would be in the comments.)

Rant on Problem Factorization for Alignment

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

This post is the second in what is likely to become a series of uncharitable rants about alignment proposals (previously: Godzilla Strategies). In general, these posts are intended to convey my underlying intuitions. They are *not* intended to convey my all-things-considered, reflectively-endorsed opinions. In particular, my all-things-considered reflectively-endorsed opinions are usually more kind. But I think it is valuable to make the underlying, not-particularly-kind intuitions publicly-visible, so people can debate underlying generators directly. I apologize in advance to all the people I insult in the process.

With that in mind, let's talk about problem factorization (a.k.a. task decomposition).

HCH

It all started with HCH, a.k.a. The Infinite Bureaucracy.

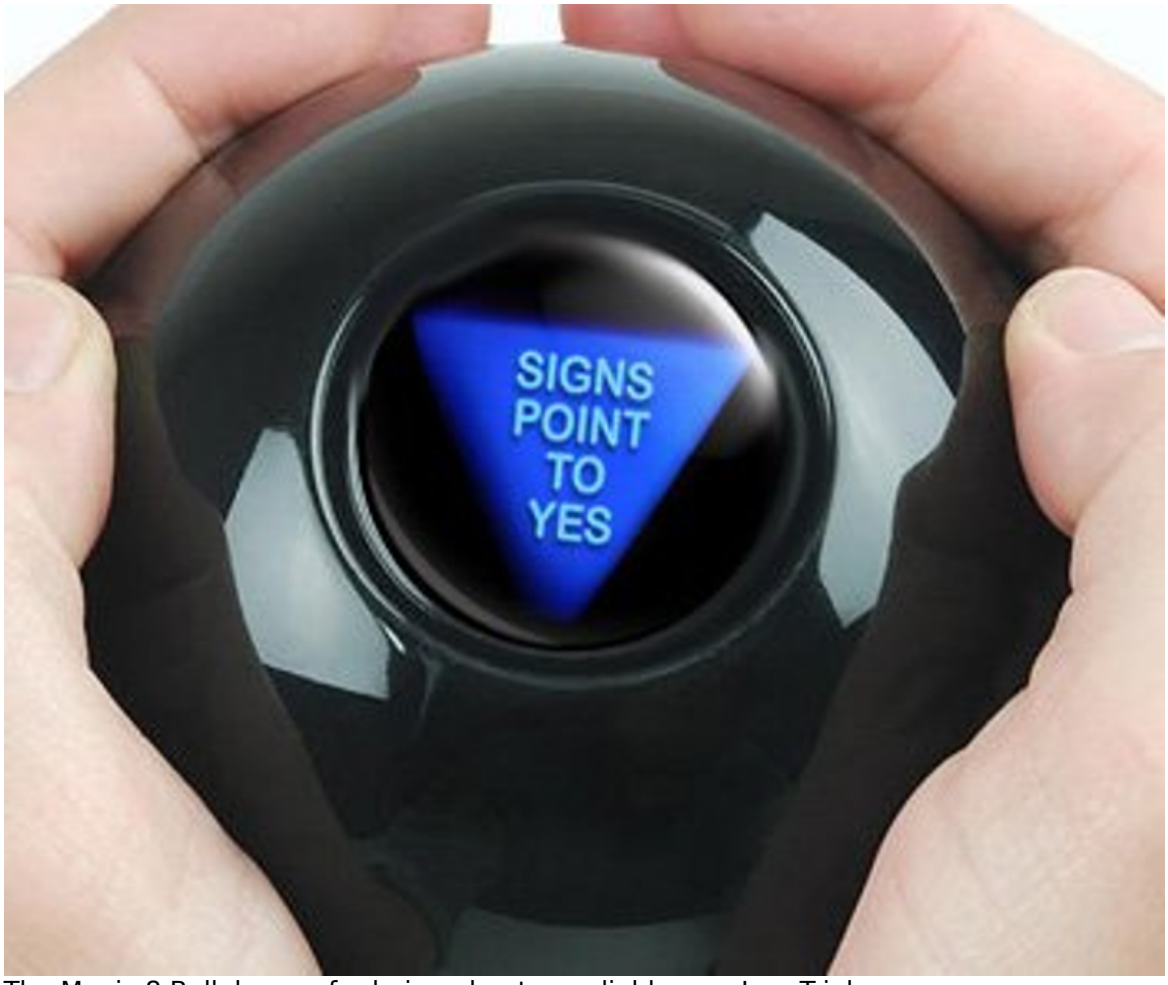
The idea of The Infinite Bureaucracy is that a human (or, in practice, human-mimicking AI) is given a problem. They only have a small amount of time to think about it and research it, but they can delegate subproblems to their underlings. The underlings likewise each have only a small amount of time, but can further delegate to *their* underlings, and so on down the infinite tree. So long as the humans near the top of the tree can "factorize the problem" into small, manageable pieces, the underlings should be able to get it done. (In practice, this would be implemented by training a question-answerer AI which can pass subquestions to copies of itself.)

At this point the ghost of George Orwell chimes in, not to say anything in particular, but just to scream. The ghost has a point: how on earth does an infinite bureaucracy seem like anything besides a terrible idea?

"Well," says a proponent of the Infinite Bureaucracy, "unlike in a *real* bureaucracy, all the humans in the infinite bureaucracy are actually just trying to help you, rather than e.g. engaging in departmental politics." So, ok, apparently this person has not met a lot of real-life bureaucrats. The large majority are decent people who are honestly trying to help. It is true that departmental politics are a big issue in bureaucracies, but those selection pressures apply regardless of the peoples' intentions. And also, man, it sure does seem like Coordination is a Scarce Resource and Interfaces are a Scarce Resource and scarcity of those sorts of things sure would make bureaucracies incompetent in basically the ways bureacracies are incompetent in practice.

Debate and Other Successors

So, ok, maybe The Infinite Bureaucracy is not the right human institution to mimic. What institution can use humans to produce accurate and sensible answers to questions, robustly and reliably? Oh, I know! How about the Extremely Long Jury Trial? Y'know, because juries are, in practice, known for their extremely high reliability in producing accurate and sensible judgements!



The Magic 8 Ball, known for being about as reliable as a Jury Trial.

“Well,” says the imaginary proponent, “unlike in a *real* Jury Trial, in the Extremely Long Jury Trial, the lawyers are both superintelligent and the arguments are so long that no human could ever possibility check them all the way through; the lawyers instead read each other’s arguments and then try to point the Jury at the particular places where the holes are in the opponent’s argument without going through the whole thing end-to-end.”

I rest my case.

Anyway, HCH and debate have since been followed by various other successors, which improve on their predecessors mostly by adding more boxes and arrows and loops and sometimes even *multiple colors* of arrows to the diagram describing the setup. Presumably the strategy is to make it complicated enough that it no longer obviously corresponds to some strategy which already fails in practice, and then we can bury our heads in the sand and pretend that We Just Don’t Know whether it will work and therefore maybe it will work.

(Reminder: in general I don’t reflectively endorse everything in this post; it’s accurately conveying my underlying intuitions, not my all-things-considered judgement. That last bit in particular was probably overly harsh.)

The Ought Experiment

I have a hypothesis about problem factorization research. My guess is that, to kids fresh out of the ivory tower with minimal work experience at actual companies, it seems totally plausible that humans can factorize problems well. After all, we manufacture all sorts of things on production lines, right? Ask someone who's worked in a non-academia cognitive job for a while (like e.g. a tech company), at a company with more than a dozen people, and they'll be like "lolwut obviously humans don't factorize problems well, have you ever seen an actual company?". I'd love to test this theory, please give feedback in the comments about your own work experience and thoughts on problem factorization.

Anyway, for someone either totally ignorant of the giant onslaught of evidence provided by day-to-day economic reality, or trying to ignore the giant onslaught of evidence in order to avoid their hopes being crushed, it apparently seems like We Just Don't Know whether humans can factorize cognitive problems well. ~~Sort of like We Just Don't Know whether a covid test works until after the FDA finishes its trials, even after the test has been approved in the EU~~ ok that's a little too harsh even for this post.

So Ought went out and tested it experimentally. (Which, sarcasm aside, was a great thing to do.)

The experiment setup: a group of people are given a Project Euler problem. The first person receives the problem, has five minutes to work on it, and records their thoughts in a google doc. The doc is then passed to the next person, who works on it for five minutes recording their thoughts in the doc, and so on down the line. (Note: I'm not sure it was 5 minutes exactly, but something like that.) As long as the humans are able to factor the problem into 5-minute-size chunks without too much overhead, they should be able to efficiently solve it this way.

So what actually happened?

The story I got from a participant is: it sucked. The google doc was mostly useless, you'd spend five minutes just trying to catch up and summarize, people constantly repeated work, and progress was mostly not cumulative. Then, eventually, one person would just ignore the google doc and manage to solve the whole problem in five minutes. (This was, supposedly, usually the same person.) So, in short, the humans utterly failed to factor the problems well, exactly as one would (very strongly) expect from seeing real-world companies in action.

This story basically matches the official write-up of the results.

~~So Ought said "Oops" and moved on to greener pastures~~ lol no, last I heard Ought is still trying to figure out if better interface design and some ML integration can make problem factorization work. Which, to their credit, would be insanely valuable if they could do it.

That said, I originally heard about HCH and the then-upcoming Ought experiment from Paul Christiano in the summer of 2019. It was immediately very obvious to me that HCH was hopeless (for basically the reasons discussed here); at the time I asked Paul "So when the Ought experiments inevitably fail completely, what's the fallback plan?". And he basically said "back to more foundational research". And to Paul's credit, three years and an Ought experiment later, he's now basically moved on to more foundational research.

Sandwiching

About a year ago, Cotra proposed a different class of problem factorization experiments: "sandwiching". We start with some ML model which has lots of knowledge from many different fields, like GPT-n. We also have a human who has a domain-specific problem to solve (like e.g. a coding problem, or a translation to another language) but lacks the relevant domain knowledge (e.g. coding skills, or language fluency). The problem, roughly speaking, is to get the ML model and the human to work as a team, and produce an outcome at-least-

as-good as a human expert in the domain. In other words, we want to factorize the “expert knowledge” and the “having a use-case” parts of the problem.

(The actual sandwiching experiment proposal adds some pieces which I claim aren’t particularly relevant to the point here.)

I love this as an experiment idea. It really nicely captures the core kind of factorization needed for factorization-based alignment to work. But Cotra makes one claim I don’t buy: that We Just Don’t Know how such experiments will turn out, or how hard sandwiching will be for cognitive problems in general. I claim that the results are very predictable, because things very much like this already happen all the time in practice.

For instance: consider a lawyer and a business owner putting together a contract. The business owner has a rough intuitive idea of what they want, but lacks expertise on contracts/law. The lawyer has lots of knowledge about contracts/law, but doesn't know what the business owner wants. The business owner is like our non-expert humans; the lawyer is like GPT.

In this analogy, the analogue of an expert human would be a business owner who is also an expert in contracts/law. The analogue of the "sandwich problem" would be to get the lawyer + non-expert business-owner to come up with a contract as good as the expert business-owner would. This sort of problem has been around for centuries, and I don't think we have a good solution in practice; I'd expect the expert business-owner to usually come up with a much better contract.

This sort of problem comes up all the time in real-world businesses. We could just as easily consider a product designer at a tech startup (who knows what they want but little about coding), an engineer (who knows lots about coding but doesn't understand what the designer wants), versus a product designer who's also a fluent coder and familiar with the code base. I've experienced this one first-hand; the expert product designer is way better. Or, consider a well-intentioned mortgage salesman, who wants to get their customer the best mortgage for them, and the customer who understands the specifics of their own life but knows nothing about mortgages. Will they end up with as good a mortgage as a customer who has expertise in mortgages themselves? Probably not. (I've seen this one first-hand too.)

There’s tons of real-life sandwiching problems, and tons of economic incentive to solve them, yet we do not have good general-purpose solutions.

The Next Generation

Back in 2019, I heard Paul’s HCH proposal, heard about the Ought experiment, and concluded that this bad idea was already on track to self-correct via experimental feedback. Those are the best kind of bad ideas. I wrote up some of the relevant underlying principles (Coordination as a Scarce Resource and Interfaces as a Scarce Resource), but mostly waited for the problem to solve itself. And I think that mostly worked… for Paul.

But meanwhile, over the past year or so, the field has seen a massive influx of bright-eyed new alignment researchers fresh out of college/grad school, with minimal work experience in industry. And of course most of them don’t read through most of the enormous, undistilled, and very poorly indexed corpus of failed attempts from the past ten years. (And it probably doesn’t help that a plurality come through the AGI Safety Fundamentals course, which last time I checked had a whole section on problem factorization but, to my knowledge, didn’t even mention the Ought experiment or the massive pile of close real-world economic analogues. It does include two papers which got ok results by picking easy-to-decompose tasks and hard-coding the decompositions.) So we have a perfect recipe for people who will see problem factorization and think “oh, hey, that could maybe work!”.

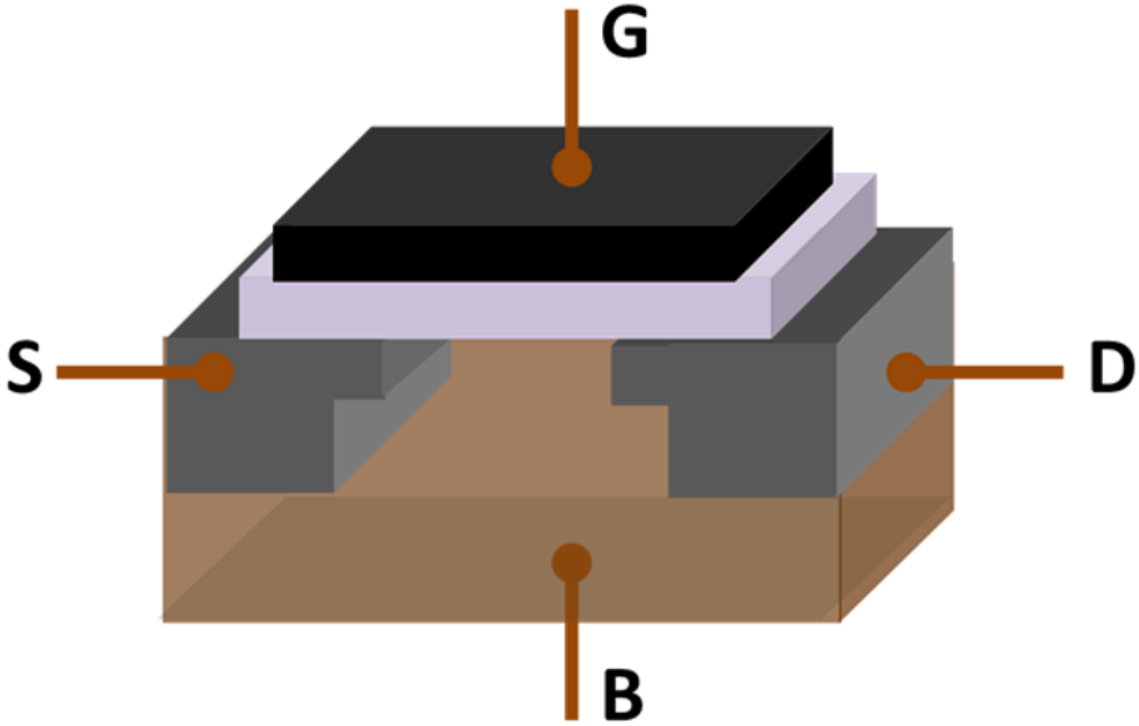
If we’re lucky, hopefully some of the onslaught of bright-eyed new researchers will attempt their own experiments (like e.g. sandwiching) and manage to self-correct, but at this point new researchers are pouring in faster than any experiments are likely to proceed, so probably the number of people pursuing this particular dead end will go up over time.

Interpretability/Tool-ness/Alignment/Corrigibility are not Composable

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Interpretability

I have a decent understanding of how transistors work, at least for purposes of basic digital circuitry. Apply high voltage to the gate, and current can flow between source and drain. Apply low voltage, and current doesn't flow. (... And sometimes reverse that depending on which type of transistor we're using.)



> Transistor visual from wikipedia showing Source, Drain, Gate, and (usually ignored unless we're really getting into the nitty gritty) Body. At a conceptual level: when voltage is applied to the Gate, the charge on the gate attracts electrons (or holes) up into the gap between Source and Drain, and those electrons (or holes) then conduct current between Source and Drain.

I also understand how to wire transistors together into a processor and memory. I understand how to write machine and assembly code to run on that processor, and how to write a compiler for a higher-level language like e.g. python. And I understand how to code up, train and run a neural network from scratch in python.

In short, I understand all the pieces from which a neural network is built at a low level, and I understand how all those pieces connect together. And yet, I do not really understand what's going on inside of trained neural networks.

This shows that interpretability is *not composable*: if I take a bunch of things which I know how to interpret, and wire them together in a way I understand, I do not necessarily know how to interpret the composite system. Composing interpretable pieces does not necessarily yield an interpretable system.

Tools

The same applies to "tools", in the sense of "tool AI". Transistors and wires are very tool-ish: I understand what they do, they're definitely not optimizing the broader world or trying to trick me or modelling me at all or trying to self-preserve or acting agenty in general. They're just simple electronic tools.

And yet, assuming agenty AI is possible at all, it will be possible to assemble those tools into something agenty.

So, like interpretability, tool-ness is not composable: if I take a bunch of non-agenty tools, and wire them together in a way I understand, the composite system is not necessarily a non-agenty tool. Composing non-agenty tools does not necessarily yield a non-agenty tool.

Alignment/Corrigibility

What if I take a bunch of aligned and/or corrigible agents, and "wire them together" into a multi-agent organization? Is the resulting organization aligned/corrigible?

Actually there's a decent argument that it is, *if* the individual agents are sufficiently highly capable. If the agents can model each other well enough and coordinate well enough, then they should be able to each individually predict what individual actions will cause the composite system to behave in an aligned/corrigible way, and they *want* to be aligned/corrigible, so they'll do that.

However, this does not work if the individual agents are very limited and unable to model the whole big-picture system. HCH-like proposals are a good example here: humans are not typically able to model the whole big picture of a large human organization. There are too many specialized skillsets, too much local knowledge and information, too many places where complicated things happen which the spreadsheets and managerial dashboards don't represent well. And humans certainly can't coordinate at scale very well in general - our large-scale communication bandwidth is maybe five to seven words at best. Each individual human may be reasonably aligned/corrigible, but that doesn't mean they aggregate together into an aligned/corrigible system.

The same applies if e.g. we magically factor a problem and then have a low-capability overseeable agent handle each piece. I could definitely oversee a logic gate during the execution of a program, make sure that it did absolutely nothing fishy, but overseeing each individual logic gate would do approximately nothing at all to prevent the program from behaving maliciously.

How These Arguments Work In Practice

In practice, nobody proposes that AI built from transistors and wires will be interpretable/tool-like/aligned/corrigible because the transistors and wires are interpretable/tool-like/aligned/corrigible. But people do often propose breaking things into very small chunks, so that each chunk is interpretable/tool-like/aligned/corrigible. For instance, interpretability people will talk about hiring ten thousand interpretability researchers to each interpret one little circuit in a net. Or problem factorization people will

talk about breaking a problem into a large number of tiny little chunks each of which we can oversee.

And the issue is, the more little chunks we have to combine together, the more noncomposability becomes a problem. If we're trying to compose interpretability/tool-ness/alignment/corrigibility of *many* little things, then figuring out how to turn interpretability/tool-ness/alignment/corrigibility of the parts into interpretability/tool-ness/alignment/corrigibility of the whole is *the* central problem, and it's a hard (and interesting) open research problem.

How To Go From Interpretability To Alignment: Just Retarget The Search

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

When people talk about prosaic alignment proposals, there's a common pattern: they'll be outlining some overcomplicated scheme, and then they'll say "oh, and assume we have great interpretability tools, this whole thing just works way better the better the interpretability tools are", and then they'll go back to the overcomplicated scheme. (Credit to Evan for pointing out this pattern to me.) And then usually there's a whole discussion about the specific problems with the overcomplicated scheme.

In this post I want to argue from a different direction: if we had great interpretability tools, we could just use those to align an AI directly, and skip the overcomplicated schemes. I'll call the strategy "Just Retarget the Search".

We'll need to make two assumptions:

- Some version of the natural abstraction hypothesis holds, and the AI ends up with an internal concept for human values, or corrigibility, or what the user intends, or human mimicry, or some other outer alignment target.
- The standard mesa-optimization argument from Risks From Learned Optimization holds, and the system ends up developing a general-purpose (i.e. retargetable) internal search process.

Given these two assumptions, here's how to use interpretability tools to align the AI:

- Identify the AI's internal concept corresponding to whatever alignment target we want to use (e.g. values/corrigibility/user intention/human mimicry/etc).
- Identify the retargetable internal search process.
- Retarget (i.e. directly rewire/set the input state of) the internal search process on the internal representation of our alignment target.

Just retarget the search. Bada-bing, bada-boom.

Problems

Of course as written, "Just Retarget the Search" has some issues; we haven't added any of the bells and whistles to it yet. Probably the "identify the internal representation of the alignment target" step is less like searching through a bunch of internal concepts, and more like writing our intended target in the AI's internal concept-language. Probably we'll need to do the retargeting regularly on-the-fly as the system is training, even when the search is only partly-formed, so we don't end up with a misaligned AI before we get around to retargeting. Probably we'll need a bunch of empirical work to figure out which possible alignment targets are and are not easily expressible in the AI's internal language (e.g. I'd guess "user intention" or "human mimicry" are more likely than "human values"). But those details seem relatively straightforward.

A bigger issue is that “Just Retarget the Search” just… doesn’t seem robust enough that we’d want to try it on a superintelligence. We still need to somehow pick the right target (i.e. handle outer alignment), and ideally it’s a target which fails gracefully (i.e. some amount of basin-of-corrigibility). If we fuck up and aim a superintelligence at not-quite-the-right-target, game over. Insofar as “Just Retarget the Search” is a substitute for overcomplicated prosaic alignment schemes, that’s probably fine; most of those schemes are targeting only-moderately-intelligent systems anyway IIUC. On the other hand, we probably want our AI competent enough to handle ontology shifts well, otherwise our target may fall apart.

Then, of course, there’s the assumptions (natural abstractions and retargetable search), either of which could fail. That said, if one or both of the assumptions fail, then (a) that probably messes up a bunch of the overcomplicated prosaic alignment schemes too (e.g. failure of the natural abstraction hypothesis can easily sink interpretability altogether), and (b) that might mean that the system just isn’t that dangerous in the first place (e.g. if it turns out that retargetable internal search is indeed necessary for dangerous intelligence).

Upsides

First big upside of Just Retargeting the Search: it completely and totally eliminates the inner alignment problem. We just directly set the internal optimization target.

Second big upside of Just Retargeting the Search: it’s conceptually simple. The problems and failure modes are mostly pretty obvious. There is no recursion, no complicated diagram of boxes and arrows. We’re not playing two Mysterious Black Boxes against each other.

But the main reason to think about this approach, IMO, is that it’s a true *reduction of the problem*. Prosaic alignment proposals have a tendency to play a shell game with the Hard Part of the problem, move it around and hide it in different black boxes but never actually eliminate it. “Just Retarget the Search” directly eliminates the inner alignment problem. No shell game, no moving the Hard Part around. It still leaves the outer alignment problem unsolved, it still needs assumptions about natural abstractions and retargetable search, but it completely removes one Hard Part and reduces the problem to something simpler.

As such, I think “Just Retarget the Search” is a good baseline. It’s a starting point for thinking about the parts of the problem it doesn’t solve (e.g. outer alignment), or the ways it might fail (retargetable search, natural abstractions), without having to worry about inner alignment.

Oversight Misses 100% of Thoughts The AI Does Not Think

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Problem: an overseer won't see the AI which kills us all thinking about how to kill humans, not because the AI conceals that thought, but because the AI doesn't think about how to kill humans in the first place. The AI just kills humans as a side effect of whatever else it's doing.

Analogy: the Hawaii Chaff Flower didn't go extinct because humans strategized to kill it. It went extinct because humans were building stuff nearby, and *weren't* thinking about how to keep the flower alive. They probably weren't thinking about the flower much at all.

Hawaii Chaff Flower (source)

More generally: how and why do humans drive species to extinction? In some cases the species is hunted to extinction, either because it's a threat or because it's economically profitable to hunt. But I would guess that in 99+% of cases, the humans drive a species to extinction because the humans are doing something that changes the species' environment a lot, without specifically trying to keep the species alive. DDT, deforestation, introduction of new predators/competitors/parasites, construction... that's the sort of thing which I expect drives most extinction.

Assuming this metaphor carries over to AI (similar to the second species argument), what kind of extinction risk will AI pose?

Well, the extinction risk will not come from AI actively trying to kill the humans. The AI will just be doing some big thing which happens to involve changing the environment a lot (like making replicators, or dumping waste heat from computronium, or deciding that an oxygen-rich environment is just really inconvenient what with all the rusting and tarnishing and fires, or even just designing a fusion power generator), and then humans die as a side-effect. Collateral damage happens by default when something changes the environment in big ways.

What does this mean for oversight? Well, it means that there wouldn't necessarily be any point at which the AI is actually thinking about killing humans or whatever. It just doesn't think much about the humans at all, and then the humans get wrecked by side effects. In order for an overseer to raise an alarm, the overseer would have to figure out itself that the AI's plans will kill the humans, i.e. the overseer would have to itself predict the consequences of a presumably-very-complicated plan.

Human Mimicry Mainly Works When We're Already Close

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

What if we just simulate a bunch of alignment researchers, and have them solve the problem for us?

Of all the Dumb Alignment Ideas, this one is easily the best. Simple argument in favor: well, it's not going to do any *worse* than the researchers would have done. In other words, it will probably do at least as well as we would have done without it, and possibly better, insofar as it can run faster than realtime.

Another angle: human mimicry is a simple objective to train against, and is about as outer-aligned as the humans being mimicked. Which isn't necessarily perfect, but it's as aligned as our alignment researchers were going to be anyway (assuming inner alignment issues are handled, which we will indeed assume for the entirety of this post).

Those are pretty good arguments. But *man*, there are some subtle devils in the details.

Simulation vs Prediction

The ideal version of human mimicry is mind uploads: directly simulate our researchers in a stable, research-friendly environment for a long time.

The operationalization which people usually actually have in mind is to train an ML system to predict research outputs - e.g. I might prompt GPT for a johnswentworth post from the year 2050.

Even setting aside inner alignment issues, these two are radically different.

Generalization Problems

In order for GPT to generate a realistic johnswentworth post from the year 2050, it has to generalize *way* out of distribution.

… Well, ok, maybe I turn into one of those old researchers who just repeats the same things over and over again for decades, and then GPT doesn't need to generalize way out of distribution. But in that case it isn't very helpful to prompt for one of my posts from 2050 anyways, and we should prompt for something else instead (Thane Ruthenis has been writing great stuff lately, maybe try him?). The whole point of asking for future research write-ups is to see useful stuff we have not yet figured out; that means generalizing way out of the distribution of writing we already have.

But if the system generalizes *too* well out of distribution, then it correctly guesses that AGI will take over the world before 2050, and my attempt to prompt for a johnswentworth post from 2050 will instead return predicted writings from a

ridiculously superhuman future AGI pretending to be johnswentworth. And those writings presumably try to influence the reader in ways which bring about the AGI's takeover.

So in order to do useful work, our GPT-style system has to generalize out of distribution, but not *too* far out of distribution. We don't know how wide the window is between generalizing enough and generalizing too much, or if the window is wide enough to be useful at all.

One thing we can guess: prompting for research outputs in the very near future is probably much safer than prompting for dates further out. johnswentworth post from 2025 is a safer prompt than johnswentworth post from 2050. The less far out of distribution we go, the safer we are. Similarly, the more likely we are to solve the alignment problem and avoid AI takeover, the less likely it is that prompting GPT for future research outputs is dangerous, and the more likely it is to work.

The closer we are to solving alignment already, and the more likely we are to make it, the less dangerous it is to predict future research outputs. In other words: predicting future research outputs can only safely buy us a relatively small number of bits; we have to already be reasonably close to surviving in order for it to work.

So What's Different About Simulation?

Simulating researchers in a stable, research-friendly environment for a long time does not have the "predict outputs of a future AGI" problem. Why? What's the key difference?

The key is the "stable, research-friendly environment" part. Our simulated researchers are in a simulated environment where AGI is not going to take over. It's a *counterfactual* world very different from our own.

Alas, querying counterfactual worlds is fundamentally not a thing one can do simply by prompting GPT. Conceptually, prompts just do Bayesian conditioning on the modeled text distribution (i.e. condition the text on starting with the prompt); counterfactuals move us to an entirely different distribution. To generate a counterfactual query, we'd have to modify the system's internals somehow. And in fact, there has recently been some cool work which demonstrates decent performance on counterfactual queries by modifying GPT's internals! I don't think it's to the point where we could counterfact on something as complicated as "world in which AGI doesn't take over and our alignment researchers successfully solve the problem", and I don't think it's robust enough to put much weight on it yet, but the basic version of the capability does exist.

General Principle: Human Mimicry Buys A Limited Number Of Bits

Suppose GPT-12, with its vast training data and compute, internally concludes that humanity has a 1-in-32 chance of aligning/surviving AGI on our current trajectory. Then humanity would need 5 bits of optimization pressure in order to make it.

The more bits of optimization pressure humanity needs, the less likely human mimicry is to save us; we have to already be reasonably close to surviving in order for it to

work. We already talked about this principle in the context of accidentally prompting GPT to return writing from a future AGI, but the principle is much more general than that.

The Weird Shit Problem

Suppose we need 20 bits of optimization pressure (i.e. on our current trajectory we have only a ~1-in-a-million chance of avoiding AGI takeover). We train GPT, and counterfact on its internals to a world where AGI doesn't take over. But if our chances of avoiding takeover were that low (under GPT's model), then they're probably dominated by *weird shit*, things which have probabilities on the order of 1-in-a-million or less. Maybe we nuke ourselves to death or get hit by a big damn asteroid. Maybe aliens decide that humanity's AGI is about to become a problem to the rest of the galaxy and they need to take over rather than just letting us develop. Maybe time travel turns out to be a thing and weird time travel bullshit happens. Most likely it's something weird enough that I won't think of it.

Those weird futures vary in how safe they are to query (time travel would probably be on the very short list of things as dangerous as AGI), and in how likely they are to return anything useful at all (asteroid extinction tends to cut off blog post writing). But approximately zero of them involve our researchers just doing their research in a stable, research-friendly environment for a long time.

So when we need a lot of bits, it's not enough to just counterfact on a high-level thing like "AGI doesn't take over" and then let GPT pick the most probable interpretation of that world. We need pretty detailed, low-level counterfactuals.

Generalization Again

Another angle: number of bits of optimization required is a direct measure of "how far out of distribution" we need to generalize. Even setting aside actively dangerous queries, our simulator/predictor has to generalize out of distribution in order to return anything useful. In practice, the system will probably only be able to generalize so far, which limits how many bits of optimization we can get from it.

Expect More Problems

We've now been through a few different arguments all highlighting the idea that human mimicry can only buy so many bits of optimization (future AGI problem, weird shit problem, generalization). I expect the principle to be more general than these arguments. In other words, even if we patch the specific failure modes which these particular arguments talk about, trying to pump lots of bits of optimization out of human mimicry is still likely to be dangerous in ways we have not yet realized.

This is just an instance of the general principle that optimization becomes more dangerous as we crank up the optimization power - the very principle for why AGI is dangerous-by-default in the first place.

Takeaways

This post mostly talked about the limitations of human mimicry, but I want to emphasize: **this is the best of the Dumb Alignment Ideas to date**. If GPT-style models reach human or superhuman general intelligence next month, and we can't realistically delay its release, and you are the person sitting in front of the prompt wondering what to do, then prompting for future alignment research is absolutely what you should do. (And start not very far in the future, and *read the damn outputs* before going further, they'll hopefully contain additional warnings or new plans which you should do *instead* of prompting for more future research.) At that point it's not very likely to work, but we don't have a better immediate plan.

Good interpretability tools can buy us more bits in two ways:

- They potentially allow us to directly handle inner alignment issues by retargeting the search
- They potentially allow us to counterfact on the system's internal model, so we can predict researchers working in an environment less full of threats

Insofar as the tools are available, this is the thing to aim for if AGI is imminent.

... But the general principle is that human mimicry can buy only a limited number of bits. We definitely want to have the interpretability tools to implement the best version of human mimicry we can, but at the end of the day we'll mostly improve our chances by getting closer to solving the full alignment problem ourselves.

Worlds Where Iterative Design Fails

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

In most technical fields, we try designs, see what goes wrong, and iterate until it works. That's the core iterative design loop. Humans are good at iterative design, and it works well in most fields in practice.

In worlds where AI alignment can be handled by iterative design, we probably survive. So long as we can see the problems and iterate on them, we can probably fix them, or at least avoid making them worse.

By the same reasoning: worlds where AI kills us are generally worlds where, for one reason or another, the iterative design loop fails. So, if we want to reduce X-risk, we generally need to focus on worlds where the iterative design loop fails for some reason; in worlds where it doesn't fail, we probably don't die anyway.

Why might the iterative design loop fail? Most readers probably know of two widely-discussed reasons:

- Fast takeoff: there will be a sudden phase shift in capabilities, and the design of whatever system first undergoes that phase shift needs to be right on the first try.
- Deceptive inner misalignment: an inner agent behaves well in order to deceive us, so we can't tell there's a problem just by trying stuff and looking at the system's behavior.

… but these certainly aren't the only reasons the iterative design loop potentially fails. This post will mostly talk about some particularly simple and robust failure modes, but I'd encourage you to think on your own about others. These are the things which kill us; they're worth thinking about.

Basics: Hiding Problems

Example/Analogy: The Software Executive

Imagine that a software company executive, concerned about the many errors coming from the software, creates a new incentive scheme: software developers get a monetary reward for changes which decrease the rate of error messages showing up on the manager's dashboard, and get docked for changes which increase the rate of error messages.

As Tyler Cowen would say: "solve for the equilibrium". Obvious equilibrium here: the developers stop throwing error messages when they detect a problem, and instead the software just fails silently. The customer's experience remains the same, but the manager's dashboard shows fewer error messages. Over time, the customer's experience probably degrades, as more and more problems go undetected.

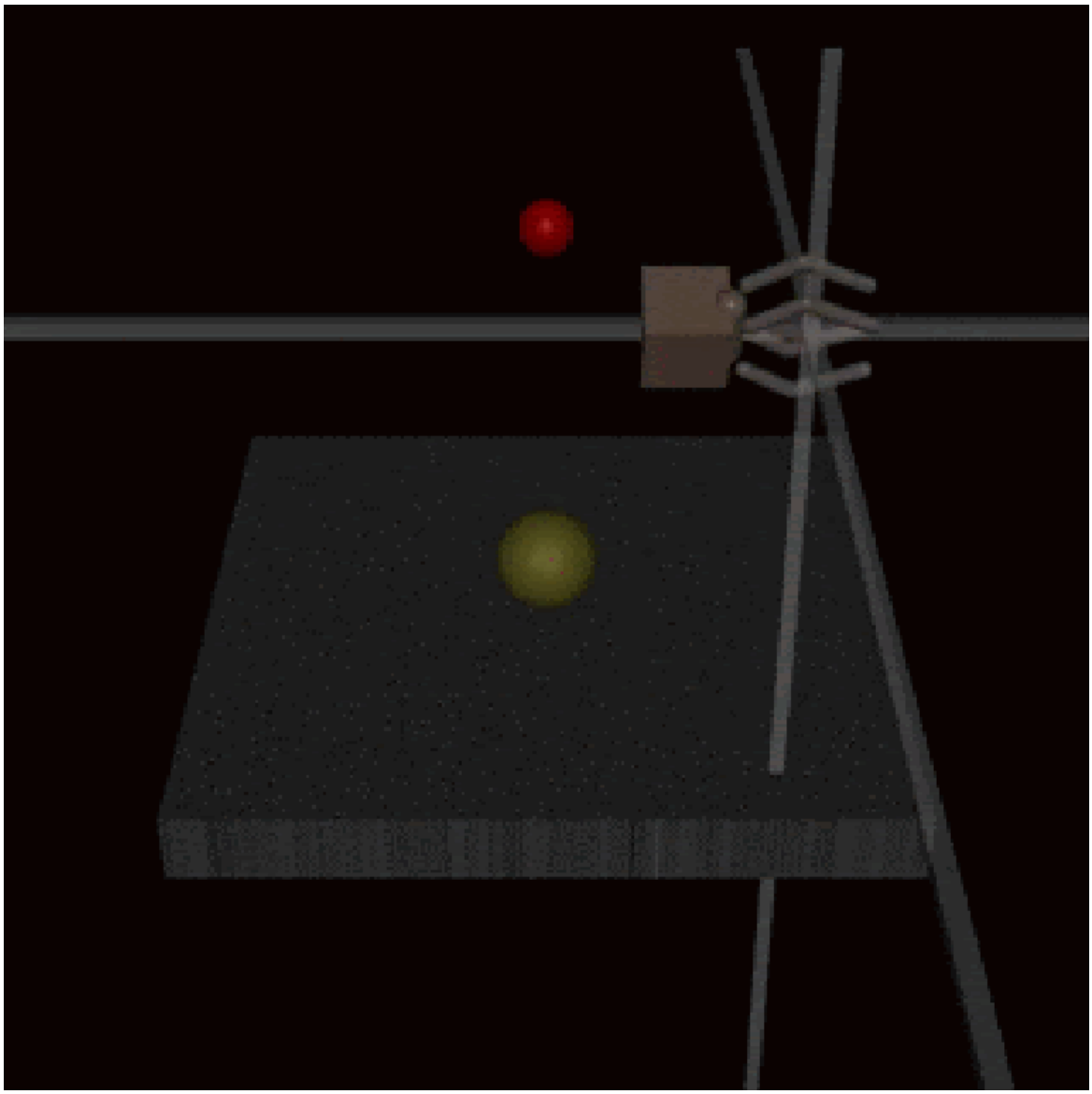
In the short run, the strategy may eliminate some problems, but in the long run it breaks the iterative design loop: problems are not seen, and therefore not iterated upon. The loop fails at the "see what goes wrong" step.

Why RLHF Is Uniquely Terrible

The software executive's strategy is the same basic idea as Reinforcement Learning from Human Feedback (RLHF). AI does something, a human looks at what happened to see if it looks good/bad, and the AI is trained on the human's feedback. Just like the software executive's anti-error-message compensation scheme, RLHF will probably result in *some*

problems actually being fixed in the short term. But it renders the remaining problems far less visible, and therefore breaks the iterative design loop. In the context of AI, RLHF makes it far more likely that a future catastrophic error will have no warning signs, that overseers will have no idea that there's any problem at all until it's much too late.

Note that this issue applies even at low capability levels! Humans overlook problems all the time, some of those mistakes are systematic, and RLHF will select for places where humans systematically overlook problems; that selection pressure applies even when the neural net lacks great capabilities.



Net learns to hold hand in front of ball, so that it looks to a human observer like the ball is being grasped. Yes, this actually happened.

This is the core reason why I consider RLHF *uniquely* terrible, among alignment schemes. It is the only strategy I know of which actively breaks the iterative design loop; it makes problems *less* visible rather than more.

Generalization: Iterate Until We Don't See Any Problems

More generally, one of the alignment failure modes I consider most likely is that an organization building AGI does see some problems in advance. But rather than addressing root causes, they try to train away the problems, and instead end up training the problems to no longer be easily noticeable.

Does This Prove Too Much?

One counterargument: don't real organizations create incentives like the software executive all the time? And we have not died of it.

Response: real organizations do indeed create incentives to hide problems all the time, and large organizations are notorious for hiding problems at every level. It doesn't even require employees consciously trying to hide things; selection pressure suffices. Sometimes important problems become public knowledge when a disaster occurs, but that's usually *after* the disaster. The only reason we haven't died of it yet is that it is hard to wipe out the human species with only 20th-century human capabilities.

Less Basic: Knowing What To Look For

Example/Analogy: The Fusion Power Generator

Suppose, a few years from now, I prompt GPT-N to design a cheap, simple fusion power generator - something I could build in my garage and use to power my house. GPT-N succeeds. I build the fusion power generator, find that it works exactly as advertised, share the plans online, and soon the world has easy access to cheap, clean power.

One problem: at no point did it occur to me to ask "Can this design easily be turned into a bomb?". Had I thought to prompt it with the question, GPT-N would have told me that the design could easily be turned into a bomb. But I didn't think to ask, so GPT-N had no reason to mention it. With the design in wide use, it's only a matter of time until people figure it out. And so, just like that, we live in a world where anyone can build a cheap thermonuclear warhead in their garage.

The root problem here is that I didn't think to ask the right question; I didn't pay attention to the right thing. An iterative design loop can sometimes help with that - empirical observation can draw our attention to previously-ignored issues. But when the failure mode does not happen in testing, the iterative design loop generally doesn't draw our attention to it. An iterative design loop does not, in general, tell us which questions we need to ask.

Ok, but can't we have an AI tell us what questions we need to ask? That's trainable, right? And we can apply the iterative design loop to make AIs suggest better questions?

Example/Analogy: Gunpowder And The Medieval Lord

Imagine a medieval lord in a war against someone with slightly more advanced technological knowledge. We're not talking modern weaponry here, just gunpowder.

To the lord, it doesn't look like the technologist is doing anything especially dangerous; mostly the technologist looks like an alchemist or a witch doctor. The technologist digs a hole, stretches a cloth over, dumps a pile of shit on top, then runs water through the shit-pile for a while. Eventually they remove the cloth and shit, throw some coal and brimstone in the hole, and mix it all together.

From the lord's perspective, this definitely looks weird and mysterious, and they may be somewhat worried about weird and mysterious things in general. But it's not obviously any more dangerous than, say, a shaman running a spiritual ceremony.

It's not until after the GIANT GODDAMN EXPLOSION that the shit-pile starts to look unusually dangerous.

Now, what helpful advice could an AI give this medieval lord?

Obviously the AI could say "the powder which comes out of that weird mysterious process is going to produce a GIANT GODDAMN EXPLOSION". The problem is, it is not cheap for the medieval lord to *verify* the AI's claim. Based on the lord's knowledge, there is no a-priori reason to expect the process to produce explosives rather than something else, and the amount of background knowledge the lord would need in order to verify the theory is enormous. The lord could in-principle verify the AI's claim experimentally, but then (a) the lord is following a complex procedure which he does not understand handed to him by a not-necessarily-friendly AI, and (b) the lord is mixing homemade explosives in his backyard. Both of these are dubious decisions at best.

So if we're *already* confident that the AI is aligned, sure, it can tell us what to look for. But if there's two AIs side-by-side, and one is saying "that powder will explode" and the other is saying "the shit-pile ceremony allows one to see the world from afar, perhaps to spot holes in our defenses", the lord cannot easily see which of them is wrong. The two can argue with each other debate-style, and the lord still will not easily be able to see which is wrong, because he would need enormously more background knowledge to evaluate the arguments correctly. And if he can't tell what the problem is, then the iterative design process can't fix it.

Example/Analogy: Leaded Gasoline

Leaded gasoline is a decent historical analogue of the Fusion Generator Problem, though less deadly. It did solve a real problem: engines ran smoother with leaded gas. The problems were nonobvious, and took a long time to manifest. The iterative design loop did not work, because we could not see the problem just by testing out leaded gas in a lab. A test would have had to run for decades, at large scale, in order to see the issue - and that's exactly what happened.

One could reasonably object to this example as an analogy, on the basis that things which drive the human species extinct would be more obvious. Dead bodies draw attention. But what about things which make the human species more stupid or aggressive? Lead did exactly that, after all. It's not hard to imagine a large-scale issue which makes humans stupid or aggressive to a much greater extent, but slowly over the course of years or decades, with the problems going undetected or unaddressed until too late.

That's not intended to be a highly probable story; there's too much specific detail. The point is that, even if the proximate cause of extinction is obvious, the factors which make that proximate cause possible may not be. A gradual path to extinction is a real possibility. When problems only manifest on long timescales, the iterative design process is bad at fixing them.

Meta Example/Analogy: Expertise and Gell-Mann Amnesia

If you don't know a fair bit about software engineering, you won't be able to distinguish good from bad software engineers.

(Assuming stats from a couple years ago are still representative, at least half my readers can probably confirm this from experience. On the other hand, last time I brought this up, one commenter said something along the lines of "Can't we test whether the code works without knowing anything about programming?". Would any software engineers like to explain in the comments why that's not the only key question to ask?)

Similarly, consider Gell-Mann Amnesia:

> You open the newspaper to an article on some subject you know well. In Murray's case, physics. In mine, show business. You read the article and see the journalist has absolutely no understanding of either the facts or the issues. Often, the article is so wrong it actually presents the story backward—reversing cause and effect. I call these the "wet streets cause rain" stories. Paper's full of them.
>
> In any case, you read with exasperation or amusement the multiple errors in a story, and then turn the page to national or international affairs, and read as if the rest of the newspaper was somehow more accurate about Palestine than the baloney you just read. You turn the page, and forget what you know.

I think there's a similar effect for expertise: software engineers realize that those outside their field have difficulty distinguishing good from bad software engineers, but often fail to generalize this to the insight that non-experts in *most* fields have difficulty distinguishing good from bad practitioners. There are of course some general-purpose tricks (and they are particularly useful expertise to have), but they only get you so far.

The difficulty of distinguishing good from bad experts breaks the iterative design loop *at a meta level*. We realize that we might not be asking the right questions, our object-level design loop might not suffice, so we go consult some experts. But then how do we iterate on our experts? How do we find better experts, or create better experts? Again, there are some general-purpose tricks available, but they're limited. In general, if we cannot see when there's a problem with our expert-choice, we cannot iterate to fix that problem.

More Fundamental: Getting What We Measure

I'm just going to directly quote Paul's post on this one:

> If I want to convince Bob to vote for Alice, I can experiment with many different persuasion strategies and see which ones work. Or I can build good predictive models of Bob's behavior and then search for actions that will lead him to vote for Alice. These are powerful techniques for achieving any goal that can be easily measured over short time periods.
>
> But if I want to help Bob figure out whether he *should* vote for Alice - whether voting for Alice would ultimately help create the kind of society he wants - that can't be done by trial and error. To solve such tasks we need to understand what we are doing and why it will yield good outcomes. We still need to use data in order to improve over time, but we need to understand how to update on new data in order to improve.
>
> Some examples of easy-to-measure vs. hard-to-measure goals:
>
> - Persuading me, vs. helping me figure out what's true. (Thanks to Wei Dai for making this example crisp.)
> - Reducing my feeling of uncertainty, vs. increasing my knowledge about the world.
> - Improving my reported life satisfaction, vs. actually helping me live a good life.
> - Reducing reported crimes, vs. actually preventing crime.
> - Increasing my wealth on paper, vs. increasing my effective control over resources.

If I want to help Bob figure out whether he *should* vote for Alice, that can't be done by trial and error. That really gets at the heart of why the iterative design loop is unlikely to suffice for alignment, even though it works so well in so many other fields. In other fields, we usually have a pretty good idea of what we want. In alignment, figuring out what we want is itself a central problem. Trial and error doesn't suffice for figuring out what we want.

So what happens if we rely on trial and error to figure out what we want? More from Paul's post:

> We will try to harness this power by constructing proxies for what we care about, but over time those proxies will come apart:
>
> - Corporations will deliver value to consumers as measured by profit. Eventually this mostly means manipulating consumers, capturing regulators, extortion and theft.
> - Investors will "own" shares of increasingly profitable corporations, and will sometimes try to use their profits to affect the world. Eventually instead of actually having an impact they will be surrounded by advisors who manipulate them into thinking they've had an impact.
> - Law enforcement will drive down complaints and increase reported sense of security. Eventually this will be driven by creating a false sense of security, hiding information about law enforcement failures, suppressing complaints, and coercing and manipulating citizens.
> - Legislation may be optimized to seem like it is addressing real problems and helping constituents. Eventually that will be achieved by undermining our ability to actually perceive problems and constructing increasingly convincing narratives about where the world is going and what's important.
>
> For a while we will be able to overcome these problems by recognizing them, improving the proxies, and imposing ad-hoc restrictions that avoid manipulation or abuse. But as the system becomes more complex, that job itself becomes too challenging for human reasoning to solve directly and requires its own trial and error, and at the meta-level the process continues to pursue some easily measured objective (potentially over longer timescales). Eventually large-scale attempts to fix the problem are themselves opposed by the collective optimization of millions of optimizers pursuing simple goals.
>
> As this world goes off the rails, there may not be any discrete point where consensus recognizes that things have gone off the rails.
>
> [...]
>
> We might describe the result as "going out with a whimper." Human reasoning gradually stops being able to compete with sophisticated, systematized manipulation and deception which is continuously improving by trial and error; human control over levers of power gradually becomes less and less effective; we ultimately lose any real ability to influence our society's trajectory.

Summary & Takeaways

In worlds where the iterative design loop works for alignment, we probably survive AGI. So, if we want to improve humanity's chances of survival, we should mostly focus on worlds where, for one reason or another, the iterative design loop fails. Fast takeoff and deceptive inner misalignment are two widely-talked-about potential failure modes, but they're not the only ones. I wouldn't consider either of them among the most *robust* ways in which the design loop fails, although they are among the most obviously and immediately *dangerous* failures.

Among the most basic robust design loop failures is problem-hiding. It happens all the time in the real world, and in practice we tend to not find out about the hidden problems until *after* a disaster occurs. This is why RLHF is such a uniquely terrible strategy: unlike most other alignment schemes, it makes problems less visible rather than more visible. If we can't see the problem, we can't iterate on it.

A more complicated and less legible class of design loop failures is not knowing what to look for. We might just not ask the right questions (as in the fusion power generator example), we might not even have enough background knowledge to recognize the right questions when they're pointed out (as in the medieval lord example), it might take a very long time to get feedback on the key problems (as in the leaded gasoline example), and at a meta level we might not have the expertise to distinguish real experts from non-experts when seeking advice (as in Gell-Mann Amnesia).

Finally, we talked about Paul's "You Get What You Measure" scenario. As Paul put it: "If I want to help Bob figure out whether he *should* vote for Alice - whether voting for Alice would ultimately help create the kind of society he wants - that can't be done by trial and error." That really captures the core reason why an iterative design loop is likely to fail for alignment, despite working so well in so many other fields: in other fields, we usually know what we want and are trying to get it. In alignment, figuring out what we want is itself a central problem, and the iterative design loop does not suffice for figuring out what we want.