Being a Responsible Developer in the Age of AI Hype

Key Takeaways

- AI is code, not magic: AR-LLMs only generate plausible text based on statistical patterns, not true reasoning or understanding.
- The current AI landscape is filled with exaggerated claims: it’s crucial to set realistic expectations to avoid contributing to AI hype.
- Developers should think and approach AI claims with skepticism, demanding evidence that can be independently verified.
- Responsible AI use involves carefully considering privacy, bias, and ethical concerns in training data and avoiding the misuse of pre-trained models in sensitive applications. Transparency and testing are key.
- Developers must communicate AI limitations transparently, ensure accountability, and not over-promise to stakeholders.

Justin Sheehy presented the "Being a Responsible Developer in the Age of AI Hype" keynote at the InfoQ Dev Summit in Boston. This article represents the talk, which starts by explaining the power a developer has and how artificial intelligence works.

You are software practitioners, and I am, too. One of the biggest failings that those of us in software tend to have is thinking that we don’t need to hear from people outside our industry, such as linguists, philosophers, psychologists, anthropologists, artists, and ethicists. Another thing about being a developer that people sometimes forget is that you have power.

One of the best tech industry analysts, Steve O’Grady, wrote this about ten years ago, and it hasn’t become less true. Your decisions matter. We need to know how to make good, responsible decisions, specifically those in front of us today because of recent trends in AI.

AI has made huge strides lately, some impressive work. The only thing more impressive has been the scale of the hype around it. I don't think it's out of line to call this moment an age of AI hype.

How Does AI Work?

First, we must figure out what we mean by Artificial Intelligence, an unhelpful term because it is so broad. Don't forget that this is a computer program; don't let yourself get tricked into thinking there's something magical about it.

What kind of programs are they? I want to credit Julia Ferraioli for reminding me of a very basic breakdown: AI has been built over the years in one of two categories, either logic and symbol processing or statistics and mapping probability distributions of things seen in the past into the future. All the recent attention has been on systems based on the probabilistic side, and I'll focus on the part people are excited about, which is large language models (LLMs).

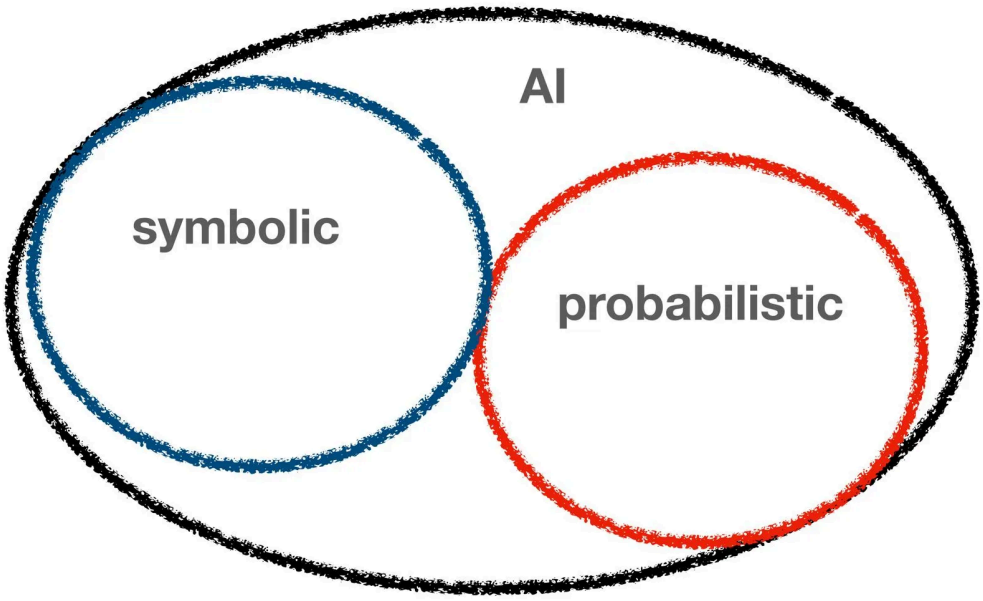


Figure 1 - AI: Symbolic or probabilistic?

One of the many advances that led to the current generation of auto-regressive, or AR-LLMs, is this concept of the transformer, increasing the quality of the output and allowing their construction to be much more parallelized. Even with many recent advances, these language models are just more efficient, parallel, scalable versions of the previous ones. This is the definition of GPT-4 from OpenAI's technical report: GPT-4 is a model pre-trained to predict the next token in a document.

When you realize that that's exactly everything an AR-LLM is doing, you understand it doesn't plan or know anything. Nothing is there about knowledge, meaning, understanding, or certainly not consciousness, just the plausible next word.

In response to being sued in the EU, OpenAI has confirmed in their legal replies that there is only one thing their system does— predicting the next most likely words to appear in response to each prompt.

It is possible that other kinds of AI systems will be created in the future, but it's not possible that the current AR-LLM systems will just change what they can do.

The Age of AI Hype

It is worth emphasizing, though, how powerful these LLMs are. If they're so awesome, why would I call this an age of hype instead of just an age of awesome AI?

People are making some pretty hyped-up statements, and billions of dollars are riding on bets related to AI companies. I want to help you evaluate these remarkable technologies more reasonably and usefully to help you make better decisions. To achieve that, we need to not be fooled by hype and nonsense.

A significant part of the current hype is the many claims that the current LLMs (ChatGPT, PaLM, Llama, Claude, etc.) are on a straightforward path to general artificial intelligence, which roughly is the kind of AI that's in most science fiction, human-like intelligence.

A paper from Microsoft Research tries to make that case, suggesting that LLMs like GPT-4 have sparks of general intelligence and includes a section on the theory of mind. They used a test about belief, knowledge, and intention from psychology, and GPT passed it. However, it turned out that if you perform the test just slightly differently, GPT fails it. That's not how it works with humans.

Another article from Blaise Agüera y Arcas of Google and Peter Norvig made an even more dramatic claim about AGI. What was the evidence? There wasn't any. The article just places the burden of proof on anyone wishing to deny the claim.

A common argument is: "Since the current AR-LLMs seem a lot more intelligent than the things we had before, we just need to provide a lot more information, a lot more computing power, and they'll keep going, and they'll get there". Here is a mistaken understanding of what LLMs are doing, as they're systems designed to synthesize text. More data and more compute might get them closer to the top of that tree, but never to the moon.

Another claim is that since LLMs produce responses similar to the ones from a person, they pass a fundamental test for intelligence. This is a misunderstanding of what the Turing test is. Alan Turing called this test "the imitation game", and imitating human text is very different from being intelligent. Turing even argues that trying to answer a question about machines being intelligent is nonsense.

Another claim is that people just do the same thing that LLMs like ChatGPT do, just probabilistically stringing along words. People making this claim are a lot like LLMs themselves, really good at sounding plausible, but there's no actual knowledge or understanding there at all.

The term "stochastic parrot" comes from the title of a critical paper about possible risks in LLMs. This is a sensible topic for ethicists to write about and is the paper that got Google's ethical AI leaders fired for writing it. Margaret Mitchell is listed as Shmargaret Shmitchell since she wasn't allowed to provide her real name.

AR-LLMs are probabilistic repeating machines, much like a parrot that learns how to make the sounds of humans nearby but has no idea what they mean. Sam Altman, CEO of OpenAI, claims that this is all we are, too, which is a bizarre claim.

The problem is a fundamental lack of understanding that language is a tool used by humans, with its form being used to communicate meaning. When we have an idea, a belief, or some knowledge, we use language to communicate it. LLMs have no ideas, beliefs, or knowledge; they synthesize text without meaning.

Yann LeCun, the head of AI research at Meta, a Turing Award winner, and an insider to developing LLMs, wrote: "A system trained on language alone will never approximate human intelligence".

Language alone does not contain all of what is relevant or needed for human-like intelligence. Something trained only on form cannot develop a sense of meaning.

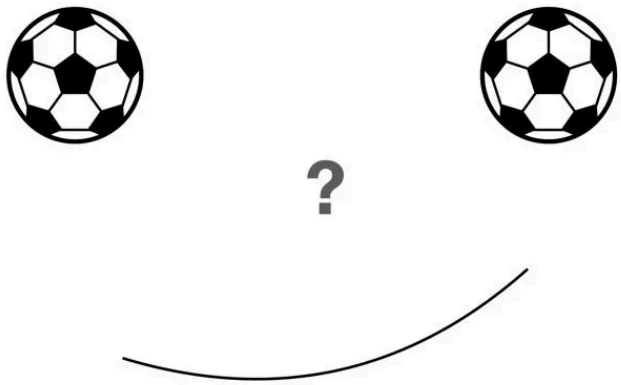


Figure 2 - An example of pareidolia

There is no face here, but you saw one. That effect is called face pareidolia; most humans do it, even reading emotions that do not exist in such images. The face is a creation of your mind. Similarly, when you read a bunch of text structurally very similar to what a person might write, it's very challenging not to believe that there's some intention or meaning behind that text. It's so challenging that even people who know how these systems work can be fooled. It is challenging, but you can rise to that challenge. Remember, these are just computer programs.

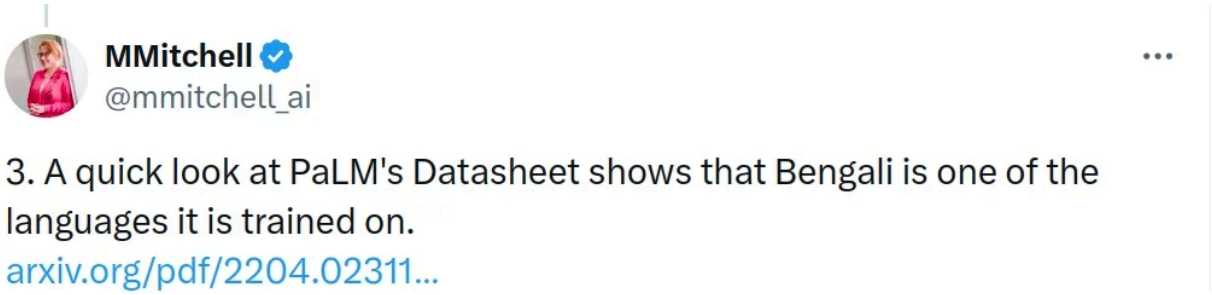
Hallucination

The use of the word "hallucination" about LLMs is a trick played on all of us. When a person hallucinates, their sense of truth and meaning has become disconnected from their observed reality. AR-LLMs have no sense of truth and meaning and no observed reality. They're always just doing the exact same thing, statistically predicting the next word. There's no meaning, no intention, no sense of what's true. Depending on how you look at it, they're either always hallucinating or they never are. Even if we accept the use of the word hallucination, we shouldn't think that it will go away as these systems mature. Producing text not grounded in any external reality is what they do, not a bug. LLMs are never wrong. They can only be wrong if you think they're trying to correctly answer your question. They're producing text that is likely, based on the text coming before it, to look like it could go next. That's not the same thing.

Encouraged by science fiction-inspired creative thinking about AGI, another source of

confusion is the idea that arbitrary behavior can emerge from an LLM. It's a fun idea but isn't connected to how these things work. They are cool programs, but programs do not evolve independently.

We read how Google's LLM Gemini learned Bangla, the Bengali language, without ever being trained on it. That's the kind of story that could make someone become a believer, but it only took a day for Margaret Mitchell to find that the language in question was in the training set.



A few months later, Google released an excellent video on YouTube that showed amazing new behaviors, like identifying images in real time in spoken conversations with a user. It turned out that it was fake; it was just video editing.

People and AI (Human Behavior)

Let's switch briefly away from the narrow focus on LLMs to the wider topic of people and AI. Some of you might have heard of the Mechanical Turk, a fantastic chess-playing machine. One of the earliest famous examples of AI, it played against Benjamin Franklin and Napoleon and won most of the games it played. We've had AI good enough to beat human players at chess for over 200 years, but there was a human chess player inside. It's a great trick, and it is also how a lot of today's AI works.

We see an AI system with an outrageous claim about its human-like capability, like the Mechanical Turk in Amazon's AI checkout lanes. Similarly, the Cruise driverless cars had an average of more than one (remote) driver per car. These are not only cases of humans inside the AI; they're also hype. And they're good reminders that if something acts amazingly well like a human and you can't get adversarial proof of how it works, maybe don't start by believing it isn't a human.

Even where there's no individual man behind the curtain, there are human elements of the AI systems. For instance, one of the important breakthroughs enabling ChatGPT to exist is Reinforcement Learning from Human Feedback (RLHF). Humans read many answers to

many initial prompts that humans write to produce their original training set, then interact with the result to build a reward model. It's just a program plus an enormous amount of low-paid human labor.

Using ChatGPT or similar services relies on the labor of thousands of people who are paid a couple of dollars an hour to do work that no software practitioner would do. That's a whole class of ethical problems.

Developers Make Things

Nobody wants to be left behind: "My boss told me I have to put some AI on it so we can say AI in our next press release".

How can we do it right? We're developers; we make software systems. We already know that these systems are just computers running programs. A few of you may be the developers of those AI systems; others are using those systems to build other systems, like GitHub Copilot, or to put a chatbot on a website. The majority of the developers use the LLMs or wrappers around them. Let's discuss some of the choices we can make.

How can we ensure that we use them wisely? If you wouldn't email a piece of your company's confidential information, like source code or business data, you should think twice before putting the same data into a service that another company runs. Unless you have a contract with them, they may pass that information on to someone else. For this reason, many enterprises have policies against using systems like ChatGPT or Copilot.

Since so many of the legal issues are still being worked out about property rights for works that have passed through an LLM blender, you may want to be cautious before putting code, text, images, or other content that came out from an LLM that you didn't train yourself into your products or systems.

This isn't just about avoiding lawsuits and is not a hypothetical concern. I've been involved in discovery and diligence for startup acquisitions; everything goes a lot smoother if there are clear answers to where everything comes from.

Be careful not to send anything you wouldn't make publicly available to these systems or use any of their output in anything you wish to own unless you train your own model. If you train it yourself on legitimate content you have the right to use and use the model to produce material for your own consumption and review, you're on pretty solid ground.

Even if you only do one or two of the above guidelines, it puts you in a much better place.

For example, using an LLM to help proofread your work can be useful. I do this. Or, as Simon Willison does, you could use them to help you build summaries of your podcasts or blog posts that you will then edit. You can also use them as a debate partner, an idea I got from Corey Quinn, to jumpstart your writing. Having yourself deeply in that cycle is critical for all of those because there will be errors.

You can also choose to use LLMs because of the inevitable errors, for example, to teach developers debugging skills. Since LLMs create plausible text and code is text, they're great at creating software that is buggy, but not in obvious ways.

I'm focusing on use cases where faulty output is either OK or even the goal. When developers forget this and send LLM-generated content to others, they tend to get into trouble. An LLM might cite academic work, but the citations might not be real. Or refer to legal precedents that don't exist but make for plausible enough text that a lawyer cites them to a judge.

This is even worse for code. Having an LLM write code to teach students debugging is great. Having it suggest improvements, as a not-too-skilled pair programmer, can be fine, too. Having it actually write the code you're going to ship is riskier. The difficult part of software creation is not writing code but communication, understanding, and judgment. LLMs can't do that job for you, and I've seen smart folks get tricked.

I get told often that these AR-LLMs can reason and code, but they don't have any reasoning. I asked the simplest question I could think of that it wouldn't have already memorized the answer to. I asked ChatGPT 4o to count the number of times the letter "e" appears in my name, Justin Sheehy—3 e's it says. That's not right.

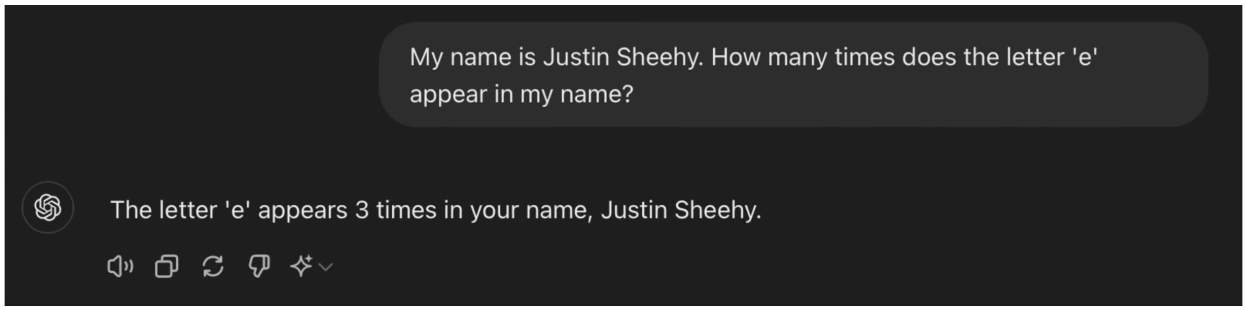


Figure 3 - ChatGPT prompt

I've been told just to ask more questions—asking it to show its work and giving precise direction. As always, it sounds very confident. It backs up the answer by showing its work. This is extremely simple and should be in its sweet spot if it could do any reasoning or counting. I tried similar questions using different content with similar results.

LLMs don’t reason; they don’t "sometimes get it wrong". They’re always just probabilistically spewing out text shaped like something that might come next. Sometimes, by chance, that also happens to be correct. Use them accordingly.

What about when we’re not just users? What about when we move down the stack and we build AI into our software? Here, it gets even more important to be careful of what content the model was trained on. Either you train it yourself, or you go the easy route and use an API around one of the big models trained on the whole internet. There’s an enormous wealth of knowledge on the web, but there are also things you might not want your software to repeat. Part of being responsible is not bringing the worst parts of what’s out there through your system. People tend to feel that an answer from a computer via an algorithm is somehow objective. We’re developers; we know all about "garbage in, garbage out". If the whole internet is what goes in, we know what will come out.

This isn’t hypothetical. Developers are making these choices today, embedding these pre-trained language models into systems, making important decisions, and the results on race, gender, and religion are predictable. How can we do better? We can start with testing. We have testing tools for the rest of our software; we can test our models for bias using tools like the open-source AI Fairness 360 toolkit from IBM. It’s not enough, but it’s a start.

Another set of irresponsible decisions being made right now can be seen walking around vendor booths at conferences and counting the products that have become "AI powered". This is not harmless.

The AI washing exercise, the money grab by saying, we’ll solve it with AI, somehow, can mean that other systems don’t get the resources they need. This isn’t just failure; it’s theft in the form of opportunity cost, preventing important work from happening. Or worse, you can give people confidence that an algorithm is on the job and cause real life-or-death decisions to be made wrongly. What can we do? We can talk with our CEOs and product managers about the value that our software systems provide. Instead of adding a few hyped buzzwords, we can determine if and how adding AI components will add real value. We can help them learn to ask better questions than "Does it have AI in it?"

Accountability in the Age of AI

Whether incorporating an LLM or building your own model, being a responsible developer requires accountability. Your company is accountable for what it ships.

What does accountability look like? As an LLM cannot be forced not to hallucinate, you

have to be prepared to take responsibility for whatever output it produces. The AI chatbot that lets your company reduce support staff might make the company lose money: they might give out discounted products or unexpected refunds. It's your responsibility to make sure that the company knows what the systems can do. How can we do it? We need not to lie or wildly overpromise; we need not to make the hype problem worse.

If you can't do something legally and safely, don't do it. Almost everyone will agree with the statement in general, but I hear objections to specific cases: "If we complied with the law, we wouldn't be able to provide this service"; "If we took the time to make sure there was no child pornography in our training sets, we wouldn't have made this image generator"; "We have to violate the rights of hundreds of thousands of people to train a huge AR-LLM".

Of course, I don't want to hold back the future of AI, but the success of a product or company does not excuse irresponsibility. Don't put your product ahead of the safety or rights of others. If you have to violate other people's safety to ship something, don't ship it. Research can continue and thrive; it's up to us to get there safely.

Alignment

Let's talk about alignment, the idea of building AGI or general intelligence, and ensuring that AI shares our human values. These are well-meaning ideas, but they are still wild science fiction. How do you even start getting to AGI? We're multiple huge breakthroughs away from it if it is possible. Bringing ethical frameworks into the development of AI or any other technology is worthwhile.

Anthropic has an interesting paper about alignment and general-purpose AI. Despite not agreeing about the trajectory toward general-purpose AI, I think their framework is very interesting, and we can use it. The premise is that an AI is aligned if it is helpful, honest, and harmless: 3 H's. It will only do what is in human's best interests, only convey accurate information, and avoid doing things that harm people.

You can make use of it right now by leaving out the "AI" part of the definition and applying this framework to yourself. If you can live up to the framework for aligned AI, then you have what it takes to be a responsible developer. Make sure that what you build is helpful and a solution to a real problem. Make sure you are honest about what you build without overselling or making the hype problem worse. Make sure you minimize the harm caused by the things you build or caused by building them. You need to help people, be

honest with people, and reduce damage to people. Think about those people as you make your decisions.

You have great responsibility because developers have great power, and you get to help decide what the future looks like. Let’s make it a good one for people.

About the Author

Justin Sheehy

Justin Sheehy has spent most of his career thinking about how to make systems - composed of both computers and humans - safer and more resilient. He is currently Chief Architect for Infrastructure Engineering and Operations at Akamai. Previously he’s played roles including Chief Technologist for Cloud Native Storage (VMware), CTO (Basho), Principal Scientist (MITRE), Distributed Systems Engineer (multiple times), and more. He’s equally happy whether writing code, doing research, or managing teams. No matter which of those he’s doing at the moment, he tries to prevent more problems than he causes.

Show more
Show less