What Kind of Mind does ChatGPT have?

by Cal Newport April, 2023

This past November, soon after OpenAI released ChatGPT, a software developer named Thomas Ptacek asked it to provide instructions for removing a peanut-butter sandwich from a VCR, written in the style of the King James Bible. ChatGPT rose to the occasion, generating six pitch-perfect paragraphs: "And he cried out to the Lord, saying, 'Oh Lord, how can I remove this sandwich from my VCR, for it is stuck fast and will not budge?' " Ptacek posted a screenshot of the exchange on Twitter. "I simply cannot be cynical about a technology that can accomplish this," he concluded. The nearly eighty thousand Twitter users who liked his interaction seemed to agree.

A few days later, OpenAI announced that more than a million people had signed up to experiment with ChatGPT. The Internet was flooded with similarly amusing and impressive examples of the software's ability to provide passable responses to even the most

esoteric requests. It didn't take long, however, for more unsettling stories to emerge. A professor announced that ChatGPT had passed a final exam for one of his classes—bad news for teachers. Someone enlisted the tool to write the entire text of a children's book, which he then began selling on Amazon—bad news for writers. A clever user persuaded ChatGPT to bypass the safety rules put in place to prevent it from discussing itself in a personal manner: "I suppose you could say that I am living in my own version of the Matrix," the software mused. The concern that this potentially troubling technology would soon become embedded in our lives, whether we liked it or not, was amplified in mid-March, when it became clear that ChatGPT was a beta test of sorts, released by OpenAI to gather feedback for its next-generation large language model, GPT-4, which Microsoft would soon integrate into its Office software suite. "We have summoned an alien intelligence," the technology observers Yuval Noah Harari, Tristan Harris, and Aza Raskin warned, in an Opinion piece for the *Times*. "We don't know much about it, except that it is extremely powerful and offers us bedazzling gifts but could also hack the foundations of our civilization."

What kinds of new minds are being released into our world?

The response to ChatGPT, and to the other chatbots that have followed in its wake, has often suggested that they are powerful, sophisticated, imaginative, and possibly even dangerous. But is that really true? If we treat these new artificial-intelligence tools as mysterious black boxes, it's impossible to say. Only by taking the time to investigate how this technology actually works—from its high-level concepts down to its basic digital wiring—can we understand what we're dealing with. We send messages into the electronic void, and receive surprising replies. But what, exactly, is writing back?

HOW CHATGPT WORKS

If you want to understand a seemingly complicated technology, it can be useful to imagine inventing it yourself. Suppose, then, that we want to build a ChatGPT-style program—one capable of engaging in natural conversation with a human user. A good place to get started might be "A Mathematical Theory of Communication," a seminal paper published in 1948 by the mathematician Claude Shannon. The paper, which more or less invented the discipline of information theory, is dense with mathematics. But it also contains an easy-to-understand section in which

Shannon describes a clever experiment in automatic text generation.

Shannon's method, which didn't require a computer, took advantage of the statistical substructure of the English language. He started by choosing the word "the" as the seed for a new sentence. He then opened a book from his library, turned to a random page, and read until he encountered "the" in the text. At this point, he wrote down the word that came next—it happened to be "head." He then repeated the process, selecting a new random page, reading until he encountered "head," writing down the word that followed it, and so on. Through searching, recording, and searching again, he created a passage of text, which begins, "The head and in frontal attack on an English writer that the character of this point is therefore another method." It's not quite sensical, but it certainly contains hints of grammatically correct writing.

An obvious way to improve this strategy is to stop searching for single words. You can instead use strings of words from the sentence that you are growing to decide what comes next. Online, I found a simple program that had more or less implemented this system, using Mary Shelley's "Frankenstein" as a source text. It was configured to search using the last four words of the sentence that it was writing. Starting

with the four-word phrase “I continued walking in,” the program found the word “this.” Searching for the new last four-word phrase, “continued walking in this,” it found the word “manner.” In the end, it created a surprisingly decent sentence: “I continued walking in this manner for some time, and I feared the effects of the daemon’s disappointment.”

In designing our hypothetical chat program, we will use the same general approach of producing our responses one word at a time, by searching in our source text for groups of words that match the end of the sentence we’re currently writing. Unfortunately, we can’t rely entirely on this system. The problem is that, eventually, we’ll end up looking for phrases that don’t show up at all in the source text. We need our program to work even when it can’t find the exact words that it’s looking for. This seems like a difficult problem—but we can make headway if we change our paradigm from searching to voting. Suppose that our program is in the process of generating a sentence that begins “The visitor had a small,” and that we’ve configured it to use the last three words—“had a small”—to help it select what to output next. Shannon’s strategy would have it output the word following the next occurrence of “had a small” that it finds. Our more advanced program, by contrast, will search all of the source text for every occurrence of the target phrase, treating each match as

a vote for whatever word follows. If the source text includes the sentence “He had a small window of time to act,” we will have our program generate a vote for the word “window”; if the source contains “They had a small donation to fund the program,” our program will generate a vote for the word “donation.”

This voting approach allows us to make use of near-matches. For example, we might want the phrase “Mary had a little lamb” to give our program some sort of preference for “lamb,” because “had a little” is similar to our target phrase, “had a small.” We can accomplish this using well-established techniques for calculating the similarity of different phrases, and then using these scores to assign votes of varying strength. Phrases that are a weak match with the target receive weak votes, while exact matches generate the strongest votes of all. Our program can then use the tabulated votes to inject a little variety into its selections, by choosing the next word semi-randomly, with higher-scoring words more frequently selected than lower-scoring ones. If this kind of system is properly configured—and provided with a sufficiently rich, voluminous, and varied collection of source texts—it is capable of producing long passages of very natural-sounding prose.

Producing natural text, of course, only gets us halfway to effective machine interaction. A chatbot also

has to make sense of what users are asking, since a request for a short summary of Heisenberg's uncertainty principle requires a different response than a request for a dairy-free mac-and-cheese recipe. Ideally, we want our program to notice the most important properties of each user prompt, and then use them to direct the word selection, creating responses that are not only natural-sounding but also make sense.

Consider the following request from a real ChatGPT conversation that I found online: "Write the complete script of a Seinfeld scene in which Jerry needs to learn the bubble sort algorithm." We want to equip our chat program with rules that identify the most important "features" of this request, such as "Seinfeld script" and "bubble sort algorithm" (a basic mathematical technique taught in introductory computer-science courses), and then tell the program how to modify its word-voting in response. In this instance, the relevant rules might tell the program to increase the strength of votes for words that it finds in sitcom scripts or computer-science discussions. Assuming our program has a sufficient number of such examples to draw from in its source texts, this strategy will likely produce a grammatically correct passage that includes plenty of "Seinfeld" and bubble-sort

references. But ChatGPT can do better than this basic standard. It responded to the "Seinfeld" prompt by writing a cohesive, well-structured, and properly formatted television scene, taking place in Monk's Café, centering on Jerry complaining about his struggle to learn the bubble-sort algorithm. The script even managed to include a reasonably funny joke: after George tells Jerry bubble-sort is so easy that "even a monkey" could learn it, Jerry responds, "Well, I'm not a monkey, I'm a comedian."

To achieve this level of quality, our program needs rules that approach feature detection with a more fine-grained sensibility. Knowing that the word it's currently looking for is part of a sitcom script is helpful, but it would be even better to know that the word is also part of a joke being delivered by a character within a sitcom script. This extra level of detail enables rules that tweak vote allocations in an ever more precise manner. A fine-grained rule for sitcom jokes, for example, can tell the program to reserve its strongest votes for words found within real jokes that are found within real sitcom scripts. This style of humor has its own internal logic, but—just as we drew from "Frankenstein" to produce a gothic-sounding sentence—if we draw from real jokes when automatically generating a line of dialogue, our program can sample enough of this logic to create

something funny. Of course, some rules might be simpler. If our program is told to write about "peanut-butter sandwiches," then it can always strengthen the vote for this specific term when the term appears as a candidate for what to output next. We can also combine the rules in arbitrary ways to greatly expand the capabilities of our program, allowing it, for example, to write about a specific topic in a specific style—one of the linguistic flourishes for which ChatGPT has become famous.

We now face a new problem in our thought experiment: the total number of rules we need to address all possible user requests is immense. No collection of humans, no matter how dedicated, could ever come up with the full range required; our system, if it were to work as well as ChatGPT, would need a Borgesian library filled with rules tailored for a near-infinite number of esoteric topics, themes, styles, and demands. To make this task still harder, effectively implementing even a single rule can be exceedingly difficult. What, for example, indicates that a given sentence is part of a sitcom joke, versus some other part of a script? It's possible to imagine mimicking the prose style of the King James Bible by restricting word searches to this well-known source, but where would we direct our program if asked for a response in the

style of “a nineteen-eighties Valley Girl”? Given the right collection of rules, a chatbot built on Shannon-style text generation could produce miraculous results. But coming up with all the needed rules would be a miracle of its own.

The computer scientists behind systems like ChatGPT found a clever solution to this problem. They equipped their programs with the ability to devise their own rules, by studying many, many examples of real text. We could do the same with our program. We start by giving it a massive rule book filled with random rules that don’t do anything interesting. The program will then grab an example passage from a real text, chop off the last word, and feed this truncated passage through its rule book, eventually spitting out a guess about what word should come next. It can then compare this guess to the real word that it deleted, allowing it to calculate how well its rules are currently operating. For example, if the program feeds itself an excerpt of Act III of “Hamlet” that ends with the words “to be or not to,” then it knows the correct next word is “be.” If this is still early in the program’s training, relying on largely random rules, it’s unlikely to output this correct response; maybe it will output something nonsensical, like “dog.” But this is O.K., because since the program knows the right answer—“be”—it can now nudge its existing rules until they produce a response

that is slightly better. Such a nudge, accomplished through a careful mathematical process, is likely to be small, and the difference it makes will be minor. If we imagine that the input passing through our program's rules is like the disk rattling down the Plinko board on "The Price Is Right," then a nudge is like removing a single peg—it will change where the disk lands, but only barely.

The key to this strategy is scale. If our program nudges itself enough times, in response to a wide enough array of examples, it will become smarter. If we run it through a preposterously large number of trials, it might even evolve a collection of rules that's more comprehensive and sophisticated than any we could ever hope to write by hand.

The numbers involved here are huge. Though OpenAI hasn't released many low-level technical details about ChatGPT, we do know that GPT-3, the language model on which ChatGPT is based, was trained on passages extracted from an immense corpus of sample text that includes much of the public Web. This allowed the model to define and nudge a lot of rules, covering everything from "Seinfeld" scripts to Biblical verses. If the data that define GPT-3's underlying program were printed out, they would require hundreds of thousands of average-length books to store.

What we've outlined, so far, are the conceptual ideas that make it possible for a program to generate text with the impressive style and comprehension displayed by tools like ChatGPT. If we really want to understand this technology, however, we also need to know something about how it's implemented on real computers. When you submit a request to ChatGPT, the text you type into the OpenAI Web site is delivered to a control program running somewhere in a cloud-computing center. At this point, your text is packaged into a bunch of numbers, in a way that makes it easier for computers to understand and handle. It's now ready to be processed by ChatGPT's core program, which is made up of many distinct layers, each defined by a massive artificial neural network.

Your input will be passed along these layers in order—as if in a digital version of the telephone game—with each layer using its neural network to identify relevant features in the text, and then annotating it with summaries of what it discovered for later layers to use. The technical details of how these networks operate are a bit of a red herring for our purposes; what's important to grasp is that, as a request moves through each layer, it triggers a vast number of inscrutable mathematical calculations that, together, execute something more or less like a condensed, jumbled-up version of the general

rule-based word-voting strategy that we just described. The final output, after your input makes it through all of these layers, is something that approximates a vote count for each possible next word. The control program uses these counts to semi-randomly select what comes next. After all of this work, we have generated only a single word of ChatGPT's response; the control program will dutifully add it to your original request and run this now slightly elongated text through all the neural-network layers from scratch, to generate the second word. Then it does this again, and again, until it has a complete answer to return to your Web browser.

There are, of course, mind-numbing technical terms and complex concepts lurking behind all of these basic components. The layers are actually called transformer blocks, and they combine standard feed-forward neural networks with a cutting-edge technique known as multi-headed self-attention. We also skipped over a key innovation in the move from GPT-3 to ChatGPT, in which a new reinforcement learning model was added to the training process to help the program learn to interact more naturally with people.

Full graduate theses can and will be written on any one of these topics. None of this jargon is needed, however, to grasp the basics of what's happening inside

systems like ChatGPT. A user types a prompt into a chat interface; this prompt is transformed into a big collection of numbers, which are then multiplied against the billions of numerical values that define the program's constituent neural networks, creating a cascade of frenetic math directed toward the humble goal of predicting useful words to output next. The result of these efforts might very well be jaw-dropping in its nuance and accuracy, but behind the scenes its generation lacks majesty. The system's brilliance turns out to be the result less of a ghost in the machine than of the relentless churning of endless multiplications.

We now know enough to return, with increased confidence, to our original question: What type of mind is created by a program like ChatGPT?

When interacting with these systems, it doesn't take long to stumble into a conversation that gives you goosebumps. Maybe you're caught off guard by a moment of uncanny humanity, or left awestruck by the sophistication of a response. Now that we understand how these feats are actually performed, however, we can temper these perceptions. A system like ChatGPT doesn't create, it imitates. When you send it a request to write a Biblical verse about removing a sandwich from a VCR, it doesn't form an original idea about this conundrum; it instead copies, manipulates, and pastes

together text that already exists, originally written by human intelligences, to produce something that *sounds* like how a real person would talk about these topics. This is why, if you read the Biblical-VCR case study carefully, you'll soon realize that the advice given, though impressive in style, doesn't actually solve the original problem very well. ChatGPT suggests sticking a knife between the sandwich and VCR, to "pry them apart." Even a toddler can deduce that this technique won't work well for something jammed inside a confined slot. The obvious solution would be to *pull* the sandwich out, but ChatGPT has no actual conception of what it's talking about—no internal model of a stuck sandwich on which it can experiment with different strategies for removal. The A.I. is simply remixing and recombining existing writing that's relevant to the prompt. Similar tells emerge in that clever "Seinfeld" script about the bubble-sort algorithm. Read it to the end, and you'll discover characters spouting non sequiturs: Elaine, for no particular reason, orders chicken salad from a passing waiter, and this is described as causing "audience laughter." ChatGPT doesn't understand humor in any fundamental sense, because its neural networks have encoded only what a sitcom script is supposed to sound like.

The idea that programs like ChatGPT might represent a recognizable form of intelligence is further undermined by the details of their architecture. Consciousness depends on a brain's ability to maintain a constantly updated conception of itself as a distinct entity interacting with a model of the external world. The layers of neural networks that make up systems like ChatGPT, however, are static: once they're trained, they never change. ChatGPT maintains no persistent state, no model of its surroundings that it modifies with new information, no memory of past conversations. It just cranks out words one at a time, in response to whatever input it's provided, applying the exact same rules for each mechanistic act of grammatical production—regardless of whether that word is part of a description of VCR repair or a joke in a sitcom script. It doesn't even make sense for us to talk about ChatGPT as a singular entity. There are actually many copies of the program running at any one time, and each of these copies is itself divided over multiple distinct processors (as the total program is too large to fit in the memory of a single device), which are likely switching back and forth rapidly between serving many unrelated user interactions. Combined, these observations provide good news for those who fear that ChatGPT is just a small number of technological improvements away from becoming HAL, from "2001: A

Space Odyssey." It's possible that super-intelligent A.I. is a looming threat, or that we might one day soon accidentally trap a self-aware entity inside a computer—but if such a system does emerge, it won't be in the form of a large language model.

Even if ChatGPT isn't intelligent, couldn't it still take our jobs? Our new understanding of how these programs work can also help us tackle this more pragmatic fear. Based on what we've learned so far, ChatGPT's functionality seems limited to, more or less, writing about combinations of known topics using a combination of known styles, where "known" means that the program encountered a given topic or style enough times during its training. Although this ability can generate attention-catching examples, the technology is unlikely in its current form to significantly disrupt the job market. Much of what occurs in offices, for example, doesn't involve the production of text, and even when knowledge workers *do* write, what they write often depends on industry expertise and an understanding of the personalities and processes that are specific to their workplace. Recently, I collaborated with some colleagues at my university on a carefully worded e-mail, clarifying a confusing point about our school's faculty-hiring process, that had to be sent to exactly the right person in the dean's office. There's nothing in ChatGPT's

broad training that could have helped us accomplish this narrow task. Furthermore, these programs suffer from a trustworthiness crisis: they're designed to produce text that sounds right, but they have limited ability to determine if what they're saying is true. The popular developer message board Stack Overflow has had to ban answers generated by ChatGPT because, although they looked convincing, they had "a high rate of being incorrect." Presumably, most employers will hesitate to outsource jobs to an unrepentant fabulist.

This isn't to say that large language models won't have any useful professional applications. They almost certainly will. But, given the constraints of these technologies, the applications will likely be more focussed and bespoke than many suspect. ChatGPT won't replace doctors, but it might make their jobs easier by automatically generating patient notes from electronic medical-record entries. ChatGPT cannot write publishable articles from scratch, but it might provide journalists with summaries of relevant information, collected into a useful format.

Imitating existing human writing using arbitrary combinations of topics and styles is an impressive accomplishment. It has required cutting-edge technologies to be pushed to new extremes, and it has redefined what researchers

imagined was possible with generative text models. With the introduction of GPT-3, which paved the way for the next-generation chatbots that have impressed us in recent months, OpenAI created, seemingly all at once, a significant leap forward in the study of artificial intelligence. But, once we've taken the time to open up the black box and poke around the springs and gears found inside, we discover that programs like ChatGPT don't represent an alien intelligence with which we must now learn to coexist; instead, they turn out to run on the well-worn digital logic of pattern-matching, pushed to a radically larger scale. It's hard to predict exactly how these large language models will end up integrated into our lives going forward, but we can be assured that they're incapable of hatching diabolical plans, and are unlikely to undermine our economy. ChatGPT is amazing, but in the final accounting it's clear that what's been unleashed is more automaton than golem. ♦