I Do Not Think It Means What You Think It Means: Artificial Intelligence, Cognitive Work & Scale

Kevin Scott

Over the past decade, AI technologies have advanced by leaps and bounds. Progress has been so fast, voluminous, and varied that it can be a challenge even for experts to make sense of it all. In this essay, I propose a framework for thinking about AI systems, specifically the idea that they are ultimately tools developed by humans to help other humans perform an increasing breadth of their cognitive work. Our AI systems for assisting us with our cognitive work have become more capable and general over the past few years. This is in part due to a confluence of novel AI algorithms and the availability of massive amounts of data and compute. From this, researchers and engineers have been able to construct large, general models that serve as flexible and powerful building blocks that can be composed with other software to drive breakthroughs in the natural and physical sciences, to solve hard optimization and strategy problems, to perform perception tasks, and even to assist with complex cognitive tasks like coding.

When I say the word "work," what do I mean? In the mornings, when I tell my children that "I'm going to work," they understand that I am about to get into a car, drive to my office, and, for the rest of the day, do a set of things alongside my coworkers for an employer who pays my salary. When I tell my wife that "I'm going to work in the shop for a while," she understands that I am headed to my workshop where I will use a variety of tools that I hold dear to tinker around on personal projects. When I say that "I'm going to work in the garden" or "I'm going to work on this essay," the people to whom I am speaking almost always understand what I mean. Work in all these contexts means me, a human being, applying effort to achieve some effect. In these contexts, we all have some shared understanding of what the applied efforts entail, and why the effects are worth achieving.

In the late eighteenth century, accelerating into the nineteenth and twentieth centuries, individual members of society had cause to think about work in new ways. As society industrialized and humans devised new ways to use machines to

https://doi.org/10.1162/DAED_a_01901

do work, nearly every aspect of human life changed. As these machines became increasingly complex, and as we began to use them to perform types of work that had previously been performed through a combination of human labor and less powerful tools, we needed new language and new scientific, technical, and social shared understandings for these new forms of machine-assisted work.

Driven by the intellectual and industrial revolutions of the period, by the last half of the nineteenth century, scientists like Nicolas Léonard Sadi Carnot, James Prescott Joule, Rudolf Clausius, Lord Kelvin, James Maxwell, Ludwig Boltzmann, and others had given us a simple but powerful definition of work – weight lifted through a height – and a rich scientific theory – thermodynamics – that helped us better understand not just the natural world, but how to better engineer, build, and direct the new forms of machine work shaping society. That nineteenth-century scientific definition of work is very much relevant today, but it is characteristic of its time. The work that it defines is physical. Understanding the nature of physical work was and is necessary to understand the machinery of the universe and was essential in constructing an industrial society.

When I get in my car, drive to my office, and do things alongside my coworkers, "weight lifted through a height" is perhaps not the most relevant definition of the work that I, and many others, do every day. I meet with people. I listen. I coach and mentor. I attempt to make a very small number of meaningful decisions. I read and digest information. I think. I imagine. I code. I write. With all these efforts, the effects that I am trying to achieve are the solutions of problems. For me those might be: Can we use our AI supercomputers to make molecular dynamics simulations go much faster so that we can solve a more interesting set of problems in biology? How can we make sure that our next machine learning model does not produce adverse effects? Can we work around firmware issues to prevent a compute shortage in our AI training clusters? Can I understand enough of what a coworker is trying to achieve to meaningfully assist them? The interesting thing about all these problems and their solutions is that the work required to solve them is almost entirely cognitive.

If you are reading this essay, I would wager that you earn some or all of your living doing cognitive work. Perhaps, if you made a full accounting of your work time, you would discover that, if not most of the effort that you exert in your work, then a majority of the effects that you produce are more of the mind than the body. I am not arguing that our bodies are mere instruments of the mind. And I am certainly not arguing that one form of work is superior to another. I am attempting to make a more prosaic assertion: I am a knowledge worker; and you may be, too. Moreover, even though we understand the nature of our work well enough to do it, and more of us are earning our living this way with each passing year, we have not yet crisply defined what cognitive work is nor how to measure

it. As AI technologies become more capable, and as we use them to do more things that are inarguably cognitive work, this lack of a foundational definition makes it increasingly difficult to predict and engineer the changes that machines will bring to cognitive work in the coming years. Will AI become yet another instrument or tool that we use to express our humanity and creativity, that allows us to better explore and understand ourselves and the world around us, and that evolves the nature of work once again just as the machines of the industrial revolution have done over the past two centuries? Or will AI become something else?

Wikipedia dodges defining what knowledge work is by defining the knowledge worker instead as someone whose main capital is knowledge. The knowledge worker entry then lists examples: "programmers, physicians, pharmacists, architects, engineers, scientists, design thinkers, public accountants, lawyers, editors, and academics, whose job is to '*think for a living*.'"[1] Not bad. But not good enough to build a theory of cognitive work as useful as thermodynamics was for physical work.

Since the middle of the twentieth century, we have had mathematician Claude Shannon's quantification of information and an information theory with connections to and, in some respects, directly inspired by classical thermodynamics.[2] Intuitively, it seems safe to say that information is the precursor to knowledge. In some sense, building the bridge from the rigor of information theory to a useful theory of cognitive work has been one of the great challenges facing the discipline of AI since its founding in the summer of 1956. You can well imagine that our ancestors faced a similar quandary in the eighteenth and nineteenth centuries as they architected the industrialization of society. Sometimes the machines came before we really understood why they worked and the best way to build them, much less the complex network of social implications their construction and use entailed. But our ancestors built those machines anyway because it was blindingly obvious why they were useful.

In 2022, we have more clues about what a theory of cognitive work might be, although the theory itself may not be a new one. Of the ten attendees of the 1956 Dartmouth Summer Research Conference on Artificial Intelligence, which coined the term *artificial intelligence* and helped to establish AI as a discipline, Ray Solomonoff's name is less well-known than Marvin Minsky, John McCarthy, or Claude Shannon. Even though the subdiscipline of AI called machine learning has only in the past two decades taken over as the primary thrust of AI research and commercial activity, from the beginning, Solomonoff envisioned machine systems that could use probability and data to learn to solve complex problems.

Perhaps the most important of Solomonoff's insights was his theory of inductive inference. This theory is in some ways a resolution of tension between two ancient ideas: Occam's razor and Epicurus's principle of multiple explanations. We are probably all familiar with Occam's razor, which states that when faced with

a choice between multiple consistent explanations of an observed phenomenon, we should choose the simplest. Epicurus's principle, on the other hand, states that we should consider all consistent explanations.

Solomonoff's resolution, while mathematically quite sophisticated, is a relatively simple idea. You formulate the explanations of observable phenomena as programs for an abstract computing device, specifically a universal Turing machine.[3] The shorter a program is, the more concise it is at explaining observed phenomena. We can now use this conciseness as a precise measure of simplicity for Occam's razor. We then use the tools of Bayesian probability and a universal prior to compute the posterior probability of the range of computable explanations for any observed phenomenon.

When we train modern machine learning models, to be clear, we are not performing Solomonoff induction, which Solomonoff himself proved to be uncomputable. Regardless, Solomonoff induction is an interesting framework for thinking about cognitive work given that it is complete, at least over the universe of computable explanations.[4] Although I am biased by my computer science training, I would argue that it is not hard to imagine how you could explain almost any observable phenomenon by at least some arbitrarily long program. The beauty of Solomonoff induction is that, to quote Ilya Sutskever, chief scientist of OpenAI, "compression equals generalization." An incomprehensibly long explanation of a single phenomenon is nowhere near as powerful as a single concise description of many phenomena. Solomonoff induction gives us a framework for thinking very precisely about exactly this.

I may have just invoked too little theoretical computer science to frustrate the real theoretical computer scientists, and too much to frustrate everyone else, with the question still lingering: how does this help us understand cognitive work? Let us step back a moment to the work that we all do as knowledge workers. Much of our work involves the use of a bunch of cognitive tools that humans have developed over millennia, and frameworks for refining and composing these tools with one another that help us solve problems orders and orders of magnitude more complex than our ancestors could, even though biologically we are most certainly not orders of magnitude smarter. Our ability to refine these cognitive tools, to rigorously ensure that they work, and then to compose them may very well be the human version of "compression is generalization," the way for us to do more even though we likely have no more real cognitive capacity than the ancients.

Take two of these tools I am guessing that many of us use to do our work: mathematics and the scientific method. The modern body of mathematics that we learn in high school and university, and increasingly the computational tools that we use to support our mathematical activities, lets us reason about phenom-

ena we can neither see, touch, nor otherwise sense. Perhaps more important, it allows us to make predictions and reason about phenomena that have never actually occurred. With millennia-old mathematics, our ancestors could design aqueducts that supported sophisticated ancient civilizations by allowing them to move water around for irrigation, drinking, and sanitation. With twentieth- and twenty-first-century mathematics and computation, we can design lithographic structures on silicon wafers that move electrons around with near atomic-level precision. We carry devices made with these silicon artifacts in our pockets and backpacks that give us a way to connect and communicate with billions of other humans, access the world's knowledge, create our work, and engage in almost any form of commerce imaginable. To get from aqueducts to microprocessors, we have had to build a whole modern cognitive architecture composed of layers upon layers of cognitive tools that we and our predecessors have contributed to.

When I stop to think hard about the tools that I use to do my work, they do feel like an amazing compression algorithm that lets me get more mileage out of the brain I was born with. In computer science, this effect is hard to miss. The programs that I wrote as a young computing professional were longer and accomplished far less than the ones I write today. And the margin by which a line of code has become more powerful is far greater than the productivity I have gained through polishing my programming skills over the years. The tools that are available to me now are orders of magnitude more powerful than they were when I began coding in the 1980s. Moreover, whether you are an engineer, a scientist, a writer, or an artist, what has become clear over the past handful of years is that the AI systems that we are building today will likely have an equally momentous impact on the cognitive work that we are all able to do in the future.

In the same way that an engineer might assemble metal alloys, hydraulic pistons, electric motors, shafts, bearings, and electronics into a machine that performs mechanical work, like a forklift, engineers of AI systems increasingly rely upon deep neural networks (DNNs) to build software systems capable of performing cognitive work. In a real sense, the widespread use of DNNs today is made possible by large amounts of data and compute needed to train them. In 2009, machine learning scholar Andrew Ng and his colleagues at Stanford proposed the use of graphics processing units (GPUs) – devices capable of quickly and efficiently performing the sorts of arithmetic necessary for creating realistic video games – for training DNNs.[5] While Ng did not invent the DNN, his innovative use of the computational power of GPUs to train them helped to bring about a new age of machine learning with the DNN as its most powerful building block.

Over the past decade or so, the amount of compute used to train the DNN building blocks of our AI tools for cognitive work has increased exponentially. In 2018, OpenAI scientists noted that from 2012 to 2018, the amount of compute used

in the largest AI training runs had increased by a factor of three hundred thousand.[6] Why? In a world of diminishing returns from Moore's law, it certainly is not because compute is cheap. These investments only make sense insofar as scale makes DNNs better building blocks for doing cognitive work. And arguably they have, in two notable ways.

In the first half of the nineteenth century, mechanical engineer Claude-Louis Navier and physicist George Gabriel Stokes developed a set of partial differential equations to describe the motion of viscous fluids. The Navier-Stokes flow equations are, in my opinion, among the most beautiful in all of mathematics. They very concisely describe an enormous range of phenomena in hydraulics, aeronautics, oceanography, and atmospheric sciences. They inform everything from the design of the pipes carrying water to our homes, to the design of the aircraft that take us on holiday, to the weather forecasts we use to plan our days. The problem with these equations is that, when used to model extremely complex physical objects or environments, they can become extraordinarily expensive to solve. Prior to the advent of computers and numerical solvers for partial differential equations (PDEs), one could only model relatively simple systems with high fidelity, or complex systems only with simplifying assumptions to make the calculations feasible. Even now with extremely powerful supercomputers, certain problems that could benefit from high-fidelity solutions to Navier-Stokes are computationally infeasible.

That is, until recent work by a team of computer scientists at Caltech. Zongyi Li and colleagues devised a way to use deep neural networks to solve the Navier-Stokes PDEs up to three orders of magnitude faster, under some circumstances, than prior state-of-the-art solvers.[7] In my graduate research, I was often happy to improve the performance of a system by 5 percent. One thousand times more performance is, to torture an overused word, incredible.

The pattern that Li and his colleagues employed is one that is becoming increasingly widespread in the sciences. This is the first notable way in which models trained with large amounts of compute are becoming better building blocks for cognitive work. With an abundance of compute, DNNs can be trained using accurate but slow simulators or solvers for numerical, combinatorial, or even symbolic problems to encode something about the structure of a problem domain that we have yet to be able to model in other ways, such as through mathematics, or heuristics, or code. These DNNs can then be used to solve problems, allowing scientists to approach their work in new ways. Sometimes these techniques may make expensive things quicker or cheaper so that more people can solve more problems. Sometimes they may mean creating the ability to tackle problems so large or complex that they were previously impossible to solve. And the better news is that it seems as if this pattern is widely applicable and just beginning to be widely adopted. There is much to look forward to in the years to come.

The second way that scale is allowing us to construct better building blocks for performing cognitive work involves the use of self-supervised learning for building deep neural networks that behave as building blocks or platforms for a wide range of uses.

Before we dive into an explanation of self-supervised models, it is useful to understand a bit about supervised models, which drove much of the progress in the early years of the DNN boom. The first decade or more of machine learning systems that I built were all supervised. If you wanted to train a model to predict when someone was going to click on an ad, whether a piece of email was spam, or whether a picture contains an image of a kitten, you had to provide the supervised learning algorithms with lots and lots of examples of both good and bad ad clicks, spammy and nonspammy emails, or pictures with or without kittens in them. Providing those examples and counterexamples is an exercise called labeling and is time consuming and expensive given the volume of labeled training data required to achieve good performance.

For those of us following the field of machine learning closely, the last several years have brought extraordinary progress in solving problems related to human perception (recognizing the objects in images or the words spoken to a device), strategic game playing (beating the best human players at Go or Dota), and, most recently, in natural language understanding. The progress in natural language understanding began to accelerate in 2018 with the publication of a paper by Jacob Devlin, a software engineer at Google, which introduced the notion of *pretraining* for language models.[8] By now, this will feel like a familiar pattern. BERT, RoBERTa, DeBERTa, and other models use a set of techniques to learn the structure of natural language in a process that researchers in the field call pretraining.[9] Pretraining in these language models, like many of the most powerful contemporary deep learning systems, is self-supervised. In other words, the models learn without direct human supervision.

Once pretrained, a model, with the things it has learned about language structure, can be used to solve a wide range of problems in natural language processing. In many cases, a pretrained model needs to be fine-tuned to a particular task with some supervision. In some cases, the pretrained model itself is good enough. For instance, researchers at the Allen Institute for Artificial Intelligence used BERT in a test-taking system they had built called Aristo that was able to score higher than 90 percent on the multiple-choice component of the New York Regents eighth-grade science exam, and exceeded 83 percent on the twelfth-grade test.[10] My colleagues at Microsoft Research used their DeBERTa model to, for the first time, surpass the human baseline on the SuperGLUE benchmark, which entails solving nontrivial natural language problems, such as processing a complicated passage of text and then answering true or false questions about the passage, or

resolving the referent of a pronoun in an ambiguous passage of text. The best natural language models are now able to exceed expert human performance on these benchmarks.

When examining these systems, it is always important to ask: are these models capable of doing what they do because they have superhumanly big memories from which they recall the answers to problems someone else has solved, or have they compressed what they have seen in a way that lets them generalize solutions of problems no one has ever solved before? While there is ongoing debate about what, if anything, our contemporary self-supervised language models are "learning,"[11] for both those systems as well as those in which the data-fueling model learning is generated in simulation, it does seem that large data and compute are allowing us to encode useful things about problem domains that no human has previously encountered.

Perhaps the two most impressive recent illustrations of how large models trained with large compute can produce interesting results are OpenAI's Generative Pre-trained Transformer 3 (GPT-3) and Codex models.[12] At the time of its release, GPT-3 was ten times larger than the largest nonsparse language model. There are many benefits to scale, although perhaps the two most important are: when properly trained, larger models tend to have better performance on the same task than smaller models; and larger models tend to be useful in a broader range of tasks, either with fine-tuning or not, than smaller models. Because GPT-3 is useful on a broad range of tasks with little or no additional fine-tuning, it has been possible to offer an application programming interface to developers to allow them to probe the utility of the model for the problems they are interested in solving.

One of the biggest surprises of the GPT-3 model is that it generalized something about the structure of computer programming languages that allowed it to synthesize code that did not exist in its training data. This realization led to OpenAI fine-tuning a model for computer code called Codex, and in collaboration with GitHub, developing a coding assistant product called Copilot that can write code from natural language prompts. As the Codex model and the Copilot product get better, they will not only assist programmers with their cognitive work, but may also lower the barrier to entry to programming itself. Given that Codex and Copilot work by allowing humans to describe in natural language an effect they would like accomplished through code, the task of coding may become more approachable to many, many more people.

This ability to train on one set of data and to transfer what is learned to a broad range of tasks is called transfer learning. Transfer learning, perhaps more than anything else over the next few years, is likely to accelerate our progress on AI. It allows us to think about models as reusable building blocks, what I call *platform models*, and researchers at Stanford are calling *foundation*

models.[13] Moreover, based on the trends of the past few years, for transfer learning to work better, we will need bigger and more sophisticated models, which in turn will require more training compute.

AI systems designed to assist us with our cognitive work will no doubt continue to surprise us. I have been surprised so many times over the past two decades by what AI scientists and researchers have been able to accomplish that I have learned to heed the second half of Arthur C. Clarke's first law: When a distinguished but elderly scientist states that something is possible, they are almost certainly right. When they state that something is impossible, they are very probably wrong. Somewhere in the surprises of the future that await us, I am looking forward to systems that can help me to write my code, to sharpen my writing, to help me better manage the deluge of information I crave, and to assist me with the art and artifacts I make in my workshop. Hopefully, as our eighteenth- and nineteenth-century forebears did with physical work, we will also sharpen our definitions of cognitive work, develop new mechanisms for measuring it, and get better at constructing AI building blocks and tools to help us with these tasks. But more than anything, I look forward to what happens when folks who are more imaginative and creative than I am are able to incorporate new AI-based cognitive tools into their work, to make things that awe and inspire, and to solve those vexing problems that face society as we race forward to an ever more complicated future.

ABOUT THE AUTHOR

Kevin Scott is Chief Technology Officer and Executive Vice President of Technology and Research at Microsoft. He is the author of *Reprogramming the American Dream: From Rural America to Silicon Valley – Making AI Serve Us All* (2020), is a coinventor on multiple patents, and is host of the podcast *Behind the Tech*.

ENDNOTES

[1] "Knowledge Worker," Wikipedia, last updated October 5, 2021, https://en.wikipedia.org/wiki/Knowledge_worker (accessed January 27, 2022; emphasis added).

[2] Claude Shannon and Warren Weaver, *A Mathematical Theory of Communication* (Champaign: University of Illinois Press, 1949).

[3] Alan Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society* (1936): 230–265.

[4] I am deliberately avoiding the use of the term *intelligence* given that its use is so burdened with imprecision and poor analogy.

[5] Rajat Raina, Anand Madhavan, and Andrew Y. Ng, "Large-Scale Deep Unsupervised Learning Using Graphics Processors," in *Proceedings of the 26th International Conference on Machine Learning* (New York: Association for Computing Machinery, 2009).

[6] Dario Amodei and Danny Hernandez, "AI and Compute," OpenAI, May 16, 2018, https://openai.com/blog/ai-and-compute/.

[7] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, et al., "Fourier Neural Operator for Parametric Partial Differential Equations," presented at the 9th International Conference on Learning Representations, virtual event, May 3–7, 2021.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv (2020), https://arxiv.org/pdf/1810.04805.pdf.

[9] Yinhan Liu, Myle Ott, Naman Goyal, et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv (2019), https://arxiv.org/pdf/1907.11692.pdf; and Pencheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen, "DeBERTa: Decoding-Enhanced BERT with Disentangled Attention," arXiv (2020), https://arxiv.org/pdf/2006.03654.pdf.

[10] Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, et al., "From 'F' to 'A' on the N.Y. Regents Science Exams: An Overview of the Aristo Project," arXiv (2021), https://arxiv.org/pdf/1909.01958.pdf.

[11] Emily Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell, "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜" in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (New York: Association of Computing Machinery, 2021).

[12] Tom Brown, Benjamin Mann, Nick Ryder, et al., "Language Models Are Few-Shot Learners," arXiv (2020), https://arxiv.org/pdf/2005.14165v4.pdf; and Mark Chen, Jerry Tworek, and Heewoo Jun, "Evaluating Large Language Models Trained on Code," arXiv (2021), https://arxiv.org/abs/2107.03374.

[13] Rishi Bommasani, Drew A. Hudson, and Ehsan Adeli, "On the Opportunities and Risks of Foundation Models," arXiv (2021), https://arxiv.org/abs/2108.07258v1.