CHAPTER 29

THE FUTURE OF AI

In which we try to see a short distance ahead.

In Chapter 2, we decided to view AI as the task of designing approximately rational agents. A variety of different agent designs were considered, ranging from reflex agents to knowledge-based decision-theoretic agents to deep learning agents using reinforcement learning. There is also variety in the component technologies from which these designs are assembled: logical, probabilistic, or neural reasoning; atomic, factored, or structured representations of states; various learning algorithms from various types of data; sensors and actuators to interact with the world. Finally, we have seen a variety of applications, in medicine, finance, transportation, communication, and other fields. There has been progress on all these fronts, both in our scientific understanding and in our technological capabilities.

Most experts are optimistic about continued progress; as we saw on page 46, the median estimate is for approximately human-level AI across a broad variety of tasks somewhere in the next 50 to 100 years. Within the next decade, AI is predicted to add trillions of dollars to the economy each year. But as we also saw, there are some critics who think general AI is centuries off, and there are numerous ethical concerns about the fairness, equity, and lethality of AI. In this chapter, we ask: where are we headed and what remains to be done? We do that by asking whether we have the right components, architectures, and goals to make AI a successful technology that delivers benefits to the world.

29.1 AI Components

This section examines the components of AI systems and the extent to which each of them might accelerate or hinder future progress.

Sensors and actuators

For much of the history of AI, direct access to the world has been glaringly absent. With a few notable exceptions, AI systems were built in such a way that humans had to supply the inputs and interpret the outputs. Meanwhile, robotic systems focused on low-level tasks in which high-level reasoning and planning were largely ignored and the need for perception was minimized. This was partly due to the great expense and engineering effort required to get real robots to work at all, and partly because of the lack of sufficient processing power and sufficiently effective algorithms to handle high-bandwidth visual input.

The situation has changed rapidly in recent years with the availability of ready-made programmable robots. These, in turn, have benefited from compact reliable motor drives and improved sensors. The cost of lidar for a self-driving car has fallen from $75,000 to $1,000,

and a single-chip version may reach $10 per unit (Poulton and Watts, 2016). Radar sensors, once capable of only coarse-grained detection, are now sensitive enough to count the number of sheets in a stack of paper (Yeo *et al.*, 2018).

The demand for better image processing in cellphone cameras has given us inexpensive high-resolution cameras for use in robotics. MEMS (micro-electromechanical systems) technology has supplied miniaturized accelerometers, gyroscopes, and actuators small enough to fit in artificial flying insects (Floreano *et al.*, 2009; Fuller *et al.*, 2014). It may be possible to combine millions of MEMS devices to produce powerful macroscopic actuators. 3-D printing (Muth *et al.*, 2014) and bioprinting (Kolesky *et al.*, 2014) have made it easier to experiment with prototypes.

Thus, we see that AI systems are at the cusp of moving from primarily software-only systems to useful embedded robotic systems. The state of robotics today is roughly comparable to the state of personal computers in the early 1980s: at that time personal computers were becoming available, but it would take another decade before they became commonplace. It is likely that flexible, intelligent robots will first make strides in industry (where environments are more controlled, tasks are more repetitive, and the value of an investment is easier to measure) before the home market (where there is more variability in environment and tasks).

Representing the state of the world

Keeping track of the world requires perception as well as updating of internal representations. Chapter 4 showed how to keep track of atomic state representations; Chapter 7 described how to do it for factored (propositional) state representations; Chapter 10 extended this to first-order logic; and Chapter 14 described probabilistic reasoning over time in uncertain environments. Chapter 22 introduced recurrent neural networks, which are also capable of maintaining a state representation over time.

Current filtering and perception algorithms can be combined to do a reasonable job of recognizing objects ("that's a cat") and reporting low-level predicates ("the cup is on the table"). Recognizing higher-level actions, such as "Dr. Russell is having a cup of tea with Dr. Norvig while discussing plans for next week," is more difficult. Currently it can sometimes be done (see Figure 27.17 on page 1015) given enough training examples, but future progress will require techniques that generalize to novel situations without requiring exhaustive examples (Poppe, 2010; Kang and Wildes, 2016).

Another problem is that although the approximate filtering algorithms from Chapter 14 can handle quite large environments, they are still dealing with a factored representation—they have random variables, but do not represent objects and relations explicitly. Also, their notion of time is restricted to step-by-step change; given the recent trajectory of a ball, we can predict where it will be at time $t + 1$, but it is difficult to represent the abstract idea that what goes up must come down.

Section 18.1 explained how probability and first-order logic can be combined to solve these problems; Section 18.2 showed how we can handle uncertainty about the identity of objects; and Chapter 27 showed how recurrent neural networks enable computer vision to track the world; but we don't yet have a good way of putting all these techniques together. Chapter 25 showed how word embeddings and similar representations can free us from the strict bounds of concepts defined by necessary and sufficient conditions. It remains a daunting task to define general, reusable representation schemes for complex domains.

Selecting actions

The primary difficulty in action selection in the real world is coping with long-term plans—such as graduating from college in four years—that consist of billions of primitive steps. Search algorithms that consider sequences of primitive actions scale only to tens or perhaps hundreds of steps. It is only by imposing **hierarchical structure** on behavior that we humans cope at all. We saw in Section 11.4 how to use hierarchical representations to handle problems of this scale; furthermore, work in **hierarchical reinforcement learning** has succeeded in combining these ideas with the MDP formalism described in Chapter 16.

As yet, these methods have not been extended to the partially observable case (POMDPs). Moreover, algorithms for solving POMDPs are typically using the same atomic state representation we used for the search algorithms of Chapter 3. There is clearly a great deal of work to do here, but the technical foundations are largely in place for making progress. The main missing element is an effective method for *constructing* the hierarchical representations of state and behavior that are necessary for decision making over long time scales.

Deciding what we want

Chapter 3 introduced search algorithms to find a goal state. But goal-based agents are brittle when the environment is uncertain, and when there are multiple factors to consider. In principle, utility-maximization agents address those issues in a completely general way. The fields of economics and game theory, as well as AI, make use of this insight: just declare what you want to optimize, and what each action does, and we can compute the optimal action.

In practice, however, we now realize that the task of picking the right utility function is a challenging problem in its own right. Imagine, for example, the complex web of interacting preferences that must be understood by an agent operating as an office assistant for a human being. The problem is exacerbated by the fact that each human is different, so an agent just "out of the box" will not have enough experience with any one individual to learn an accurate preference model; it will necessarily need to operate under preference uncertainty. Further complexity arises if we want to ensure that our agents are acting in a way that is fair and equitable for society, rather than just one individual.

We do not yet have much experience with building complex real-world preference models, let alone probability distributions over such models. Although there are factored formalisms, similar to Bayes nets, that are intended to decompose preferences over complex states, it has proven difficult to use these formalisms in practice. One reason may be that preferences over states are really *compiled* from preferences over state histories, which are described by **reward functions** (see Chapter 16). Even if the reward function is simple, the corresponding utility function may be very complex.

This suggests that we take seriously the task of knowledge engineering for reward functions as a way of conveying to our agents what we want them to do. The idea of **inverse reinforcement learning** (Section 23.6) is one approach to this problem when we have an expert who can perform a task, but not explain it. We could also use better languages for expressing what we want. For example, in robotics, linear temporal logic makes it easier to say what things we want to happen in the near future, what things we want to avoid, and what states we want to persist forever (Littman *et al.*, 2017). We need better ways of saying what we want and better ways for robots to interpret the information we provide.

The computer industry as a whole has developed a powerful ecosystem for aggregating user preferences. When you click on something in an app, online game, social network, or shopping site, that serves as a recommendation that you (and your similar peers) would like to see similar things in the future. (Or it might be that the site is confusing and you clicked on the wrong thing—the data are always noisy.) The feedback inherent in this system makes it very effective in the short run for picking out ever more addictive games and videos.

But these systems often fail to provide an easy way of opting out—your device will autoplay a relevant video, but it is less likely to tell you "maybe it is time to put away your devices and take a relaxing walk in nature." A shopping site will help you find clothes that match your style, but will not address world peace or ending hunger and poverty. To the extent that the menu of choices is driven by companies trying to profit from a customer's attention, the menu will remain incomplete.

However, companies do respond to customers' interests, and many customers have voiced the opinion that they are interested in a fair and sustainable world. Tim O'Reilly explains why profit is not the only motive with the following analogy: "Money is like gasoline during a road trip. You don't want to run out of gas on your trip, but you're not doing a tour of gas stations. You have to pay attention to money, but it shouldn't be about the money."

Time well spent

Tristan Harris's **time well spent** movement at the Center for Humane Technology is a step towards giving us more well-rounded choices (Harris, 2016). The movement addresses an issue that was recognized by Herbert Simon in 1971: "A wealth of information creates a poverty of attention." Perhaps in the future we will have **personal agents** that stick up for our true long-term interests rather than the interests of the corporations whose apps currently fill our devices. It will be the agent's job to mediate the offerings of various vendors, protect us from addictive attention-grabbers, and guide us towards the goals that really matter to us.

Personal agent

Learning

Chapters 19, 21, 22, and 23 described how agents can learn. Current algorithms can cope with quite large problems, reaching or exceeding human capabilities in many tasks—as long as we have sufficient training examples and we are dealing with a predefined vocabulary of features and concepts. But learning can stall when data are sparse, or unsupervised, or when we are dealing with complex representations.

Much of the recent resurgence of AI in the popular press and in industry is due to the success of deep learning (Chapter 22). On the one hand, this can be seen as the incremental maturation of the subfield of neural networks. On the other hand, we can see it as a revolutionary leap in capabilities spurred by a confluence of factors: the availability of more training data thanks to the Internet, increased processing power from specialized hardware, and a few algorithmic tricks, such as generative adversarial networks (GANs), batch normalization, dropout, and the rectified linear (ReLU) activation function.

The future should see continued emphasis on improving deep learning for the tasks it excels at, and also extending it to cover other tasks. The brand name "deep learning" has proven to be so popular that we should expect its use to continue, even if the mix of techniques that fuel it changes considerably.

We have seen the emergence of the field of **data science** as the confluence of statistics, programming, and domain expertise. While we can expect to see continued development in the tools and techniques necessary to acquire, manage, and maintain **big data**, we will also

need advances in **transfer learning** so that we can take advantage of data in one domain to improve performance on a related domain.

The vast majority of machine learning research today assumes a factored representation, learning a function $h : \mathbb{R}^n \to \mathbb{R}$ for regression and $h : \mathbb{R}^n \to \{0, 1\}$ for classification. Machine learning has been less successful for problems that have only a small amount of data, or problems that require the construction of new structured, hierarchical representations. Deep learning, especially with convolutional networks applied to computer vision problems, has demonstrated some success in going from low-level pixels to intermediate-level concepts like *Eye* and *Mouth*, then to *Face*, and finally to *Person* or *Cat*.

A challenge for the future is to more smoothly combine learning and prior knowledge. If we give a computer a problem it has not encountered before—say, recognizing different models of cars—we don't want the system to be powerless until it has been fed millions of labeled examples.

The ideal system should be able to draw on what it already knows: it should already have a model of how vision works, and how the design and branding of products in general work; now it should use **transfer learning** to apply that to the new problem of car models. It should be able to find on its own information about car models, drawing from text, images, and video available on the Internet. It should be capable of **apprenticeship learning**: having a conversation with a teacher, and not just asking "may I have a thousand images of a Corolla," but rather being able to understand advice like "the Insight is similar to the Prius, but the Insight has a larger grille." It should know that each model comes in a small range of possible colors, but that a car can be repainted, so there is a chance that it might see a car in a color that was not in the training set. (If it didn't know that, it should be capable of learning it, or being told about it.)

All this requires a communication and representation language that humans and computers can share; we can't expect a human analyst to directly modify a model with millions of weights. Probabilistic models (including probabilistic programming languages) give humans some ability to describe what we know, but these models are not yet well integrated with other learning mechanisms.

The work of Bengio and LeCun (2007) is one step towards this integration. Recently Yann LeCun has suggested that the term "deep learning" should be replaced with the more general **differentiable programming** (Siskind and Pearlmutter, 2016; Li *et al.*, 2018); this suggests that our general programming languages and our machine learning models could be merged together.

Differentiable programming

Right now, it is common to build a deep learning model that is differentiable, and thus can be trained to minimize loss, and retrained when circumstances change. But that deep learning model is only one part of a larger software system that takes in data, massages the data, feeds it to the model, and figures out what to do with the model's output. All these other parts of the larger system were written by hand by a programmer, and thus are nondifferentiable, which means that when circumstances change, it is up to the programmer to recognize any problems and fix them by hand. With differentiable programming, the hope is that the entire system is subject to automated optimization.

The end goal is to be able to express what we know in whatever form is convenient to us: informal advice given in natural language, a strong mathematical law like $F = ma$, a statistical model accompanied by data, or a probabilistic program with unknown parameters that can

be automatically optimized through gradient descent. Our computer models will learn from conversations with human experts as well as by using all the available data.

Yann LeCun, Geoffrey Hinton, and others have suggested that the current emphasis on supervised learning (and to a lesser extent reinforcement learning) is not sustainable—that computer models will have to rely on **weakly supervised learning**, in which some supervision is given with a small number of labeled examples and/or a small number of rewards, but most of the learning is unsupervised, because unannotated data are so much more plentiful.

Predictive learning

LeCun uses the term **predictive learning** for an unsupervised learning system that can model the world and learn to predict aspects of future states of the world—not just predict labels for inputs that are independent and identically distributed with respect to past data, and not just predict a value function over states. He suggests that GANs (generative adversarial networks) can be used to learn to minimize the difference between predictions and reality.

Geoffrey Hinton stated in 2017 that "My view is throw it all away and start again," meaning that the overall idea of learning by adjusting parameters in a network is enduring, but the specifics of the architecture of the networks and the technique of back-propagation need to be rethought. Smolensky (1988) had a prescription for how to think about connectionist models; his thoughts remain relevant today.

Resources

Machine learning research and development has been accelerated by the increasing availability of data, storage, processing power, software, trained experts, and the investments needed to support them. Since the 1970s, there has been a 100,000-fold speedup in general-purpose processors and an additional 1,000-fold speedup due to specialized machine learning hardware. The Web has served as a rich source of images, videos, speech, text, and semi-structured data, currently adding over 10^{18} bytes every day.

Hundreds of high-quality data sets are available for a range of tasks in computer vision, speech recognition, and natural language processing. If the data you need is not already available, you can often assemble it from other sources, or engage humans to label data for you through a crowdsourcing platform. Validating the data obtained in this way becomes an important part of the overall workflow (Hirth *et al.*, 2013).

Shared model

An important recent development is the shift from shared data to **shared models**. The major cloud service providers (e.g., Amazon, Microsoft, Google, Alibaba, IBM, Salesforce) have begun competing to offer machine learning APIs with pre-built models for specific tasks such as visual object recognition, speech recognition, and machine translation. These models can be used as is, or can serve as a baseline to be customized with your particular data for your particular application.

We expect that these models will improve over time, and that it will become unusual to start a machine learning project from scratch, just as it is now unusual to do a Web development project from scratch, with no libraries. It is possible that a big jump in model quality will occur when it becomes economical to process all the video on the Web; for example, the YouTube platform alone adds 300 hours of video every minute.

Moore's law has made it more cost effective to process data; a megabyte of storage cost $1 million in 1969 and less then $0.02 in 2019, and supercomputer throughput has increased by a factor of more than 10^{10} in that time. Specialized hardware components for machine learning such as graphics processing units (GPUs), tensor cores, tensor processing units (TPUs), and

field programmable gate arrays (FPGAs) are hundreds of times faster than conventional CPUs for machine learning training (Vasilache *et al.*, 2014; Jouppi *et al.*, 2017). In 2014 it took a full day to train an ImageNet model; in 2018 it takes just 2 minutes (Ying *et al.*, 2018).

The OpenAI Institute reports that the amount of compute power used to train the largest machine learning models doubled every 3.5 months from 2012 to 2018, reaching over an exaflop/second-day for ALPHAZERO (although they also report that some very influential work used 100 million times less computing power (Amodei and Hernandez, 2018)). The same economic trends that have made cell-phone cameras cheaper and better also apply to processors—we will see continued progress in low-power, high-performance computing that benefits from economies of scale.

There is a possibility that quantum computers could accelerate AI. Currently there are some fast quantum algorithms for the linear algebra operations used in machine learning (Harrow *et al.*, 2009; Dervovic *et al.*, 2018), but no quantum computer capable of running them. We have some example applications of tasks such as image classification (Mott *et al.*, 2017) where quantum algorithms are as good as classical algorithms on small problems.

Current quantum computers handle only a few tens of bits, whereas machine learning algorithms often handle inputs with millions of bits and create models with hundreds of millions of parameters. So we need breakthroughs in both quantum hardware and software to make quantum computing practical for large-scale machine learning. Alternatively, there may be a division of labor—perhaps a quantum algorithm to efficiently search the space of hyperparameters while the normal training process runs on conventional computers—but we don't know how to do that yet. Research on quantum algorithms can sometimes inspire new and better algorithms on classical computers (Tang, 2018).

We have also seen exponential growth in the number of publications, people, and dollars in AI/machine learning/data science. Dean *et al.* (2018) show that the number of papers about "machine learning" on arXiv doubled every two years from 2009 to 2017. Investors are funding startup companies in these fields, large companies are hiring and spending as they determine their AI strategy, and governments are investing to make sure their country doesn't fall too far behind.

29.2 AI Architectures

It is natural to ask, "Which of the agent architectures in Chapter 2 should an agent use?" The answer is, "All of them!" Reflex responses are needed for situations in which time is of the essence, whereas knowledge-based deliberation allows the agent to plan ahead. Learning is convenient when we have lots of data, and necessary when the environment is changing, or when human designers have insufficient knowledge of the domain.

AI has long had a split between symbolic systems (based on logical and probabilistic inference) and connectionist systems (based on loss minimization over a large number of uninterpreted parameters). A continuing challenge for AI is to bring these two together, to capture the best of both. Symbolic systems allow us to string together long chains of reasoning and to take advantage of the expressive power of structured representations, while connectionist systems can recognize patterns even in the face of noisy data. One line of research aims to combine probabilistic programming with deep learning, although as yet the various proposals are limited in the extent to which the approaches are truly merged.

Agents also need ways to control their own deliberations. They must be able to use the available time well, and cease deliberating when action is demanded. For example, a taxi-driving agent that sees an accident ahead must decide in a split second whether to brake or swerve. It should also spend that split second thinking about the most important questions, such as whether the lanes to the left and right are clear and whether there is a large truck close behind, rather than worrying about where to pick up the next passenger. These issues are usually studied under the heading of **real-time AI**. As AI systems move into more complex domains, all problems will become real-time, because the agent will never have long enough to solve the decision problem exactly.

Real-time AI

Clearly, there is a pressing need for *general* methods of controlling deliberation, rather than specific recipes for what to think about in each situation. The first useful idea is the **anytime algorithms** (Dean and Boddy, 1988; Horvitz, 1987): an algorithm whose output quality improves gradually over time, so that it has a reasonable decision ready whenever it is interrupted. Examples of anytime algorithms include iterative deepening in game-tree search and MCMC in Bayesian networks.

Anytime algorithm

The second technique for controlling deliberation is **decision-theoretic metareasoning** (Russell and Wefald, 1989; Horvitz and Breese, 1996; Hay *et al.*, 2012). This method, which was mentioned briefly in Sections 3.6.5 and 6.7, applies the theory of information value (Chapter 15) to the selection of individual computations (Section 3.6.5). The value of a computation depends on both its cost (in terms of delaying action) and its benefits (in terms of improved decision quality).

Decision-theoretic metareasoning

Metareasoning techniques can be used to design better search algorithms and to guarantee that the algorithms have the anytime property. Monte Carlo tree search is one example: the choice of leaf node at which to begin the next playout is made by an approximately rational metalevel decision derived from bandit theory.

Metareasoning is more expensive than reflex action, of course, but compilation methods can be applied so that the overhead is small compared to the costs of the computations being controlled. Metalevel reinforcement learning may provide another way to acquire effective policies for controlling deliberation: in essence, computations that lead to better decisions are reinforced, while those that turn out to have no effect are penalized. This approach avoids the myopia problems of the simple value-of-information calculation.

Metareasoning is one specific example of a **reflective architecture**—that is, an architecture that enables deliberation about the computational entities and actions occurring within the architecture itself. A theoretical foundation for reflective architectures can be built by defining a joint state space composed from the environment state and the computational state of the agent itself. Decision-making and learning algorithms can be designed that operate over this joint state space and thereby serve to implement and improve the agent's computational activities. Eventually, we expect task-specific algorithms such as alpha–beta search, regression planning, and variable elimination to disappear from AI systems, to be replaced by general methods that direct the agent's computations toward the efficient generation of high-quality decisions.

Reflective architecture

Metareasoning and reflection (and many other efficiency-related architectural and algorithmic devices explored in this book) are necessary because making decisions is *hard*. Ever since computers were invented, their blinding speed has led people to overestimate their ability to overcome complexity, or, equivalently, to underestimate what complexity really means.

The truly gargantuan power of today's machines tempts one to think that we could bypass all the clever devices and rely more on brute force. So let's try to counteract this tendency. We begin with what physicists believe to be the speed of the ultimate 1kg computing device: about 10^{51} operations per second, or a billion trillion trillion times faster than the fastest supercomputer as of 2020 (Lloyd, 2000).[1] Then we propose a simple task: enumerating strings of English words, much as Borges proposed in *The Library of Babel*. Borges stipulated books of 410 pages. Would that be feasible? Not quite. In fact, the computer running for a year could enumerate only the 11-word strings.

Now consider the fact that a detailed plan for a human life consists of (very roughly) twenty trillion potential muscle actuations (Russell, 2019), and you begin to see the scale of the problem. A computer that is a billion trillion trillion times more powerful than the human brain is much further from being rational than a slug is from overtaking the starship Enterprise traveling at warp nine.

With these considerations in mind, it seems that the goal of building rational agents is perhaps a little too ambitious. Rather than aiming for something that cannot possibly exist, we should consider a different normative target—one that *necessarily* exists. Recall from Chapter 2 the following simple idea:

$$agent = architecture + program\,.$$

Now fix the agent architecture (the underlying machine capabilities, perhaps with a fixed software layer on top) and allow the agent program to vary over all possible programs that the architecture can support. In any given task environment, one of these programs (or an equivalence class of them) delivers the best possible performance—perhaps not close to perfect rationality, but still better than any other agent program. We say that this program satisfies the criterion of **bounded optimality**. Clearly it exists, and clearly it constitutes a desirable goal. The trick is finding it, or something close to it.

Bounded optimality

For some elementary classes of agent programs in simple real-time environments, it is possible to identify bounded-optimal agent programs (Etzioni, 1989; Russell and Subramanian, 1995). The success of Monte Carlo tree search has revived interest in metalevel decision making, and there is reason to hope that bounded optimality within more complex families of agent programs can be achieved by techniques such as metalevel reinforcement learning. It should also be possible to develop a constructive theory of architecture, beginning with theorems on the bounded optimality of suitable methods of combining different bounded-optimal components such as reflex and action–value systems.

General AI

Much of the progress in AI in the 21st century so far has been guided by competition on narrow tasks, such as the DARPA Grand Challenge for autonomous cars, the ImageNet object recognition competition, or playing Go, chess, poker, or Jeopardy! against a world champion. For each separate task, we build a separate AI system, usually with a separate machine learning model trained from scratch with data collected specifically for this task. But a truly intelligent agent should be able to do more than one thing. Alan Turing (1950) proposed his list (page 1033) and science fiction author Robert Heinlein (1973) countered with:

[1] We gloss over the fact that this device consumes the entire energy output of a star and operates at a billion degrees centigrade.

> A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyse a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects.

So far, no AI system measures up to either of these lists, and some proponents of general or human-level AI (HLAI) insist that continued work on specific tasks (or on individual components) will not be enough to reach mastery on a wide variety of tasks; that we will need a fundamentally new approach. It seems to us that numerous new breakthroughs will indeed be necessary, but overall, AI as a field has made a reasonable exploration/exploitation tradeoff, assembling a portfolio of components, improving on particular tasks, while also exploring promising and sometimes far-out new ideas.

It would have been a mistake to tell the Wright brothers in 1903 to stop work on their single-task airplane and design an "artificial general flight" machine that can take off vertically, fly faster than sound, carry hundreds of passengers, and land on the moon. It also would have been a mistake to follow up their first flight with an annual competition to make spruce wood biplanes incrementally better.

We have seen that work on components can spur new ideas; for example, generative adversarial networks (GANs) and transformer language models each opened up new areas of research. We have also seen steps towards "diversity of behaviour." For example, machine translation systems in the 1990s were built one at a time for each language pair (such as French to English), but today a single system can identifying the input text as being one of a hundred languages, and translate it into any of 100 target languages. Another natural language system can perform five distinct tasks with one joint model (Hashimoto *et al.*, 2016).

AI engineering

The field of computer programming started with a few extraordinary pioneers. But it didn't reach the status of a major industry until a practice of software engineering was developed, with a powerful collection of widely available tools, and a thriving ecosystem of teachers, students, practitioners, entrepreneurs, investors, and customers.

The AI industry has not yet reached that level of maturity. We do have a variety of powerful tools and frameworks, such as TensorFlow, Keras, PyTorch, CAFFE, Scikit-Learn and SCIPY. But many of the most promising approaches, such as GANs and deep reinforcement learning, have proven to be difficult to work with—they require experience and a degree of fiddling to get them to train properly in a new domain. We don't have enough experts to do this across all the domains where we need it, and we don't yet have the tools and ecosystem to let less-expert practitioners succeed.

Google's Jeff Dean sees a future where we will want machine learning to handle millions of tasks; it won't be feasible to develop each of them from scratch, so he suggests that rather than building each new system from scratch, we should start with a single huge system and, for each new task, extract from it the parts that are relevant to the task. We have seen some steps in this direction, such as the transformer language models (e.g., BERT, GPT-2) with billions of parameters, and an "outrageously large" ensemble neural network architecture that scales up to 68 billion parameters in one experiment (Shazeer *et al.*, 2017). Much work remains to be done.

The future

Which way will the future go? Science fiction authors seem to favor dystopian futures over utopian ones, probably because they make for more interesting plots. So far, AI seems to fit in with other powerful revolutionary technologies such as printing, plumbing, air travel, and telephony. All these technologies have made positive impacts, but also have some unintended side effects that disproportionately impact disadvantaged classes. We would do well to invest in minimizing the negative impacts.

AI is also different from previous revolutionary technologies. Improving printing, plumbing, air travel, and telephony to their logical limits would not produce anything to threaten human supremacy in the world. Improving AI to its logical limit certainly could.

In conclusion, AI has made great progress in its short history, but the final sentence of Alan Turing's (1950) essay on *Computing Machinery and Intelligence* is still valid today:

> *We can see only a short distance ahead,*
> *but we can see that much remains to be done.*