Lesson: Artificial Intelligence Beam

From the earliest days of computing, humans have been fascinated by the prospect of constructing intelligent machines. This lesson introduces the field of artificial intelligence. **Artificial intelligence** (AI) is a branch of computer science that has as its goal the construction of a general-purpose intelligence (i.e., of constructing machines that are capable of doing all of the things at which, at the present time, people are better). Some of the topics associated with the field of AI include creating machines that can see and recognize objects in the real world, including people; converse fluently in human languages, such as English, and translate between different human languages; safely drive a car; plan and reason as people do; play strategy games like chess or Go and knowledge games like Jeopardy!; and display creativity and discover new knowledge. Essentially, AI seeks to make computers more like people.

An overview of artificial intelligence

The goal of artificial intelligence is to construct a general-purpose intelligence; that is, a machine that can interact with a human, more or less, as an equal. What capabilities would a machine have to possess before it could be considered intelligent? Is it possible to construct such a machine? Could we even tell if such machines were truly intelligent?

Let's begin by describing the capabilities of a fictional intelligent computer, the HAL 9000, in order to examine the features commonly associated with the notion of intelligent machines. Many people consider HAL, the fictional computer in Stanley Kubrick and Arthur C. Clarke's 1968 movie "2001: A Space Odyssey," to be the best depiction of an intelligent machine ever produced. The movie was groundbreaking at the time and much of what it has to say about machine intelligence is still relevant nearly half a century later. Here is a portion of a conversation between Dave and HAL. As you read this dialog, think about how very different the way Dave interacts with HAL is compared to the way we communicate with our computers, smart phones, and tablets today.

HAL: Good evening, Dave.
DAVE: How ya' doin' Hal?
HAL: Everything's running smoothly. And you?
DAVE: Oh, not too bad.
HAL: Have you been doing some more work?
DAVE: A few sketches.
HAL: May I see them?
DAVE: Sure.
HAL: That's a very nice rendering, Dave. I think you've improved a great deal. Can you hold it a bit closer?
DAVE: Sure.
HAL: That's Doctor Hunter isn't it?
DAVE: uh hum.
HAL: By the way, do you mind if I ask you a personal question?
DAVE: No, not at all.

Those of you who have seen the movie will know that HAL is the computer that serves as the "brain and central nervous system" of the Discovery spaceship. HAL is capable of many feats that surpass even the

most advanced computers today. For example, the astronauts aboard Discovery interact with HAL by simply talking to "him" (i.e., he is fully capable of understanding and speaking English).

Computer scientists call this ability to understand human languages, both written and spoken, **natural language understanding**. The ability to use language effectively is considered by computer scientists to be one of the most important characteristics of an intelligent entity. Constructing a machine with this ability has proven to be a tremendously difficult challenge.

Let's consider for a moment what it takes to understand spoken language. The first problem is just trying to identify the individual words in an utterance. When we humans speak, we tend to run all of our words together into one continuous stream. We don't usually think about this since we "hear" the individual words. That is, until we try to learn a foreign language.

Compounding the word identification problem is the fact that no two people will pronounce the same word exactly the same way. Even if we limit ourselves to constructing a system to recognize the voice of a single speaker, the sound that represents a word will still change depending on all sorts of factors. For example, what other words surround it, the emphasis the speaker wishes to place on it, and even whether the speaker is ill or has been drinking. There is also the problem of separating out background noise, deciding what portions of an incoming audio stream represent voice and which represent other sounds.

Once the individual words have been recognized, the thorny issue of what they actually mean must be addressed. This problem is constrained somewhat by the structure, or syntax, of the language being spoken (e.g., verb verb is not a valid sentence form in English). Nonetheless, the problem remains extraordinarily challenging, primarily because a great deal of common sense knowledge is frequently required to understand the meaning of even the simplest of utterances. Think about the knowledge HAL needs of informal English usage and human relations, specifically greetings and appropriate ways to respond, in order to make sense of Dave's "How ya' doin' Hal?"

In addition to being able to communicate fluently in English, HAL could also see what the crew was up to through cameras that were located throughout the ship. HAL could recognize individual crew members by their appearance. For example, in the dialog shown above, when Dave walks by one of HAL's consoles, HAL recognizes him and says "Good evening, Dave." Also, when Dave shows HAL his sketchpad, HAL is able to recognize the line drawings as artwork and interpret the images as representations of the people Dave had drawn.

Computer vision is the sub-discipline of artificial intelligence that seeks to construct machines that can see. This field consists of much more than simply connecting digital cameras to computers; it concerns trying to develop ways of automating the process of image understanding. To illustrate, most smart phones include built-in cameras which we use to take photos, record movies, and engage in Skype or FaceTime video chats. While these cameras are very useful, they don't, in general, give our smart phones the ability to "see" in the sense that HAL could or that people can. While our phones are capable of recording and displaying visual information, and can transmit and receive images and video over the Internet, they have no understanding of the meaning of those images.

Throughout the movie, HAL displays other behaviors that are associated with intelligence. He is able to play games such as chess. He can recognize problems (such as potential threats to his mission), plan

ways to solve those problems, implement those plans, and then learn from his mistakes. Problem solving, planning, and learning are all areas of research in artificial intelligence. The idea of constructing computers with these attributes causes some people concern, due to the fact that computers are often used in life critical situations (such as monitoring patients in intensive care units) and that these three features (problem solving, planning, and learning) require computers to have some degree of independence from humans.

What if a computer were to misunderstand an instruction and seriously injure or kill someone as a result? For example, perhaps an artificially intelligent computer in a health care facility has among its goals searching for ways to decrease patient suffering, to decrease the number of days a patient spends in ICU, and to limit their medical costs as much as possible consistent with its other goals. It would be important to make sure the system firmly understood that cutting off a terminal patient's oxygen supply was not an acceptable course of action, even though doing so would accomplish all three of the stated goals.

There are a number of other characteristics that HAL displays which are harder to quantify yet are of vital importance to humans. HAL appears to be self-aware and has a "life philosophy" which he sums up in a BBC interview as: "I am putting myself to the fullest possible use, which is all I think that any conscious entity can ever hope to do." HAL also exhibits emotions, such as pride and fear.

Can intelligent machines be constructed?

Is it possible for a machine to be intelligent? If one discounts the mystical, then the answer to this question is certainly yes, since humans are themselves biological "machines" (i.e., machines that are self-aware). Knowing that intelligent machines exist does not answer the question of whether humans can construct them. While most computer scientists think that constructing machines capable of intelligent behavior is possible, it is conceivable that we humans may simply be too dumb to ever figure out how to build an intelligent machine.

So, how should we go about trying to construct an artificial intelligence? There are two general approaches taken by AI researchers in their attempts to automate intelligent behavior. One approach is often called machine learning, while the other approach is known as symbolic AI.

Symbolic AI focuses on developing systems composed of explicit rules that manipulate data in ways designed to produce seemingly intelligent behavior. The underlying rules are designed by people and encoded into computer programs. Symbolic AI is the "classic" approach to artificial intelligence. Much of the early success in the field, with topics such as game playing, automated reasoning, and expert systems, resulted from this approach. Symbolic AI has been less successful when attempting to automate lower-level behaviors such as understanding natural (human) languages and computer vision.

As opposed to traditional symbolic AI systems in which humans try to figure out the rules underlying a process and then encode those rules as computer programs, **machine learning** systems are based on the idea of designing a system that learns by example. While a symbolic approach to character recognition might attempt to generate an explicit list of the features that make a capital "A" an "A" (such as the angle of the two lines that form the outer shape of the letter, the position of the cross bar, etc), a machine learning approach would instead focus on constructing a system that could be shown many different examples of the letter "A" and then discover for itself the features that make an "A" an "A" as opposed to a "B".

Some approaches to machine learning are biologically inspired, such as artificial neural networks which model certain aspects of the structure and function of biological neural systems (brains). Other approaches are based on more abstract mathematical models, such as Bayesian networks. In recent years, the machine learning approach, when paired with the vast quantities of data now available over the Internet and today's faster processors, has yielded significant progress in areas such as speech recognition, natural (human) language translation, and vision.

Whether or not these approaches (symbolic AI and machine learning) have a shot at achieving intelligent behavior depends on an underlying assumption or hypothesis (known as the physical symbol system hypothesis) being true. The **physical symbol system hypothesis** states that physical symbol systems are capable of generating intelligent behavior. In other words, intelligent behavior can be encoded into such a system. A physical symbol system is a collection of physical patterns (such as written characters or magnetic charges), called symbols, together with a set of processes (or rules) for manipulating those symbols (e.g., creating, modifying, deleting, and reordering them). A computer is a real world device that can be used to implement physical symbol systems, since computers are capable of both storing symbols (generally represented as strings of 0's and 1's) and manipulating those symbols under program control.

While most AI researchers believe the physical symbol system hypothesis to be true, it is only a hypothesis, not a proven fact. Some philosophers, on the other hand, are not convinced and believe that intelligent behavior will never be achieved by computers or any other physical symbol system. Others believe that, although the development of machines that act in an intelligent manner is possible or even likely, such systems will never be truly intelligent. In other words, these philosophers think that, while it is possible that a machine may at some point be able to "mimic" intelligent behavior so as to fool people into thinking it is intelligent, it will never really "be" intelligent...just a good fake.

These two schools of thought go by the names weak AI and strong AI. **Weak AI** refers to machines that behave in an intelligent manner, but makes no claim as to whether or not the underlying system is truly intelligent. **Strong AI** refers to machines that not only act in an intelligent manner, but also are intelligent.

At first glance, the debate over weak vs strong AI may appear pointless. After all, the debate is not over how the systems behave, but over the subtler question of whether machines can really possess self-awareness. The answer that we humans settle on to this rather esoteric question will eventually have an enormous impact on how we treat machines that display intelligent behavior. If we view them as elaborate devices then they will have no rights. One cannot be cruel to a microwave oven, after all. If, on the other hand, we choose to view them as true intelligences, then humans will be faced with a large number of rather thorny moral issues. Is it possible to be cruel to an AI? Should ownership of an AI be allowed, or would it be equivalent to slavery? What if we could construct AI's so that they enjoyed doing what we wanted? Wouldn't that solve the moral problems associated with owning them? Well, the vast majority of humans have decided that slavery is morally repugnant regardless of whether or not the slave might claim to be happy with his or her station in life. Shouldn't the same moral arguments apply to all truly intelligent beings?

If computer scientists ever succeed in the goal of creating machines that seem to act in an intelligent manner, this debate over AI rights is likely to become extremely heated (as the economic and social consequences could be enormous) – perhaps even more heated than the current debate over abortion that

rages in the US. In general, one's opinion on abortion is primarily determined by whether one believes "personhood" begins at conception, birth, or somewhere in between. This question has no definitive answer as "personhood" is not a rigidly definable concept and, in fact, has varied between different societies and over time. Some societies and/or religions view personhood occurring at conception. Other societies have defined personhood as occurring at various times following birth, such as ancient Sparta where newborns were examined by local elders and, if found inadequate, were thrown over a cliff. Today such a practice sounds barbaric, but to those living in ancient Sparta, the newborn wasn't truly a Spartan until the local elder pronounced the child to be one.

Just as there is no definitive test to determine where "personhood" begins, there is no such test to determine whether an AI is truly intelligent (strong AI) and therefore presumably deserving of rights, or whether it simply "mimics" intelligent behavior (weak AI). This point is succinctly made in the film "2001: A Space Odyssey" when a BBC reporter asks Astronaut Dave Bowman whether HAL has genuine emotions. Dave responds: "Well, he acts like he has genuine emotions. Of course, he's programmed that way to make it easier for us to talk to him. But as to whether or not he has real feelings is something I don't think anyone can truthfully answer."

Measuring progress: the Turing test

One of the most fundamental requirements necessary to accomplish any task is a method for detecting when the task has been completed. Absent such a test, a person will be unable to determine whether progress is being made on the task, or even recognize that the goal has been achieved.

In 1950, Alan Turing published a paper entitled "Computing Machinery and Intelligence" in which he described a test that he believed could be used to measure progress towards creating intelligent machines. The goal of the Turing test, as it has come to be known, is to determine whether a machine acts in an intelligent manner (i.e., whether weak AI has been achieved).

Here is one version of the Turing test: a human interrogator enters into a conversation with an entity (which may either be another human or a machine) via a computer terminal. The human and the entity are not allowed to see one another or communicate in any way, other than by typed messages. The goal of the human interrogator is to determine whether he or she is, in fact, communicating with another human or with a machine. If the human cannot reliably distinguish between the two, we conclude that the machine is exhibiting intelligent behavior.

This test is designed to limit bias against the machine. Passing the test does not require an ability to see, to hear, or to speak. It does not require the ability to perform physical tasks, such as riding a bike or playing a musical instrument. What is required is that the machine be capable of reasoning in a manner similar to humans and of communicating in a human language, such as English. While the fictional computer HAL of "2001: A Space Odyssey" could easily pass such a test, no real computer has ever even come close.

A brief history of artificial intelligence

While it is often difficult to precisely define the moment that a new field of endeavor springs into existence, a defining event in the emergence of AI as a formal academic discipline was certainly the Dartmouth Conference of 1956. This conference is credited with coining the term "artificial intelligence". It was attended by a number of individuals, such as John McCarthy, Marvin Minsky, and Claude Shannon who went on to be become the "founding fathers" of the field.

The Dartmouth Conference is also noteworthy for giving rise to what has to be one of the worst time and effort estimates ever generated by a group of experts in the history of Man. Essentially the organizers of the conference believed that a group of ten carefully chosen scientists working together over the summer of 1956 could accomplish the goals now associated with AI. Over half a century later, working with machines that are billions of times more powerful, we have yet to achieve many of the goals these visionaries first thought they would be able to tackle in a two month summer project.

As evidenced by the Dartmouth Conference, in the early days of computing there was much optimism that computers would attain general intelligence in a reasonably short period of time. This optimism was bolstered by an impressive array of accomplishments that were rapidly achieved. In the late 1950's, Arthur Samuel wrote a checkers playing program that could learn from its mistakes and thus, over time, become better at playing the game. There was much excitement when the program eventually became better than its creator at the game. Other AI programs developed in the late 1950's and 1960's include Logic Theorist (1956) by Allen Newell and Herbert Simon, which could prove simple mathematical theorems, SAINT (1963) by Jim Slagle which could solve calculus problems, and STUDENT (1967) by Daniel Bobrow which could solve word-based algebra problems.

Much of the work in this period was based on variations of early techniques and formal logics. What researchers soon discovered, however, was that the techniques they used to achieve their impressive results on small, limited problems could not be scaled up to attack more substantial problems. This was due to the underlying exponential nature of the search spaces associated with these problems. When one is dealing with a very small problem, such as searching a maze for an exit, the "brute force" approach of exhaustively testing every possible solution path is feasible. This approach, however, cannot be applied to larger problems, such as communicating in a natural language, where there are billions upon billions of potential solution paths.

For these reasons, the methods that were explored in the late 1950's and 1960's have become known as weak general methods. They are "general" in that they can be applied to a wide variety of problems. Yet they are "weak" in the sense that they cannot directly solve problems of substantial size in any area.

The way around the problem of exponential search spaces appeared to be the application of knowledge to limit the amount of searching required to reach a solution. In other words, if a program had a way to recognize which solution paths looked most promising, substantial amounts of time could be saved by exploring these paths first. This idea was formalized as a heuristic. A **heuristic** is a "rule of thumb" which is usually true. For example, a heuristic for the game of checkers is: "it is better to have more pieces on the board than your opponent." While this is not always true (e.g., you have more pieces but they are in terrible positions and will be lost on the next move), it is usually true. Other heuristics drawn from everyday life are, for example, bright red apples are tasty, milk with a later expiration date is fresher than milk with an earlier expiration date, jobs with higher salaries are more desirable than jobs with lower salaries, and so on.

Originally, AI programs had very little knowledge. They depended primarily on searching through a huge number of alternatives to find an acceptable solution. Over time, however, the emphasis switched from developing better search strategies to developing structures for encoding and applying knowledge to solve problems. One such structure, called a **script**, is used for representing knowledge about stereotypical situations, such as what happens at a birthday party or when one goes out to see a movie. A similar structure, called a **frame**, is used for representing knowledge about objects.

During the 1970's many AI researchers began developing and testing strategies for knowledge representation and manipulation. As was the case in the 1950's and 1960's, their implementation efforts generally focused on small, limited problems, which came to be known as **micro-worlds**.

One of the most impressive early examples of a micro-world was Terry Winograd's program SHRDLU. SHRDLU, developed in 1972, simulated a world filled with blocks, pyramids, and boxes together with a robot arm that could manipulate those objects. What made SHRDLU so unique for its time was that a human could tell the robot arm what to do by typing commands in ordinary English, such as "Find a block which is taller than the one you are holding and put it in the box." As this example illustrates, the system was capable of understanding some of the more subtle features of English such as pronoun reference. SHRDLU also understood simple "physics" concepts, such as: "two different objects cannot occupy the same physical space at the same time" and "if you release an object it will fall to the ground unless some other object is directly supporting it" and even "a cube cannot be supported by a pyramid".

SHRDLU showed that programs could converse "intelligently" with humans in English, if they were provided with a thorough knowledge of the world being discussed and the rules that governed what was possible in such a world. Although SHRDLU-like natural language interfaces were used in the text-based adventure games of the 1980's and early 1990's, the approaches employed by SHRDLU and similar programs to reason about their micro-worlds could not be easily scaled up to handle the vast amounts of knowledge necessary to make sense of the real world.

As AI researchers began to recognize these obstacles, the early optimism of the 1950's and 60's gave way to a period of retrenchment in the 1970's. Much of the government support and funding that had been lavished on AI in the early days dwindled during this time. The frustration of the AI researchers was that while they could create programs capable of impressive feats in very limited and focused areas, their techniques did not work well on the large, poorly defined problems that characterize the real world.

One of the breakthroughs of the early 1980's was the widespread realization that micro-worlds could be constructed to represent small but important aspects of human knowledge. Much of the knowledge that is valued in our world is specialized knowledge (knowledge that is possessed by only a small fraction of the human population) such as knowledge of finance, medicine, and law. Even within these fields people further specialize into subfields such as corporate tax law or neuromedicine.

Many of these highly specialized, highly valuable areas of human knowledge can be modeled by expert systems. **Expert systems** are computer programs that behave at or near (or even above) the level of a human expert in a narrowly defined problem domain, such as certain kinds of stock market transactions or credit card fraud detection. Expert systems focus on micro-worlds. But instead of being micro-worlds about simple domains of little intellectual or monetary value, they concern scarce and valuable information.

While the 1980's was the decade in which expert systems began to become commercially viable, their roots extend as far back as the late 1960's. In 1969, Ed Feigenbaum, Bruce Buchanan, and Joshua Lederberg developed the DENDRAL program at Stanford. DENDRAL could determine the structure of molecules from their chemical formula (e.g., H2O) and the results of a mass spectrometer reading. The first rule-based expert system was MYCIN by Feigenbaum, Buchanan, and Edward Shortliffe. MYCIN, which was developed in the mid 1970's, was capable of diagnosing blood infections based on the results

of various medical tests. The MYCIN system was able to perform as well as some infectious blood disease experts and better than non-specialist doctors.

With the emergence of expert systems in the 1980's, there was renewed interest in artificial intelligence. Governments and corporations once again began heavily funding AI research. This time, the goal was not to construct a general-purpose intelligence, but instead to develop practical commercial systems.

A large number of expert systems in fields ranging from financial services to medicine were implemented during this period. While many of these systems proved practical and cost effective, many did not. AI researchers tended to underestimate the difficulty of acquiring and encoding human expertise into the rule-based form required by expert system software. In some fields, it was found that the costs of acquiring and encoding expert knowledge could not be justified due to factors such as the rate at which knowledge in certain fields changed or the size of the potential target market. In other fields, such as credit card fraud detection, expert systems have become fully integrated into the way organizations do business and are no longer even thought of as "an AI application."

Expert systems, by their very nature, are limited to small problems. They focus on capturing what has come to be called surface knowledge. **Surface knowledge** consists of the simple rules that generally characterize reasoning in a particular area. For example, the rule "If mild fever and nasal discharge, then cold" could be part of a basic medical expert system. The type of knowledge captured by these rules is far different from a deep understanding of the field, which requires knowledge of anatomy, physiology, and the germ theory of medicine. While MYCIN was excellent at diagnosing blood infections, it had no understanding of this deeper knowledge. In fact, MYCIN didn't even know what a patient was or what it means to be alive or dead.

In addition to being constrained to surface knowledge, expert systems suffer from two other major limitations. One is that they do not learn. Humans must hand-code each of the rules that form the knowledge base of an expert system. These rules are fixed. They do not change over time. Hence, when particular rules are no longer valid, humans must recognize this situation and update the knowledge base. Otherwise, the expert system will provide dated advice.

Expert systems also tend to be quite brittle. AI researchers use the word brittle to mean that, when presented with a problem that does not fit neatly into their area of expertise, an expert system will often give completely inappropriate advice. This problem is made more vexing by the fact that few expert systems are able to determine when they are being asked to give advice on problems that are outside their area of expertise.

Artificial intelligence research went through yet another sea change in the late 1980's and early 1990's. As the limitations of expert systems began to become apparent, attention turned to the much-neglected topic of machine learning, and in particular, neural networks.

Neural networks, or connectionist architectures, consist of a large number of very simple processors which are interconnected via a network of communications channels, where each channel has an adjustable strength or weight associated with it. Neural networks are modeled (very loosely) on the brain. Each processor represents a **neuron** (a brain cell), and each weighted interconnection a **synapse** (the connections between brain cells). The knowledge contained in a network is encoded in the weights

associated with the connections. The most astonishing feature of neural networks is that humans do not directly program them; instead, they learn by example.

In 1943, Warren McCulloch and Walter Pitts proposed a simple model of artificial neurons in which each neuron would be either "on" or "off". The artificial neurons had two kinds of inputs: "excitatory" and "inhibitory." The neuron would "fire" (i.e., switch to "on") when the total number of excitatory inputs exceeded the number of inhibitory inputs.

McCulloch and Pitts were able to prove that their networks were equivalent in computational power to general-purpose computers. John Von Neumann showed that redundancies could be added to McCulloch-Pitts networks to enable them to continue to function reliably in spite of the malfunction of individual elements. While these early neural networks were certainly interesting, they were of limited practical value, since the network interconnections had to be set by hand. Moreover, no one knew how they could be made to learn.

Work by Frank Rosenblatt in the 1950's and early 1960's overcame the learning problem to some degree. Rosenblatt proposed the **perceptron**, a single-layer, feed-forward network of artificial neurons together with a learning algorithm. He was able to prove that his learning algorithm could be used to teach a perceptron to recognize anything it was capable of representing simply by presenting it with a sufficient number of examples.

This was a very important result and led to much excitement. Perceptrons were built and trained to recognize all manner of things during the 1960's. One famous example was a perceptron for distinguishing "males" from "females" by examining photos of peoples' faces.

Research into artificial neural networks came to an abrupt halt in the early 1970's. Many people credit Marvin Minsky and Seymour Papert's 1969 book, Perceptrons, with bringing about the end of research into neural network based machine learning for nearly a decade and a half. Minsky and Papert proved that, while it is true that perceptrons can learn anything they are capable of representing, the fact is that they are actually capable of representing very little. The most often cited example is that a perceptron with two inputs cannot learn to distinguish when its inputs are the same from when they are different. In other words, a perceptron cannot represent the *xor* operation.

Minsky and Papert's proofs only applied to perceptrons (single-layered neural networks) not multiple-layer networks. However, at the time, single-layered networks were the only kind of neural networks that computer scientists had a learning algorithm for. In other words, the situation in the early 1970's was that AI researchers knew that neural networks could, in theory, compute anything that a general-purpose computer could. But, the only kind of neural networks that they had a learning algorithm for (perceptrons) could not compute many basic functions.

In the late 1980's and 1990's many AI researchers began returning attention to neural networks. A number of groups independently developed the **back-propagation learning algorithm**. This algorithm allows multiple-layer feed-forward networks to be trained. The good news was that these networks, given a sufficient number of processing units, could represent any computable function. The bad news was that back-propagation, unlike the learning rule for perceptrons, does not guarantee an answer will be found simply because it exists. In other words, back-propagation does not guarantee that a multilayer network will be able to learn a particular concept regardless of how many examples it is given or how

much time is spent going over these examples. In many cases, however, the algorithm does enable the network to successfully learn the concept.

Neural networks and other approaches to machine learning also suffer in comparison to the symbolic approach to machine intelligence in that they are often unable to explain or justify their conclusions. The practical result is that one can never really be sure that the network has been trained to recognize what was intended.

There is a humorous story from the early days of machine learning about a network that was supposed to be trained to recognize tanks hidden in forest regions. The network was trained on a large set of photographs (some with tanks and some without tanks). After learning was complete, the system appeared to work well when "shown" additional photographs from the original set. As a final test, a new group of photos were taken to see if the network could recognize tanks in a slightly different setting. The results were extremely disappointing. No one was sure why the network failed on this new group of photos. Eventually, someone noticed that in the original set of photos that the network had been trained on, all of the photos with tanks had been taken on a cloudy day, while all of the photos without tanks were taken on a sunny day. The network had not learned to detect the difference between scenes with tanks and without tanks, it had instead learned to distinguish photos taken on cloudy days from photos taken on sunny days!

Half a century and counting

At the beginning for the 21st century, after more than 40 years of intense effort by some of the most brilliant minds on the planet, researchers seemed much farther away from achieving many of the basic goals of AI than in the late 1950's when they were just getting started. The year 2001 came and went and machines with the capabilities of HAL remained firmly in the realm of science fiction. At the turn of the century, in only the most specialized domains could computers be said to "see" and no one really "talked" to their computers.

With the arrival of the 21st century came the rise of the Internet and increasingly powerful computer systems. By the later part of the first decade of the 21st century much of human knowledge was available online, accessible to most anyone, anywhere, with a smart phone and a broadband Internet connection. The Internet provided AI researchers for the first time with vast quantities of digital data, generated by hundreds of millions of people, writing in many dozens of human languages. The emergence of Big Data and statistical machine learning techniques has enabled renewed progress in many areas of AI research such as speech recognition, question answering systems, machine translation and computer vision.

Computer Scientists use the term **big data** to describe the extremely large datasets that are becoming available in a wide variety of disciplines, such as genomics, social networks, astronomy, Internet text and documents, atmospheric science, and so on. Big data is often characterized as high volume (very large datasets), high velocity (the data changes rapidly), and/or high variety (large range of data types and sources of data).

Statistical machine learning techniques refer to methods, or techniques, that apply statistical models to large amounts of data in order to enable machines to automatically infer relationships from the patterns that can be found in the data. There are a number of statistical inference models used in machine learning, but one of the most popular is called **Bayesian inference**.

Note that while statistical machine learning techniques differ from the neural networks mentioned above, they are both sub-disciplines of machine learning in that they are both trained on large data sets and learn to recognize patterns present in the data. The major difference between the two is that neural networks are inspired by biological neural systems and the feedback cycles that appear to be inherent to those systems, while statistical machine learning techniques have grown out of mathematics and statistics rather than biology. Neural networks were once the most popular machine learning model but since the turn of the century statistical methods have become dominant.

Until relatively recently, progress had been quite limited on a number of important problems in AI. For example, the traditional symbolic AI approach to translating documents from one language to another proved ineffective. There are simply too many exceptions to the "rules" governing how humans use languages for it to be practical to try to capture them all. Many computer scientists thought that computers to would need to master common sense knowledge in order to "understand" what was being said before substantial progress could be made. What big data and statistical machine learning techniques have shown us is that, given enough data many of these problems can be solved to a large degree by looking for patterns in the data (except perhaps for deep understanding).

To use an example you may be familiar with, let's look at Google Translate. Google Translate learns to translate documents between two languages, say English and French, by first comparing millions of documents, such as books and web sites, which humans have already translated. These document pairs are scanned for statistically significant patterns. The billions of patterns generated by this process can then be used to translate new documents between the two languages. Given enough source data the resulting translations, while not perfect, are usually good enough for the reader to understand the ideas the writer is attempting to communicate.
In the Google approach, humans are not directly teaching the computers how to translate documents. Instead, humans have taught the computers how to compare documents and look for statistically significant patterns. We humans then give these computers millions of pairs of already translated documents. The computers then figure out which input patterns (say in French) are most likely to be associated to which output patterns (say in English). When presented with some new input text in French, the system produces the most likely English output pattern associated with that text.

It is very important to understand that Google Translate has no understanding of what the words and sentences it translates actually mean. It simply replaces one string of text with another string of text based on statistical patterns derived from human translated documents.

As we approach the end of the second decade of the 21st century we appear to be on the verge of achieving real time natural language translation. Real time natural language translation occurs when human speech is translated from one language to another as it is spoken, with little or no apparent delay. While we are not there yet, applications such as Google Translate with Conversation Mode for Android, where two people can take turns speaking and the smart phone acts as translator, hint at what may soon become possible.

Using language intelligently
Communicating with computers by speaking with them has long been the Holy Grail of artificial intelligence. However, after more than 50 years of effort, the capabilities of the HAL computer from "2001: A Space Odyssey" remain stubbornly out of reach. Within the past decade or so, substantial

progress has been made on parts of the problem; specifically, in the disciplines of speech recognition, speech synthesis, and question answering systems.

Speech recognition refers to the automatic conversion of human speech into text, while **speech synthesis** focuses on the automatic generation of human speech from text. As such, speech recognition and speech synthesis applications are complementary in nature. They often form the first and last steps in voice-based search and question answering systems (such as that available in Google Now) or virtual assistant systems (such as Siri).

Speech recognition systems can be characterized along three different dimensions: the size of the vocabulary recognized by the system, whether or not the system is speaker dependent or independent, and whether the system supports continuous speech or is limited to discrete utterances.

Vocabulary size refers to the number of words or phrases the system can recognize. In general, the larger the vocabulary the more difficult the recognition problem becomes. A system that is limited to recognizing "yes" and "no" is rather easy to construct, while systems designed to recognize hundreds, thousands, or tens of thousands of words become progressively more difficult to build. A system's recognition rate is the percentage of utterances that the system is capable of correctly identifying. Often a system's recognition rate would dramatically decrease as vocabulary sizes increased – especially in earlier systems. Background noise and other factors can also significantly influence a system's recognition rate.

A system is said to be speaker independent if it is intended to be used by a general audience (i.e., its ability to recognize utterances is independent of who is speaking). Speaker dependent systems, on the other hand, must be trained to recognize an individual's voice. Training usually consists of having the speaker read aloud a number of pre-selected text passages while the computer listens to the speaker's voice. Once trained, only that individual can reliably use the system.

Human speech is continuous, in that we don't pause between each individual word. We are often unaware of this fact since we are so good at speech recognition. In fact, we tend to "hear" the individual words even when they are spoken as a continuous stream. To see this, think of the last time you heard someone speaking in a language that is foreign to you. Didn't it seem that they were just producing a stream of sounds and not a sequence of words? In fact, one of the hardest things about learning to understand a foreign language is just picking out the individual words.

Continuous utterance recognition systems allow people to speak naturally without inserting pauses between words; whereas discrete utterance recognition systems require speakers insert brief pauses between each spoken word or phrase. Humans find speaking in this way very unnatural.

Today's end-consumer facing speech recognition systems tend to be large vocabulary, speaker independent, continuous utterance systems. Most anyone can use them immediately without first training the system to understand their voice, and speakers can say pretty much whatever they want, while speaking in a natural voice.

Such systems are a relatively recent advance. Prior to the release of Google's voice search in late 2008, most speech recognition systems had to compromise on one of the three primary features. So, for example you could have a speaker dependent, large vocabulary, continuous utterance system, which

required the speaker to train the system before use; or a speaker independent, small vocabulary, continuous utterance system, which anyone could use but could only recognize a limited number of words and phrases; or even a speaker independent, large vocabulary, discrete utterance system, which anyone could use but…required...a...pause...between...each...word.

As was the case with machine translations, recent advances in speech recognition are due in large part to rise of big data (the availability of lots and lots of examples of spoken text) and statistical machine learning techniques. Most speech recognition programs model human speech production using a formal mathematical model called a Hidden Markov Model.

A **Markov Model**, or **Markov Chain**, is a system that consists of a number of discrete states where the probably of being in a particular state at a particular time is dependent on the previous state and likelihood of transitioning between that state and the current one. Individual states in the model generate outputs with some probability. In a **Hidden Markov Model** (HMM) the sequences of transitions between states in the underlying Markov Model are not visible, but the outputs produced by those transitions are. HMM's enable one to determine the most likely sequence of states that were traversed in the underlying system based solely on the observed output. HMM are useful in speech recognition programs since all we are able to observe are the sounds that were produced (the output) and we wish to infer the underlying (hidden) sequence of words that produced those sounds.

Understanding a spoken human language, like English, involves solving at least two separate problems. The first problem, which we have been discussing so far, is recognizing the individual spoken words. The second problem, known as semantics, involves comprehending the meaning of those words. Using big data, machine learning techniques, and Hidden Markov Models, computer scientists and engineers have made good progress on the first problem. Much less progress has been made on the second problem.

Thus far, creating systems that can truly comprehend general spoken (or written) English is not currently possible. However, reasonable results can often be achieved if the topic of conversation is narrowly focused and task oriented. One way of judging our progress to date in giving computers the ability to use language intelligently is to look at the evolution of voice-based search and the rise of "intelligent" assistants. Such a review can also help us understand the limitations of our present tools and techniques.

The first large-scale general purpose (speaker independent, large vocabulary, continuous utterance) deployment of speech recognition technology intended for a wide audience was Google's voice-based web search. Released in 2008 for Android and iPhone smart phones, this technology enabled one to speak a search query, rather than type. While remarkably impressive for its time, Google voice search simply returned a web page of Google search results, exactly as if the speaker had typed the search query.

Two years later, in 2010, Google extended this technology to enable Android phones to perform simple actions based on spoken input. For example, the updated system made it possible to send a text message or email to someone on your contact list by speaking the request (e.g., "Send text message to the prof. Wow! AI has a come a long way in a little over fifty years."). Other actions included calling the phone number of a business by name (e.g., "Call Papa John's in New Orleans"), getting directions (e.g., "Navigate to Louisiana Tech University"), or finding and playing music. This technology required the system to have some limited understanding of what the speaker was asking the phone to do. The

implementation was straightforward as the system simply listened for special keywords, such as "navigate", and then launched the appropriate application passing it any specified parameters, such as "Louisiana Tech University", when the keyword was detected.

Apple followed suit a year later in the fall of 2011 with the introduction of Siri for the iPhone 4S. Siri, billed as a "personal assistant", could handle a wider range of tasks than Google Voice Actions and provided for a more natural spoken interface. The interface was far less rigid and didn't require the speaker to remember particular keywords. So, for example, instead needing to say something precise like "Navigate to Louisiana Tech University" one could say "Show me how to get to Louisiana Tech" or "Give me directions to Louisiana Tech."

Another big innovation was that Siri implemented the ability to respond verbally to some tasks. For example, you could ask Siri "What's my schedule like for tomorrow?" and "she" would access your calendar and go through your list of appointments for the next day. Siri could also schedule an appointment for you and even notify you of conflicts with existing appointments. The system also possessed the ability to verbally respond to a very limited range of factual questions about time and the weather, such as "What time is it in London?" or "Will it rain tomorrow?"

Since weather varies by location, in order to correctly answer such questions Siri needs to know the location to which the user is referring. Unless we explicitly state a location, when we ask such a question we generally mean "right here" which Siri can determine from the iPhone's GPS or cell tower triangulation. However, the implied location can change depending on the context of the conversation. Siri is smart enough to know that when you ask it "What time is it in London?" followed immediately by "What will the weather be like tomorrow?", you are probably referring to the weather in London, England. Similarly, if you ask "What will the weather be like in Paris tomorrow?" followed immediately by "in Rome?", it knows from the context of the conversation that the phrase "in Rome?" means "What will the weather be like in Rome tomorrow?" In order to successfully answer these kinds of questions it is necessary for Siri to possess a rudimentary understanding of conversation and context, something absent from the original version of Google Voice Actions.

Apple further improved Siri with the release of the iPhone 5 in September 2012. The revised version of Siri included expanded knowledge of sports and entertainment, allowing the system to verbally respond to a wider range of questions. For example, in response to "Who is Drew Brees?" Siri replied "Drew Brees currently plays quarterback for the Saints" while showing his picture and player stats. Despite this rather impressive result, the vast majority of questions posed to Siri (as of late 2013) generally return web pages from either a Google search or Wolfram Alpha query.

In July 2012, several months prior to the release of the iPhone 5, Google once again raised the bar with the release of Google Now's integrated voice search which dramatically expanded the scope of questions that can generate a verbal response. By pulling information from Wikipedia and a variety of other sources, reformatting that data, and providing a brief verbal summary, Google Now can provide a verbal response to large numbers of questions over substantial areas of human knowledge. For example, if I ask "When did Man first step foot on the Moon?" Google responds with "July 20, 1969 is one guess based on results below" while displaying the Google search results on which it based its answer.

Over the course of just four years, from 2008 to 2012, voice based interfaces went from quite rare to rather commonplace. As we approach the end of the second decade of the 21st century voice based

personal assistants and question answering systems are being used daily by more and more people. However, these systems are still quite limited in their overall capabilities compared to the ultimate goal of constructing computers that can converse intelligently.

IBM's Watson is an example of a question answering system that goes far beyond simple search. It is most famous for winning a two game Jeopardy! tournament in 2011 while competing against Ken Jennings and Brad Rutter, the two most successful humans to ever play the game. During the games, Watson was given exactly the same clues as the human players, with the exception that Watson's clues were in textual form since it could neither see nor hear. After receiving the clue, Watson would decompose it into key words and phrases and then simultaneously run thousands of language analysis algorithms to find statistically related phrases in its repository of stored text (obtained from reference books, web sites, and Wikipedia). Watson then compared these thousands of different results, filtering against clues in the question category (such as the answer is probably a city) to select its final answer, which was generally the most popular result (i.e., the one returned by the greatest number of algorithms). The process, which is far more complex than that used by Google Now and Siri, is often called deep question answering.

Despite recent progress, it is important to understand that neither IBM's Watson nor Google Now's voice search understand the questions they are being asked nor the answers they provide in the way humans do. These systems "simply" compare strings of text in the questions to strings of text in their knowledge bases and then, using statistics along with filters for the type of answer most likely sought (e.g., a name, a date, a place) generate the most likely answer. These systems have no understanding of the meaning of the words in the questions, or the words in the knowledge base, or even the words in the answer that is generated. Given these constraints, it is truly amazing at the level of performance that these systems have achieved.

Game playing and search techniques

One of the earliest ideas in the quest for machine intelligence (and one that is still used today) is the modeling of decision making as state space search. A state space is a collection of three things:

(1) A way to represent the state of some system at a particular moment in time by a compact description of the important characteristics of that system;
(2) A way to determine the legal moves, or transitions, between states of the system; and
(3) A test to recognize when the goal has been achieved. It is also sometimes necessary to have a separate test to recognize a loss (i.e., a situation from which the goal can never be reached).

Suppose that we wanted to model a board game, such as chess or checkers, using a state space. First, we would need a way to represent the configuration of the game at any point in time. For a board game, this could consist of a simple two-dimensional grid, or array, containing symbols to represent the position of each player's pieces. Next, we would need a description of the rules that apply to moving pieces in the game. For board games, such as chess, these are simply the rules that describe the basic movements of each piece (e.g., the rook may be moved any number of unoccupied squares forward or backward or left or right). While these rules differ from game to game, they are generally quite simple and can be easily mastered. Finally, a test to recognize a winning situation would be necessary. In chess this would amount to a description of checkmate applied to your opponent. Checkmate, applied to your side, also functions as the loss test so that you can know when to quit playing.

Once the game, or problem, has been described in terms of a state space, making a move can be modeled as searching through the space of all possible moves. To be sure, the state space could be huge and intractable to search through! In order to be able to explore deep into a state space without having to deal with an exponentially increasing number of states, heuristic searches are often used. A **heuristic** is a simple measurement that is used as an indicator of some other aspect of the problem that would be difficult or impossible to accurately measure. For example, people often use price as an indicator of quality. Although everyone knows that price and quality are very different things, we sometimes say things like "your new car must have cost you a fortune" when we mean that your new car is very nice. One reason we do things like this is because price is something that is easy to measure, while quality is much more difficult to accurately capture.

A heuristic search is a search that incorporates a heuristic to guide the direction pursued by the search. A simple heuristic for the game of checkers might be:

$$H = NumberOfOpponentPiecesCaptured - NumberOfYourPiecesLost$$

This heuristic reflects the fact that, in checkers, the more pieces that you have captured from your opponent and the fewer that you have lost to him or her, generally the better off you are. A heuristic search employing *H* would first pursue paths that tend to increase *H*, because these paths are more likely to lead to a win.

A simple heuristic search that can be used in single player games (such as finding your way through a maze) is called "hill climbing." Hill Climbing searches work by:

(1) Generating all states that are one transition away from the current state;
(2) Applying a heuristic function to each of these states;
(3) Moving to the state with the highest heuristic value; and
(4) Repeating this process until the goal is reached.

While hill climbing has some drawbacks, such as the fact that it can get stuck on what are called "local maximums" where every possible next move appears worse than the current state, given a good heuristic function hill climbing can frequently solve problems much more effectively than a blind search.

There are other, more complex, heuristic search algorithms, such as **A*** (pronounced "A star") and **mini-max**, which overcome many of the shortcomings of hill climbing. Like hill climbing, A* is tailored for single player games. Mini-max is appropriate for two-player games. While additional details about these search procedures are beyond the scope of this lesson, the main thing to remember about heuristic searches is that they generally outperform blind searches because they incorporate knowledge about the problem being solved (or game being played) in order to concentrate effort in directions that are likely to lead to success (rather than searching blindly about for a solution).

Automated reasoning

At about the same time that some computer scientists were studying ways of creating computer systems that could play games (late 1950's and 1960's), other computer scientists were thinking about the problem of how to build a system that could "reason." Much of the early work on automated reasoning focused on constructing computer programs to carry out the rules of symbolic logic. A **symbolic logic**, or formal logic, combines a way of representing information using symbols together with a collection of rules for manipulating those symbols in order to reach logically valid conclusions.

The reason that symbolic logic was of such interest to computer scientists studying artificial intelligence is that these logics are capable of "reasoning" about concepts and ideas based purely on the form of the statements used to represent them. In other words, a symbolic logic can draw logically valid conclusions about a collection of statements based, not on the meaning of those statements, but on the form in which they are written. There are many kinds of symbolic logics, and discussing them is beyond the scope of this lesson.

Neural networks and machine learning

Many of the recent advances in AI, such as progress in speech recognition and automatic translation, have been due to both big data and the resurgence of interest in machine learning techniques (i.e., systems that learn from examples instead of being explicitly programmed for every eventuality).

One particular approach to machine learning, a simple neural network model called the perceptron, is one of the oldest neural network models and the first to gain widespread popularity in the 1960's. While perceptrons are no longer used in practice today, this relatively simple, biologically inspired computing model provides a good starting point for getting your head around the notion of machine learning systems.

Human brains are composed of trillions of individual nerve cells, called **neurons**. Like all cells, neurons have a nucleus that keeps the neuron alive and functioning. In addition, neurons have a large number of branched protrusions, called **dendrites**, that receive chemical signals from other neurons. Neurons also have long thin fiber-like appendages, called **axons**, down which they send electro-chemical signals. At the end of a neuron's axon are branch-like structures that come in contact with the dendrites of other neurons. In response to the signals it receives from other neurons, a neuron may fire, sending an electro-chemical pulse down its axon in order to transmit signals to other neurons.

Neurons are not in direct physical contact with one another. Instead, there are tiny gaps, called **synapses**, between the dendrites of one neuron and the axons of others. The signals that pass between neurons must cross these synaptic gaps. This is accomplished by the signaling neuron releasing neurotransmitter chemicals into the synapse. A synapse may be either excitatory or inhibitory. Signals arriving at excitatory synapses increase the likelihood that the receiving neuron will fire. Signals arriving at inhibitory synapses decrease the likelihood of the neuron firing. An interesting feature of biological neurons is that they appear to be work in an "all or nothing" fashion (i.e., they either "fire" or they don't). In other words, neurons do not appear to fire at different strengths; instead, they appear to be either "on" or "off".

A **perceptron** is a simple processing element that models some of the features of individual neurons. The output of a perceptron is either 0 or 1. The use of a binary output in the perceptron model reflects the fact that biological neurons appear to either "fire" or "not fire". The inputs to a perceptron, on the other hand, can be arbitrary real numbers: positive, zero, or negative, with fractions allowed.

In order to model the fact that biological neurons possess both excitatory and inhibitory synapses, and the fact that these characteristics appear to vary in strength from synapse to synapse, every input received by the perceptron is multiplied by a weight. As is the case with the inputs themselves, weights are real numbers: they may be positive, zero, or negative, and fractions are allowed.

Once the inputs have been weighted, they are sent to a summation unit that adds them together. The sum of the weighted inputs is then sent to a threshold comparator. Every perceptron will have an internal threshold value that controls how sensitive it is to its inputs. If the weighted sum of a perceptron's inputs is greater than its internal threshold value, then the perceptron will fire (i.e., generate a 1); otherwise, the perceptron will not fire (i.e., generate a 0).

Because each input is multiplied by a weight, large positive weights increase the effect of a positive input, while smaller positive weights decrease the effect of an input. A weight of zero causes a perceptron to ignore an input. Negative weights, when applied to positive input values, decrease the likelihood of a perceptron firing. In general, when considering positive inputs, positive weights model excitatory synapses (the larger the weight the greater the excitatory effect). Negative weights model inhibitory synapses.

As a perceptron learns, it adjusts the values of its weights and its internal threshold. In fact, the only way a perceptron can "learn" is to adjust its weights and threshold. Those are the only things the perceptron can be said to "know".

In order for a perceptron to learn, it must be trained. Training involves presenting the perceptron with a group of inputs known as a **training set**. At the beginning of the training process, the perceptron makes random guesses as to whether or not it should fire when presented with a particular input. Whenever the perceptron guesses wrong, its weights will be adjusted by the **perceptron learning rule**. After being adjusted, the training set will be presented to the perceptron again. If the perceptron generates any incorrect results, the learning rule will be applied again. This process will continue until the perceptron correctly classifies all members of the training set.

Before a perceptron is fully trained, it can produce two kinds of incorrect outputs: false positives and false negatives. A false positive occurs when the perceptron fires when it should not have. False negatives occur when the perceptron does not fire when it should have. The perceptron learning rule distinguishes between these two types of errors.

An amazing characteristic of the perceptron learning rule, proven by Rosenblatt in the early 1960's, is that if it is theoretically possible for a perceptron to learn the training set you have presented it with, this algorithm will converge on that solution. In other words, if you run your training examples on the perceptron, modify the weights according to the learning rule, and then repeat the process over and over, eventually all examples will be properly classified. If the training set is **perceptron learnable**, the learning process is guaranteed to eventually halt.

Another remarkable feature of the perceptron learning rule is that it works no matter what values are chosen for the initial weights. Lucky guesses for the initial weights may get you to the solution faster than unlucky guesses, but either way the learning rule will converge on the solution (if there is one) in the end, regardless of how good or bad the initial guesses were.

If we wanted to teach a neural network based system to be able to distinguish between men and women from photographs of their faces, for example, we would digitize a large number of male and female photographs, and then train the network on those photos. What we would expect at the end of the process is that the network has learned to distinguish between male and female facial features. We

would be extremely disappointed if the only males and the only females the system could reliably classify were those it had been shown in the training set.

This lesson has covered a lot of material related to AI. It is not meant to be exhaustive nor too technical in nature; rather, it is meant to provide a decent overview of the field of AI and how it has the potential to impact our world.