14 Machine learning and natural language processing

… one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say "This is really written in English, but it has been coded in some strange symbols."
Warren Weaver[1]

Ideas of probability: The frequentists and the Bayesians

We are all familiar with the idea that a fair coin has an equal chance of coming down as heads or tails when tossed. Mathematicians say that the coin has a probability of 0.5 to be heads and 0.5 to be tails. Because heads or tails are the only possible outcomes, the probability for either heads or tails must add up to one. A coin toss is an example of *physical probability*, probability that occurs in a physical process, such as rolling a pair of dice or the decay of a radioactive atom. Physical probability means that in such systems, any given event, such as the dice landing on snake eyes, tends to occur at a persistent rate or relative frequency in a long run of trials. We are also familiar with the idea of probabilities as a result of repeated experiments or measurements. When we make repeated measurements of some quantity, we do not get the same answer each time because there may be small random errors for each measurement. Given a set of measurements, classical or *frequentist* statisticians have developed a powerful collection of statistical tools to estimate the most probable value of the variable and to give an indication of its likely error.

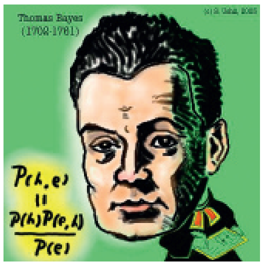
An alternative view of probability reflects the strength of our belief that the coin is fair and not what statisticians call *biased*, tending to give one result more frequently than the other. For example, perhaps we have reason to think that the coin being fair, with a 50 percent probability of coming up heads, is only one possibility. Maybe we think there is an equal chance that the coin is biased and will come up as heads 80 percent of the time. Before tossing the coin, we suppose that each of these two options is equally likely. But after tossing the coin ten times and observing eight heads, we will want to modify our beliefs. It now makes sense for us to believe that there is a greater than fifty-fifty chance that the coin is biased toward heads. The assumptions we had before tossing the coin are called *prior beliefs* – or just *priors* – because we form them prior to gathering any evidence about the situation. After incorporating the results of observations, we modify our beliefs. These modified beliefs are called *posterior beliefs* or *posteriors*. The technology of *Bayesian inference* that we explain in this section determines – in a precise, mathematical way – how we should change our prior beliefs. Bayesian inference is a decision-making technique that takes into account both observed data and prior beliefs and allows us to eliminate the least likely options. The frequentist interpretation of probability is fine for problems where we can make repeatable experiments and measurements. But such a frequentist approach cannot make predictions about nonrepeatable events, such as "What is the probability of an earthquake in Seattle next year?" or, more commonly, "What is the probability of rain in Seattle tomorrow?" The Bayesian approach can provide a mathematical basis for such predictions.

An English clergyman named Thomas Bayes (B.14.1) introduced what we now call the Bayesian approach in the 1700s. His goal was to learn the probability of a future event given only the number of times such an event had or had not occurred in the past. His paper "An Essay towards Solving a Problem in the Doctrine of Chances" contains the following example:

> Picture a newborn witnessing his first sunset. Being new to this world, he doesn't know if the sun will rise again. Making a guess, he gives the chance of a sunrise even odds and places in a bag a black marble, representing no sunrise, and a white marble, representing a sunrise. As each day passes, the child places in the bag a marble based on the evidence he witnesses – in this case, a white marble for each sunrise. Over time, the black marble becomes lost in a sea of white, and the child can say with near certainty that the sun will rise each day.[2]

This example illustrates the basic Bayesian approach. The newborn has an initial degree of belief in whether or not the sun will rise that is just a fifty-fifty guess. This belief is the baby's *prior*. As the child gathers more data, he or she can update this belief to obtain a more accurate prediction for the probability of a sunrise, the *posterior*.

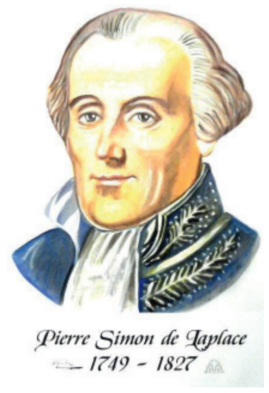
In his paper, Bayes describes a *thought experiment* in laborious detail, an experiment that we could now simulate very easily with a computer. He imagines that he turns his back to a square table and asks his assistant to throw a ball onto the table. The ball has just as much chance of landing at any place on the table as anywhere else. Bayes cannot see the table and has no idea where



B.14.1. Thomas Bayes (1701–61) was an English clergyman who did pioneering work in probability theory. In fact, his major work was published after his death and his papers were edited by the Welsh scientist, clergyman, and philosopher William Price. Bayes' paper, "An Essay towards Solving a Problem in the Doctrine of Chances," containing the famous result now known as Bayes Theorem, was published in *Philosophical Transactions of the Royal Society* in 1763. William Price had a remarkable career, was a personal friend of many of the founding fathers of the United States, and, with George Washington, received an honorary doctorate from Yale in 1781.

the ball has landed. The assistant now throws a second ball onto the table and reports to Bayes only that it landed to the left or right of the original ball. If it landed to the left, Bayes can deduce that the initial ball is slightly more likely to be in the right half of the table than in the left. The friend tosses another ball and reports that it lands to the right of the first ball. From this information, Bayes knows that the original ball cannot be at the extreme right of the table. With more and more throws, Bayes can narrow down the range of positions for the first ball and assign relative probabilities for different ranges. Bayes showed how it was possible to modify his initial guess for the position of the first ball, his prior probability, and to produce a new posterior probability by taking into account the additional data he had been given.

Although Bayes was first to suggest the use of probability to represent belief, it was the French mathematician Pierre-Simon Laplace (B.14.2) who developed this idea into a useful tool for many different types of problems. Laplace had become interested in probability through reading a book on gambling and did not initially know of Bayes' work. Laplace published his first paper on this subject in 1774 with the title "Mémoire sur la probabilité des causes par les événements" (Memoir on the Probability of the Causes of Events), so that his method is often abbreviated as just the "probability of causes." One of the first major applications of his new theory was to an analysis of the data on births in London and Paris. He wanted to know whether the data supported the suggestion of Englishman John Graunt that slightly more boys were born than girls. Using the christening records from London and Paris, Laplace concluded that he was "willing to bet that boys would outnumber girls for the next 179 years in Paris and for the next 8,605 years in London."[3] Later in his life, Laplace turned to frequentist techniques to deal with the large quantities of reliable data on all sorts of subjects. In 1810, he proved what is now called the *central limit theorem*, which justifies the taking of the average of many measurements to arrive at the most probable value for a quantity. When the French government published detailed data on such events as thefts, murders, and suicides, all the governments in Europe started studying statistical data on a whole range of subjects. Bayes' idea that the probability of future events could be calculated by determining their earlier frequency was lost in a welter of numbers. As the nineteenth century progressed, few people regarded the idea that the uncertainty of some prediction could be modified by something as subjective as "belief" as a serious scientific approach. Apart from a few isolated instances, it was not until the middle of the twentieth century that mathematicians and scientists again took seriously a Bayesian interpretation of probability and considered it a valid tool for research. Today, the Bayesian approach has a wide variety of uses. For example, doctors employ it to diagnose diseases, and genetic researchers use it to identify the particular genes responsible for certain traits.



B.14.2. Pierre-Simon Laplace (1749–1827) was one of the giants of mathematics and science. He is often referred to as the "French Newton" because his works made a major contribution to many areas of knowledge including astronomy, mechanics, calculus, statistics, and philosophy. One of his tasks as a member of the French Academy was to standardize European weights and measures and, in 1799, the meter and the kilogram were introduced as standards.

Bayes' Rule and some applications

The modern revival of Bayesian thinking began in the 1940s. In 1946, Richard Cox, a physicist at Johns Hopkins University, looked again at the fundamentals of the Bayesian view of probability. In particular, he wanted to find a consistent set of rules for reasoning about beliefs. First, he had to decide how to rank degrees of belief, such as whether the coin we talked about earlier was a fair coin with a 50 percent probability of coming up heads or a biased coin with an 80 percent probability of coming up heads. He proposed ranking how much we believe these possibilities by assigning a real number to each proposition such that the larger the number, the more we believe the proposition. He put forward two *axioms* (established rules) as being necessary for logical consistency. The first was that if we specify how much we believe something is true, we are also implicitly specifying how much we believe it is false. Using a scale of real numbers from 0 to 1 to specify beliefs, this axiom says that the belief that something is true plus the belief that the same thing is false must add up to one. This is the same as the usual *sum rule* for probabilities, which holds that the probabilities for all possible outcomes must add up to one.

Cox's second axiom is more complicated. If we specify how much we believe proposition Y is true, and then state how much we believe proposition X is true given that Y is true, then we must implicitly have specified how much we believe that both X and Y are true. Assuming some initial background information that we denote by B, we can write this belief relationship as an equation as follows:

$$\textbf{Prob (X and Y | B) = Prob (X | Y and B)} \times \textbf{Prob (Y | B)}$$

In words, this equation says that the probability that both X and Y are true, given background information B, is equal to the probability that X is true given that Y and B are true, times the probability that Y is true given B, regardless of proposition X. The vertical bar "|" separates the different propositions in these probabilities. This equation is the usual *product rule* for probabilities, which states that the probability of two independent events occurring simultaneously is the result of multiplying the individual probabilities together. The product rule is easy to derive from a frequentist approach. Note that all probabilities are conditional on the same background information B.

We can now derive the mathematical formula representing Bayes' Rule for probabilities. It is obvious that the probability that X and Y are both true does not depend on the ordering of X and Y on the left-hand side of our equation. We therefore have:

$$\textbf{Prob (X and Y | B) = Prob (Y and X | B)}$$

By expanding each side and doing a little rearrangement, we arrive at Bayes rule:

$$\textbf{Prob (X | Y and B) = Prob (Y | X and B)} \times \textbf{Prob (X | B) / Prob (Y | B)}$$

Put into words, Bayes' Rule states that the probability of your initial estimate X, given the original data B and some new evidence Y, is proportional to the probability of the new evidence, given the original data B and the assumption X, and to the probability of the estimate X, based only on the original data B. For example, the probability of drawing an ace from a deck of cards is 0.077

(4 cards divided by 52). If two cards are drawn at random, the probability of the second card being an ace depends on whether the first card was an ace. If it was, then the probability of the second card being an ace is 0.058 (3 divided by 52). If it wasn't, then the probability remains at 0.077.

Cox showed that quantifying beliefs numerically and requiring logical and consistent reasoning leads to exactly the same rules for beliefs as for physical probabilities. So there was no dispute about the validity of Bayes' Rule between frequentists and Bayesians. Instead, the controversy was about using subjective beliefs in data analysis rather than just using frequentist probabilities. The importance of Bayes' Rule for data analysis is apparent if proposition X is a *hypothesis* – that is, an idea or explanation – and Y is experimental *data*:

Prob (hypothesis | data and B) ~ Prob (data | hypothesis and B) ×
Prob (hypothesis |B)

The symbol ~ means that the left-hand side is proportional to the right-hand side of the equation. In other words, the probability of the hypothesis being true given the data is proportional to the probability that we would have observed the measured data if the hypothesis was true. The second factor on the right-hand side is the probability of our hypothesis, **Prob (hypothesis | B)**. This is the prior probability and represents our state of belief before we have included the measured data. By Bayes' Rule, we see that this prior probability is modified by the experimental measurements using the quantity **Prob (data | hypothesis and B)** – known as the *likelihood function*. This gives us the posterior probability, **Prob (hypothesis | data and B)**, which is our new belief in the hypothesis, after taking into account the new data. The likelihood function uses a statistical model that gives the probability of the observed data for various values of some unknown parameter. For estimating the parameters of a model we can ignore the denominator, **Prob (data | B)**, because it is just a scaling factor that does not depend explicitly on the hypothesis. However, for model comparisons, the denominator is important and is called the *evidence*.

As Sharon McGrayne shows in her book *The Theory That Would Not Die*, Bayesian reasoning about uncertainty persisted in some unlikely places, even in times when the frequentists were in the ascendancy. In 1918, Albert Whitney, who had taught insurance mathematics at the University of California, Berkeley, invented *credibility theory*, a Bayesian method for pricing insurance premiums by assigning weights to the available evidence based on its believability. In the 1930s, Cambridge geophysicist Harold Jeffreys studied earthquakes and tsunamis from a Bayesian point of view and published a classic text on the *Theory of Probability* in 1939, just before World War II broke out.

During World War II, a Bayesian approach to uncertainty played a decisive role in winning the battle against the German U-boats that were sinking vital U.K. supply ships in the North Atlantic. Alan Turing, working at the top-secret Bletchley Park code-breaking site, introduced Bayesian methods to help decipher the German Navy's messages encrypted using the Enigma machine. Soon after his arrival at Bletchley Park, Turing helped automate the process of searching through the huge number of possible Enigma settings. With mathematician Gordon Welchman and engineer Harold Keen, Turing designed a machine called the *bombe*, a high-speed electromechanical machine that tested possible Enigma wheel arrangements (Fig. 14.1), saving time by reducing the number of possible solutions. However, in the worst case it could still take up to four days to try all the 336 possible wheel positions, and such a delay meant that the information was of no use for rerouting ships away from the U-boat packs. In an attempt to reduce the number of wheel positions that needed to be searched, Turing and the team looked for what they called *cribs*, German words that they thought likely to occur in the unencrypted text. Many of these came from German weather ships, which often repeated phrases like "weather for the night" and "beacons lit as ordered." In addition, because the German operators spelled out numbers, the word *ein* (one) appeared in 90 percent of Enigma messages. Armed with such cribs, Turing invented a manual system that could reduce the number of wheel settings to be tested by the bombes from 336 to as few as 18. He called his system *Banburismus*, after the nearby town of Banbury where a printing shop produced the six-foot strips of cardboard needed to put his system into practice. To use the Banburismus system, staff at Bletchley punched holes corresponding to each intercepted message into a Banbury sheet. By putting one of the sheets on top of another, they could see when letter holes coincided on both sheets. This enabled Turing's team to guess a stretch of letters. They could update this guess as more data arrived, using Bayesian inference, which uses a combination of new information and prior beliefs to eliminate the least likely choices. To compare the probabilities of his guesses, Turing introduced a unit of measurement he called a *ban* – short for Banburismus. A tenth of a ban was called a *deciban* and, according to Jack Good, Turing's colleague at Bletchley, "A deciban is about the smallest weight of evidence perceptible to the intuition."[4] By June 1941, the Bletchley Park team could read messages to the German U-boats within an hour of their arrival.

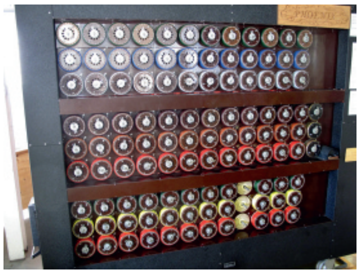


Fig. 14.1. A functioning replica of the *bombe* code-breaking machine rebuilt at Bletchley Park.

After the war, the applications of Bayesian inference multiplied rapidly. Jerome Cornfield, working at the U.S. National Institutes of Health (NIH), introduced Bayesian methods into epidemiology, the branch of medicine that studies the incidence and distribution of diseases. Using Bayes' Rule, Cornfield combined research showing the probability that someone with lung cancer was a cigarette smoker with NIH data to answer the opposite question, "What is the probability that someone who smokes will develop lung cancer?" His results showed that smokers are many times more likely to develop lung cancer than nonsmokers. At the RAND Corporation in Santa Monica, California, Fred Iklé and Albert Madansky used a Bayesian approach to estimate the probability of an accident involving nuclear weapons. Because there had only been "harmless" accidents with nuclear bombs, there was nothing that could be said about this question from a frequentist viewpoint. Their report was completed in 1958 but remained classified for more than forty years. However, the report persuaded General Curtis LeMay of the Strategic Air Command to issue orders that two people should be required to arm a nuclear weapon and also that combination locks should be installed on the warheads. A third application area was in business, where executives routinely have to make critical decisions with incomplete data and much uncertainty. At the Harvard Business School, Robert Schlaifer and Howard Raiffa introduced Bayesian methods to

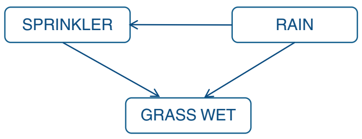


Fig. 14.2. A simple Bayesian network showing the structure of the joint probability distribution for rain, sprinkler, and grass. The diagram captures the fact that rain influences whether the sprinkler is activated, and both rain and the sprinkler influence whether the grass is wet. This is an example of a directed acyclic graph.

decision theory. Their 1961 book, *Applied Statistical Decision Theory*, contained a detailed account of Bayesian analytical methods.

The modern recognition of how Bayesian and frequentist methods can work together began in the 1980s. In 1984, Adrian Smith, a professor of statistics at the University of Nottingham in England, wrote that "efficient numerical integration procedures are the key to more widespread use of Bayesian methods."[5] Six years later, with Alan Gelfand from the University of Connecticut, Smith wrote a very influential paper showing that the difficult calculations required to apply Bayesian methods to realistic problems could be estimated using the *Monte Carlo method*. The Monte Carlo method is a forecasting technique applied in situations where statistical analysis is too difficult due to the complexity of the problem. As we have seen in Chapter 5, the method involves running multiple trials using random variables: the larger the number of trials, the better the predictions work. A technique related to the Monte Carlo method that mathematicians call a *Markov chain*, employs probability to predict sequences of events. A Markov chain, named for the Russian mathematician Andrei Markov, is a sequence of events where the probability for each event only depends on the event just before it. The combination of the two methods is known as Markov chain Monte Carlo (MCMC).

A former student of Adrian Smith's, David Spiegelhalter, working for the Medical Research Council, a government research funding agency in the United Kingdom, wrote a program for the analysis of complex statistical models using MCMC methods. Spiegelhalter's program generated random samples using a method called *Gibbs sampling*. He released his BUGS program – an acronym for Bayesian Inference Using Gibbs Sampling – in 1991. BUGS has since become one of the most widely used Bayesian software packages with more than thirty thousand downloads and applications in many different research areas ranging from geology and genetics to sociology and archaeology. Spiegelhalter has applied the Bayesian approach to clinical trials and epidemiology.

The startling growth in Bayesian applications was due both to the availability of manageable numerical methods for estimating posteriors using MCMC sampling and the widespread availability of powerful desktop computers. In our examples, we have only considered simple problems with few variables. In real problems, statisticians typically look to find relationships among large numbers of variables. At the end of the 1980s, there was a breakthrough in applying Bayesian methods. Turing Award recipient Judea Pearl (B.14.3) showed that *Bayesian networks*, graphical representations of a set of random variables and the probability of two events occurring together, were a powerful tool for performing complex Bayesian analyses. A very simple Bayesian network is shown in Figure 14.2.

B.14.3. Judea Pearl received the Turing Award in 2011 for developing a calculus for casual reasoning based on Bayesian belief networks. This new approach allowed the probabilistic prediction of future events and also the selection of a sequence of actions to achieve a given goal. His theoretical framework has given strong momentum to the renewed interest in AI among the computer science community.

As a final example of the advances made by Bayesian analysis, David Heckerman, a machine-learning researcher at Microsoft Research, says, "The whole thing about being a Bayesian is that all probability represents uncertainty and, anytime you see uncertainty, you represent it with probability. And that's a whole lot bigger than Bayes' theorem."[6] For his PhD thesis at Stanford University, Heckerman introduced Bayesian methods and graphical networks into expert systems to capture the uncertainties of expert knowledge. His "probabilistic expert system" was called Pathfinder and was used to assist medical professionals in diagnosing lymph node disease. At Microsoft

Research, Heckerman has applied Bayesian techniques to problems such as spam detection in email and troubleshooting failures in computing systems. He now leads a research team analyzing genetic data to better understand the causes of diseases such as HIV/AIDS and diabetes. His team recently performed an examination of the genomes (complete sets of DNA) of many people to find the genetic variants associated with particular diseases, using data from a Wellcome Trust study of the British population. For each of seven major diseases, the data includes genetic information about two thousand individuals with that disease. The data also include similar information for thirteen thousand individuals without any of the diseases. Using a new, computationally efficient algorithm that Heckerman's team has developed to remove false correlations, the researchers analyzed 63,524,915,020 pairs of genetic markers looking for interactions among these markers for bipolar disease, coronary artery disease, hypertension, inflammatory bowel disease (Crohn's disease), rheumatoid arthritis, and type I and type II diabetes. They processed the data from this study using twenty-seven thousand computers in the Microsoft cloud computing platform, an Internet-based service in which large numbers of processors located in a data center can be used on a pay-as-go basis. The computers ran for seventy-two hours and completed one million tasks, the equivalent of approximately 1.9 million computer hours. If the same computation had run on a typical desktop computer, the analysis would have taken twenty-five years to complete. The result was the discovery of new associations between the genome and these diseases, discoveries that could lead to breakthroughs in prevention and treatment.

Computer vision and machine learning: A state-of-the-art application

Humans find vision easy and can look at a scene and rapidly understand the objects in the scene and the context in which these objects coexist. Computer vision still has a long way to go to match human vision even though this has been a key research area in computer science since the mid-1960s. Although progress has been slow, there are now many commercial applications of computer vision algorithms, ranging from industrial inspection systems to license-plate number recognition. In the early 1990s, computer scientists developed vision-based systems to capture three-dimensional human motions. One such system could recover the three-dimensional body positions of a person moving in a special studio wearing clothing with special reflective markers, by collecting images from multiple cameras. Research also continued on algorithms that could recover three-dimensional information from video footage. However, the problems of image understanding and general object recognition remain huge challenges for computer science. Some progress has been made (Fig. 14.3), but for major advances to occur, software models of each object need to be generated from the data rather than handcrafted by the programmer. Machine learning is now recognized to be a key technology for effective object recognition. In 2001, Paul Viola and Michael Jones used machine-learning technologies to build the first object-detection framework that could provide useful detection accuracy for a variety of features. Although their system could be trained to recognize different classes of objects, they were motivated to design their

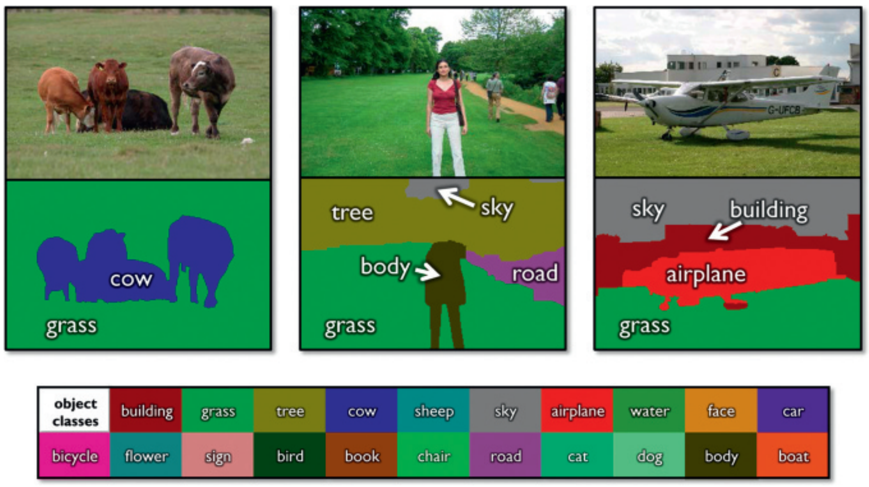


Fig. 14.3. In the last decade, progress in object recognition in natural images has been achieved with machine-learning algorithms and very large labeled training sets.

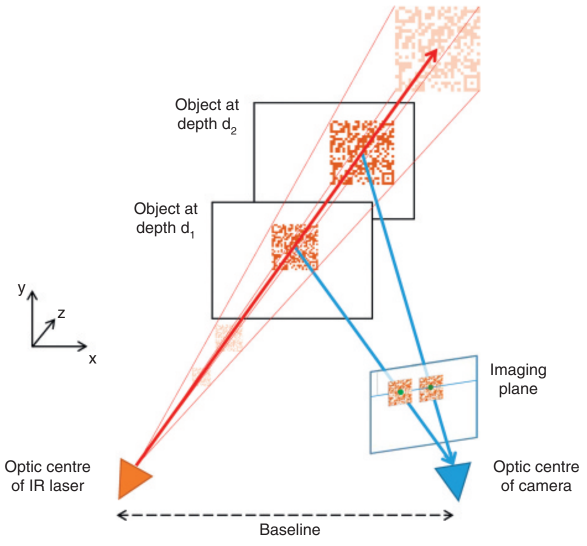


Fig. 14.4. The diagram illustrates the operation of a Kinect 3D Camera. An infrared laser illuminates the scene with a random dot pattern. By using the images of these dots, the camera sensor can determine the relative distance of objects in the scene.

framework to solve the problem of face detection. Their system is now widely used in the face detection software in digital cameras.

In 2008, the team working on the Microsoft Xbox game console met with vision researchers at Microsoft Research's laboratory in Cambridge, England. The Xbox team's ambitious goal was to develop human body–tracking software that was powerful enough to be used for playing computer games without using a game controller. Alex Kipman, from the Xbox team, had taken a new approach to the problem of three-dimensional motion capture by using depth information from an infrared three-dimensional camera. The infrared camera worked at a resolution of 320 × 240 pixels and generated images at thirty frames per second (Figs. 14.4 and 14.5). Cambridge researcher Jamie Shotton wrote:

> The depth accuracy really got me excited – you could even make out the nose and eyes on your face. Having depth information really helps for human pose

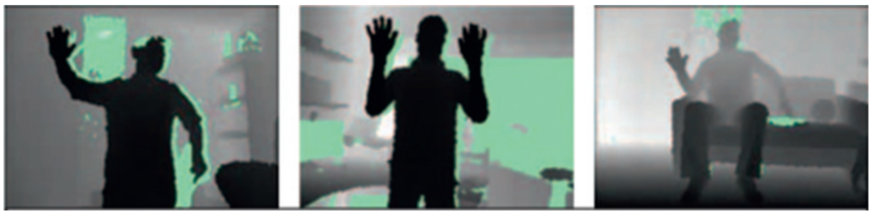


Fig. 14.5. Some example images from the Kinect camera. Nearby points are dark gray, and farther points are light gray. In the green areas, no infrared data was captured.

> estimation by removing a few big problems. You no longer have to worry about what is in the background since it is just further away. The color and texture of clothing, skin and hair are all normalized away. The size of the person is known, as the depth camera is calibrated in meters.[7]

The Xbox team had built an impressive human body–tracking system using the three-dimensional information but had been unable to make the system powerful enough for realistic game-playing situations. Shotton described the problem as follows:

> The Xbox group also came to us with a prototype human tracking algorithm they had developed. It worked by assuming it knew where you were and how fast you were moving at time t, estimating where you were going to be at time t + 1, and then refining this prediction by repeatedly comparing a computer graphics model of the human body at the prediction, to the actual observed depth image on the camera and making small adjustments. The results of this system were incredibly impressive: it could smoothly track your movements in real-time, but it had three limitations. First, you had to stand in a particular "T"-pose so it could lock on to you initially. Second, if you moved too unpredictably, it would lose track, and as soon as that happened all bets were off until you returned to the T-pose. In practice this would typically happen every five or ten seconds. Third, it only worked well if you had a similar body size and shape as the programmer who had originally designed it. Unfortunately, these limitations were all show-stoppers for a possible product.[8]

Shotton, with his colleagues Andrew Fitzgibbon and Andrew Blake, brainstormed about how they might solve these problems. The researchers knew that they needed to avoid making the assumption that, given the body position or "pose" in the previous video frame just 1/30 of a second ago, one could find the current body position by trying "nearby" poses. With rapid motions, this assumption just does not work. What was needed was a detection algorithm for a single three-dimensional image that could take the raw depth measurements and convert them into numbers that represented the body pose. However, to include all possible combinations of poses, shapes, and sizes the researchers estimated it would require approximately 10^{13} different images. This number was far too large for any matching process to run in real time on the Xbox hardware. Shotton had the idea that instead of recognizing entire natural objects, his team would create an algorithm that recognized the different body parts, such as "left hand" or "right ankle." The team designed a pattern of thirty-one different body parts and then used a *decision forest* – a collection of *decision trees* – as a classification technique to predict the probability that a given pixel belonged to a specific part of the body (Fig. 14.6). By

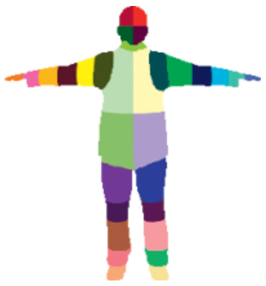


Fig. 14.6. Color-coded pattern of thirty-one different body parts used by the body pose algorithm developed by Microsoft researchers for the Kinect Xbox application.

Fig. 14.7. The 20q game from Radica uses AI technologies to guess the item you are thinking of in twenty questions or less. The game was runner-up for "Game of the Year" in 2005.

predicting these "part probabilities" from a single depth image, they were able to find accurate predictions of the three-dimensional locations of the different joints in the body. The Xbox team was then able to take these predictions and stitch them into a coherent three-dimensional "skeletal" representation of the body.

Decision trees work like the guessing game Twenty Questions (Fig. 14.7), where each question reduces the number of possible answers. For every pixel in the image, the computer asks a series of questions, such as "Is that point on the right of the image more than twelve centimeters farther away than the point under this pixel?" Based on the answer to questions like these, the program proceeds farther down the tree, asking additional questions until it can assign the pixel to a specific body part. The challenge was how best to determine the questions in the tree; the decision trees used in the final system had a depth of around twenty levels and contained nearly a million nodes. The answer was to train the system on a very large set of examples. With the Xbox team, the researchers recorded hours of video footage of actors at a motion capture studio. They filmed the actors performing actions that would be useful for gaming, such as dancing, running, fighting, driving, and so on. These data were then used to automatically animate computer graphic models of different human shapes and sizes. They then simulated the readings that the Kinect sensor would get in a simulation of these actions. The resulting training set contained millions of synthetically generated depth images and the simulated true body positions (Fig. 14.8).

The final challenge was computational. Shotton's previous work on object recognition in photographs had used training sets of only a few hundred images, and the training phase took less than a day on a single machine (Fig. 14.9). With millions of training images, the Microsoft researchers had to work out how to distribute the training on a cluster of one hundred or so computers. This distributed processing enabled them to keep the training time down to less than a day. With these advances, the researchers and the Xbox team developed very powerful skeletal tracking software and used it to create a whole variety of "controller-free" games. Microsoft launched Xbox Kinect in November 2010 with the marketing slogan "You Are the Controller." Kinect rapidly became the fastest-selling consumer electronics device in history, according to *Guinness World Records*.

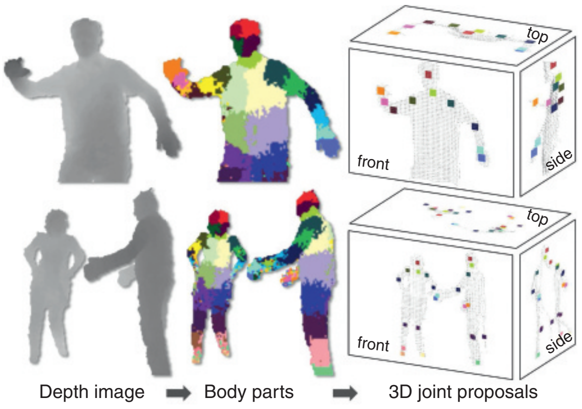


Fig. 14.8. Illustration of three-dimensional image recognition by body parts. The system learns to convert the raw depth images on the left into body part images, and then convert them to a three-dimensional stick version of the body joints.

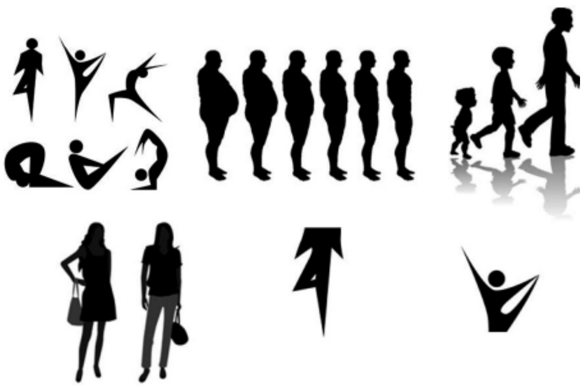


Fig. 14.9. The Kinect tracking system needs to cope with a huge variety of body poses, shapes, and sizes.

Speech and language processing

The idea of using computers to process and understand human language dates back to Turing and Shannon, two of the earliest pioneers of computing. Research into speech and language processing now covers a wide range of subdisciplines ranging from computational linguistics and natural language processing in computer science, to speech recognition and speech synthesis in electrical engineering. In the 1950s, building on earlier work on language grammars by Shannon, Noam Chomsky (B.14.4) introduced the idea of a "context-free grammar" – a mathematically precise formalism to describe how phrases in a natural language are built from smaller "blocks." Chomsky's grammar was able to capture the way in which clauses nest inside other clauses and how adjectives and adverbs are associated to nouns and verbs. Chomsky's most famous proposal is that knowledge of the formal grammar of a language explains the ability of a human to produce and interpret an infinite number of sentences from a limited set of rules and words. A second research direction in this early period was the development of probabilistic algorithms for speech and language processing. In 1952, researchers at Bell Labs built the first Automatic Speech Recognizer (ASR) system. This was a statistical system that could recognize the first ten digits with 97 percent to 99 percent accuracy – provided the speaker was male, spoke with a 350 millisecond delay between words, and the machine had been tuned to the speaker's voice profile. Otherwise the accuracy of the system fell to about 60 percent.

B.14.4. Noam Chomsky is a linguist, philosopher, cognitive scientist, and an outspoken defender of democracy and human rights. Since 1955 he has been a professor at MIT. Chomsky is a prolific author who has published more than a hundred books. His idea that children have an innate body of linguistic knowledge – the *Universal Grammar* – has been immensely influential in linguistics. His books about politics, economy, and society have often been controversial.

During the 1960s and 1970s, speech and language processing followed both these directions of research – the formal symbolic approach of Chomsky and the statistical approach of ASR systems. The symbolic approach was typically picked up by the emerging artificial intelligence (AI) community while the statistical approach was mainly followed by electrical engineers and statisticians. Two examples illustrate the different approaches of these two paradigms. As an example application from the symbolic research community, we can take Terry Winograd's SHRDLU system, written in 1972. This system was able to simulate the behavior of a robot interacting with a world of toy blocks. The program was able to accept sophisticated text commands in natural language such as "Find a block which is taller than the one you are holding and put it into the box."

Although SHRDLU was a great step forward in natural language understanding, it also showed how difficult it was to build up a computer's understanding of even a very limited world. Meanwhile, the statistical research community was making impressive improvements in speech recognition systems using Hidden Markov Models (HMM). As we have seen, a Markov model is a mathematical system with a set of possible states that undergo random transitions from one state to another, with the new state only depending on the parameters of the previous state. HMMs are systems in which the system being modeled is assumed to have an underlying Markov process but the actual Markov state transitions are not observed directly. The probabilities of the states of the hidden Markov variables have to be deduced from indirect observations. An HMM is one of the simplest Bayesian networks.

By the mid-1990s probabilistic and data-driven models had become the norm for many natural language processing applications. This trend continued in the 2000s with machine-learning techniques delivering results that were clearly superior to those of rule-based systems for almost every aspect of speech and language processing. One important example is machine translation. Instead of developing a system based on a complex set of hand-coded rules, it is now more effective to use large volumes of existing "parallel text" – the same text in both languages – as training data, and have the computer learn how words, phrases, and structures translate in context. Furthermore, with the addition of more human-produced translated text, machine translation systems continue to improve and their quality now exceeds that of the best rule-based systems. With a sufficient corpus of parallel text it is now possible to produce a machine translation system in days rather than the months it would have taken to build a rule-based system. For example, at the time of the Haiti earthquake in 2010, an English-Creole translation system for emergency aid workers was produced in less than five days by the Microsoft Research machine translation team (Fig. 14.10). It is now possible for endangered language communities to *crowdsource* their own translation system by providing enough parallel text. In this way, Microsoft's Translator Hub service has been used to produce machine translation systems for languages ranging from Hmong and Mayan, to *Star Trek*'s Klingon language, as well as translation systems using vocabularies specific to a particular industry – such as the Russian fashion industry.

The last example of advances in natural language processing concerns speech processing. HMMs enabled us to make great progress in the 1990s. Figure 14.11 shows the reduction in the Word Error Rate (WER) of such systems on a standard benchmark from the U.S. National Institute of Standards and Technology. On the "Switchboard" test data, the WER has fallen from more than 80 percent at the beginning of the 1990s to less than 30 percent by the year 2000. However, for the next decade, despite much research effort by the community, the word accuracy has remained stubbornly the same – until 2009. It was in 2009 that Geoffrey Hinton and colleagues from Toronto and Microsoft Research showed that HMMs that were pretrained using Deep Neural Networks could produce a dramatic reduction in the WER. By 2012 the WER had fallen to less than 10 percent – which is significant, because it means that computer speech processing systems are now approaching human error rates in their accuracy.

(a)

(b)

Fig. 14.10. The Haiti earthquake in 2010 reduced much of the capital, Port au Prince, to rubble. Using machine learning on the available "parallel" corpus of Creole-English texts – such as the Bible – it was possible to build a translator in less than five days. The National Palace in Port au Prince (a) before and (b) after the earthquake.

IBM's Watson and *Jeopardy!*

After IBM's success with Deep Blue and computer chess in 1997, the company had been looking for an equally audacious challenge with which to capture the public's imagination (Fig. 14.12). In 2005, after a suggestion from IBM manager Charles Lickel, the director of IBM Research, Paul Horn, tried to interest his researchers in building a machine to beat humans on *Jeopardy!* The game attracted many millions of viewers, and as Horn said, "People associated it with intelligence."[9]

The American TV quiz show *Jeopardy!* debuted on the NBC network in 1964. It is a quiz game with contestants competing to match questions to answers on a wide variety of topics. The U.S. television host Merv Griffin devised the game. He credits its strange reverse answering style, in which contestants receive clues in the form of answers and must frame their responses as questions, to a conversation with his wife:

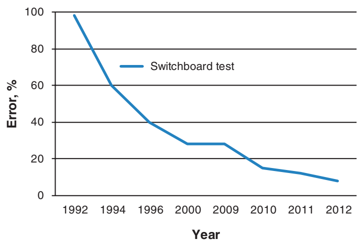
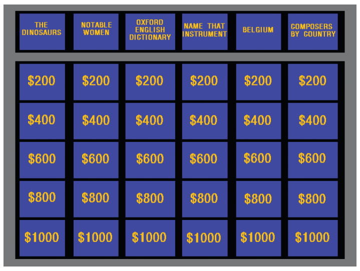


Fig. 14.11. The change in the WER with time for the U.S. National Institute of Standards "Switchboard" test. This shows the dramatic improvement made in the last few years using Deep Neural Network techniques.

> My wife Julann just came up with the idea one day when we were in a plane bringing us back to New York from Duluth. I was mulling over game show ideas, when she noted that there had not been a successful "question and answer" game on the air since the quiz show scandals. Why not do a switch, and give the answers to the contestant and let them come up with the question? She fired a couple of answers to me: "5,280" – and the question of

Fig. 14.12. The Watson computer at IBM laboratory in Yorktown Heights, New York, with the Watson logo.



THE DINOSAURS	NOTABLE WOMEN	OXFORD ENGLISH DICTIONARY	NAME THAT INSTRUMENT	BELGIUM	COMPOSERS BY COUNTRY
$200	$200	$200	$200	$200	$200
$400	$400	$400	$400	$400	$400
$600	$600	$600	$600	$600	$600
$800	$800	$800	$800	$800	$800
$1000	$1000	$1000	$1000	$1000	$1000

Fig. 14.13. A typical *Jeopardy!* game board

B.14.5. David Ferrucci graduated with a degree in biology and a PhD in computer science. His main research areas are natural language processing, knowledge representation, and discovery. He joined IBM in 1995 and led the "Watson/*Jeopardy!*" project from its inception in 2007. After an initial feasibility study, Ferucci assembled a twenty-five-member team that, in four years, developed a system that not only could "understand" spoken language but also could beat the superstar winners of the question-answering game *Jeopardy!*

course was "How many feet in a mile?" … I loved the idea, went straight to NBC with the idea, and they bought it without even looking at a pilot show.[10]

Each round of the game has six categories, each with five clues for a different amount of money (Fig. 14.13). The categories ranged from standard topics such as history, science, literature, and geography to popular culture and word games, such as puns. An example in the category of "U.S. Presidents" could be the clue "The Father of Our Country; he didn't really chop down a cherry tree." The contestant would have to reply "Who is George Washington?" As an example of wordplay, under the category "Military Ranks" could be the clue "Painful punishment practice," to which the answer is "What is corporal?" The first contestant to "buzz" after the host has read the entire clue wins the chance to have the first guess. With a correct reply, he or she wins the designated amount for the clue; a wrong reply loses the amount of the clue and allows the other contestants a chance to buzz in. The game also has three "Daily Double" clues, which allow contestants to wager a minimum of $5 up to the maximum of all their winnings, and a "Final Jeopardy!" round where the contestants write down their answer and may gamble all their winnings. If they answer correctly, they win their bet, and a successful gamble can transform the result of the game. The contestant with the highest total after the final round is the winner.

The first *Jeopardy!* superstar contestant was Ken Jennings, a computer programmer from Salt Lake City, Utah. From June until November 2004, Jennings had an amazing seventy-four-game winning streak and won more than 2.5 million dollars. Instead of the audience becoming bored, ratings for the show jumped by more than 50 percent. A key factor in Jennings's dominance was his lightning fast reflexes: he won the race to the buzzer on more than half the clues.

Paul Horn's suggestion for IBM to produce a machine to play *Jeopardy!* was controversial. It was not until a year later that he was able to persuade David Ferrucci (B.14.5), head of the Semantic Analysis and Integration Department at IBM Research, to take on the challenge. Ferrucci had many reasons for his skepticism. One of the teams he led was developing a question-answering system called Piquant, short for Practical Intelligent Question Answering Technology. Each year, there was a contest at the Text Retrieval Conference, a gathering of researchers focusing on information retrieval, in which competing teams were given a million documents on which to train their system. In these competitions, based on this very restricted knowledge base, the IBM Piquant system got two out of three questions wrong. In an initial six-month trial period, the Piquant team was trained to answer *Jeopardy!* questions using five hundred specimen clues. Although Piquant did better than a search engine–based approach that used the Web and Wikipedia, it succeeded only 30 percent of the time. From this first disappointing trial, Ferrucci concluded that he needed to adopt a much broader approach that made use of multiple AI technologies. He therefore assembled machine-learning and natural language processing experts from IBM Research and reached out to university researchers at Carnegie Mellon and MIT. Undaunted by the result of the trial, Ferucci told Horn that he would deliver a *Jeopardy!* machine that could compete with humans within twenty-four months. He gave the project the code name Blue-J. A year later, the resulting machine was christened "Watson" for IBM's first president, Thomas J. Watson.

In July 2007, Ferrucci and some IBM colleagues flew down to the Sony Pictures Studios in Culver City, California, to meet with Harry Friedman, producer of *Jeopardy!* The result was a provisional go-ahead for a human-machine match in late 2010 or early 2011. Friedman had also agreed that clues with audio or visual clips would not be used. The IBM team now had a deadline to work toward, and they began the process of "educating" Watson. They had access to twenty years of *Jeopardy!* clues from a fan website called J!Archive. From an analysis of twenty thousand clues, the team determined how often particular categories turned up. They could also study individual games, and they analyzed the seventy-four winning games of Jennings's to understand his strategy. In their "War Room" at IBM Research in Hawthorne, New York, the team plotted this information on a chart they called the "Jennings arc." He averaged more than 90 percent correct answers, and in one game Jennings won the buzzer on 75 percent of the clues. They calculated that, to beat Jennings, Watson would need to match his precision and win the race to the buzzer at least 50 percent of the time.

One of the early conclusions was that Watson did not need to know literature, music, and TV in great depth to answer the *Jeopardy!* clues. Instead, it needed to know the major facts about famous novels, brief biographies of major composers, and the stars and plotlines of popular TV shows. However, because it could not search the Web during the match, all of this information had to be loaded into Watson's memory from sources such as Wikipedia, encyclopedias, dictionaries, and newspaper articles, all in a form that the machine could understand.

The biggest obstacle for the researchers was teaching the machine to "understand" what it was supposed to look for from the cryptic *Jeopardy!* clues, which were often worded in a puzzling manner. The first algorithm to be applied was a grammatical analysis identifying nouns, verbs, adjectives, and pronouns. However, there were many possible key words that could be relevant to finding the answer, and Ferrucci and his team had to search through all the many different interpretations. Then, by using a variety of machine-learning methods and cross-checks, they assigned probabilities to a list of possible answers. All of these searches and tests took vital time, and in the game they had to come up with an answer in just a few seconds. Toward the end of 2008, Ferrucci recruited a five-person hardware team to devise a way to speed up the processing time more than a thousand-fold. How was this to be achieved? The answer was to distribute the calculations over more than two thousand processors so that Watson could explore all these paths simultaneously.

During the buildup to the contest, Watson moved on from training on sets of *Jeopardy!* clues to practice matches with previous *Jeopardy!* winners. By May 2010, Watson won against human players 65 percent of the time. The team used Watson's failures to improve and tune up their algorithms and selection criteria. They also had to insert a "profanity filter" to help Watson distinguish between polite language and profanity. After numerous glitches and many amusing mistakes, Watson's performance had climbed up the Jennings arc so that it approached the performance of experienced *Jeopardy!* winners. However, the televised match was to pit Watson against two of the very best *Jeopardy!* champions, Jennings and Brad Rutter, who had beaten Jennings in the show's "Ultimate Tournament of Champions" competition in 2005.

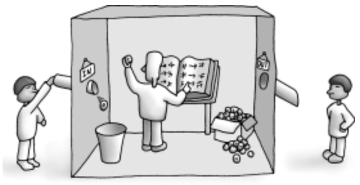
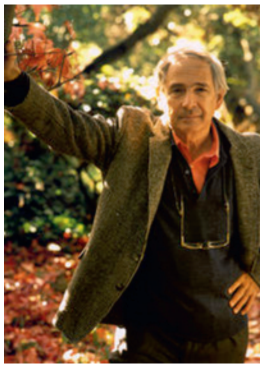


Fig. 14.14. John Searle's "Chinese room" thought experiment showed that a human can follow instructions like a computer and appear to external observers to understand Chinese, without the human having any knowledge of the language.

For the actual game, Friedman and the *Jeopardy!* team insisted that to counter Watson's advantage in electronically "pressing the buzzer," IBM would have to provide Watson with a mechanical finger to physically press a button. In addition, IBM decided that Watson needed a graphical representation for the virtual "face" of Watson. Ferrucci, mindful of the rogue AI computer HAL in Stanley Kubrick's *2001: A Space Odyssey*, suggested, "You probably want to avoid that red-eye look because when it's pulsating, it looks like HAL."[11] IBM had just launched its "Smarter Planet" initiative, a campaign to explore how information technology could promote economic growth, sustainable development, and social progress. The icon for the campaign showed planet earth with five bars, representing intelligence, radiating from it, and IBM decided that this image would be the public face of Watson. To make the game more interesting, an answer panel showed the audience – but not the other players, Watson's top five candidate answers, and the confidence the machine assigned to each one. According to Ferrucci, "This gives you a look into Watson's brain."[12]

The match took place at IBM's research center at Yorktown Heights in New York. The producers recorded the show in January and swore the audience and contestants to secrecy until after the match was broadcast in February. The match consisted of two games, and after the first game Watson had the lead. Jennings and Rutter did better in the second game, but Watson won the last Daily Double and thus won the game. In his written answer to the Final Jeopardy clue, Jennings added a postscript: "I, for one, welcome our new computer overlords."[13]

Watson won the game, but is Watson really intelligent? Ferrucci took the view that "teaching a machine to answer complex questions on a broad range of subjects would represent a notable advance, whatever the method."[14] Prior to the achievements of Deep Blue and Watson, most people would have said that activities like playing chess or competing on *Jeopardy!* required intelligence. However, just as Deep Blue playing chess employed massive computational power to search out the best chess moves, Watson used massively parallel processing to explore and rank a huge number of possible answers to the questions. The IBM Watson team did not try to model the architecture of the human brain, but instead used a host of algorithms for natural language processing and machine learning that gave the machine the ability to (mostly) correctly answer complicated questions. It made no attempt to "understand" the questions as a human would. However, from the point of view of an operational definition of intelligence as in the Turing Test, one might say that both Deep Blue and Watson are intelligent. Most experts, however, would say that Deep Blue and Watson are machines that just simulate intelligence. Such systems, sometimes called *weak AI*, can match human intelligence in a narrow field but not in broader ones. Are such systems a step on the road to *strong AI* – machines that can really think, know, and learn – or are they irrelevant to this goal? This is a question that has generated heated debate among the philosophy and computer science communities.



B.14.6. John Searle is a professor of philosophy at Berkeley in California. Within the computer science community he is best known for his controversial "Chinese room" scenario as an argument against "strong AI."

In a famous thought experiment called the "Chinese Room" (Fig. 14.14), philosopher John Searle argues (B.14.6) against the possibility of strong AI. In the thought experiment, Searle imagines someone who does not know Chinese sitting alone in a room, following directions for stringing together Chinese characters so that people outside the room think that someone inside understands and speaks Chinese. Searle explains:

> Imagine a native English speaker who knows no Chinese locked in a room full of boxes of Chinese symbols (a data base) together with a book of instructions for manipulating the symbols (the program). Imagine that people outside the room send in other Chinese symbols which, unknown to the person in the room, are questions in Chinese (the input). And imagine that by following the instructions in the program the man in the room is able to pass out Chinese symbols which are the correct answers to the questions (the output). The program enables the person in the room to pass the Turing Test for understanding Chinese but he does not understand a word of Chinese.[15]

Searle first introduced his argument in 1980, and it has generated an enormous number of responses, rebuttals, and counterrebuttals ever since. In a 2011 article, written after Watson's victory on *Jeopardy!* Searle restates his case and concludes, "Watson did not understand the questions, nor its answers, nor that some of its answers were right and some wrong, nor that it was playing a game, nor that it won – because it doesn't understand anything."[16]

Key concepts

- Bayesian network
- Bayesian inference
- Posterior beliefs
- Casual reasoning
- Human body tracking
- Universal grammar
- Statistical language translation
- Strong AI
- Chinese room

"It's a rather interesting phenomenon. Every time I press this lever, that post-graduate student breathes a sigh of relief."