7 | INFORMATION THEORY

(All I'm After Is Just a Mundane Brain)

Perhaps coming up with a theory of information and its processing is a bit like building a transcontinental railway. You can start in the east, trying to understand how agents can process anything, and head west. Or you can start in the west, with trying to understand what information is, and then head east. One hopes that these tracks will meet.

—Jon Barwise (1986)[♦]

AT THE HEIGHT OF THE WAR, in early 1943, two like-minded thinkers, Claude Shannon and Alan Turing, met daily at teatime in the Bell Labs cafeteria and said nothing to each other about their work, because it was secret.[♦] Both men had become cryptanalysts. Even Turing's presence at Bell Labs was a sort of secret. He had come over on the *Queen Elizabeth*, zigzagging to elude U-boats, after a clandestine triumph at Bletchley Park in deciphering Enigma, the code used by the German military for its critical communication (including signals to the U-boats). Shannon was working on the X System, used for encrypting voice conversations between Franklin D. Roosevelt at the Pentagon and Winston Churchill in his War Rooms. It operated by sampling the analog voice signal fifty times a second—"quantizing" or "digitizing" it—and masking it by applying a random key, which happened to bear a strong resemblance to the circuit noise with which the engineers were so familiar. Shannon did not design the system; he was assigned to analyze it theoretically and—it was hoped—prove it to be unbreakable. He accomplished this. It was clear later that these men, on their respective sides of the Atlantic, had done more than anyone else to turn cryptography from an art into a science, but for now the code makers and code breakers were not talking to each other.

With that subject off the table, Turing showed Shannon a paper he had written seven years earlier, called "On Computable Numbers," about

the powers and limitations of an idealized computing machine. They talked about another topic that turned out to be close to their hearts, the possibility of machines learning to think. Shannon proposed feeding "cultural things," such as music, to an electronic brain, and they outdid each other in brashness, Turing exclaiming once, "No, I'm not interested in developing a *powerful* brain. All I'm after is just a *mundane* brain, something like the president of the American Telephone & Telegraph Company."♦ It bordered on impudence to talk about thinking machines in 1943, when both the transistor and the electronic computer had yet to be born. The vision Shannon and Turing shared had nothing to do with electronics; it was about logic.

Can machines think? was a question with a relatively brief and slightly odd tradition—odd because machines were so adamantly physical in themselves. Charles Babbage and Ada Lovelace lay near the beginning of this tradition, though they were all but forgotten, and now the trail led to Alan Turing, who did something really outlandish: thought up a machine with ideal powers in the mental realm and showed what it could *not* do. His machine never existed (except that now it exists everywhere). It was only a thought experiment.

Running alongside the issue of what a machine could do was a parallel issue: what tasks were *mechanical* (an old word with new significance). Now that machines could play music, capture images, aim antiaircraft guns, connect telephone calls, control assembly lines, and perform mathematical calculations, the word did not seem quite so pejorative. But only the fearful and superstitious imagined that machines could be creative or original or spontaneous; those qualities were opposite to *mechanical*, which meant automatic, determined, and routine. This concept now came in handy for philosophers. An example of an intellectual object that could be called mechanical was the algorithm: another new term for something that had always existed (a recipe, a set of instructions, a step-by-step procedure) but now demanded formal recognition. Babbage and Lovelace trafficked in algorithms without naming them. The twentieth century gave algorithms a central

role—beginning here.

Turing was a fellow and a recent graduate at King's College, Cambridge, when he presented his computable-numbers paper to his professor in 1936. The full title finished with a flourish in fancy German: it was "On Computable Numbers, with an Application to the *Entscheidungsproblem*." The "decision problem" was a challenge that had been posed by David Hilbert at the 1928 International Congress of Mathematicians. As perhaps the most influential mathematician of his time, Hilbert, like Russell and Whitehead, believed fervently in the mission of rooting all mathematics in a solid logical foundation—*"In der Mathematik gibt es kein Ignorabimus,"* he declared. ("In mathematics there is no *we will not know*.") Of course mathematics had many unsolved problems, some quite famous, such as Fermat's Last Theorem and the Goldbach conjecture—statements that seemed true but had not been proved. Had not *yet* been proved, most people thought. There was an assumption, even a faith, that any mathematical truth would be provable, someday.

The *Entscheidungsproblem* was to find a rigorous step-by-step procedure by which, given a formal language of deductive reasoning, one could perform a proof automatically. This was Leibniz's dream revived once again: the expression of all valid reasoning in mechanical rules. Hilbert posed it in the form of a question, but he was an optimist. He thought or hoped that he knew the answer. It was just then, at this watershed moment for mathematics and logic, that Gödel threw his incompleteness theorem into the works. In flavor, at least, Gödel's result seemed a perfect antidote to Hilbert's optimism, as it was to Russell's. But Gödel actually left the *Entscheidungsproblem* unanswered. Hilbert had distinguished among three questions:

Is mathematics complete?

Is mathematics consistent?

Is mathematics decidable?

Gödel showed that mathematics could not be both complete and consistent but had not definitively answered the third question, at least not for all mathematics. Even though a particular closed system of formal logic must contain statements that could neither be proved nor disproved from within the system, it might conceivably be decided, as it were, by an outside referee—by external logic or rules.♦♦

Alan Turing, just twenty-two years old, unfamiliar with much of the relevant literature, so alone in his work habits that his professor worried about his becoming "a confirmed solitary,"♦ posed an entirely different question (it seemed): Are all numbers computable? This was an unexpected question to begin with, because hardly anyone had considered the idea of an *un*computable number. Most numbers that people work with, or think about, are computable by definition. The rational numbers are computable because they can be expressed as the quotient of two integers, a/b. The algebraic numbers are computable because they are solutions of polynomial equations. Famous numbers like Π and e are computable; people compute them all the time. Nonetheless Turing made the seemingly mild statement that numbers might exist that are somehow nameable, definable, and *not* computable.

What did this mean? He defined a computable number as one whose decimal expression can be calculated by finite means. "The justification," he said, "lies in the fact that the human memory is necessarily limited."♦ He also defined *calculation* as a mechanical procedure, an algorithm. Humans solve problems with intuition, imagination, flashes of insight—arguably nonmechanical calculation, or then again perhaps just computation whose steps are hidden. Turing needed to eliminate the ineffable. He asked, quite literally, what would a machine do? "According to my definition, a number is computable if its decimal can be written down by a machine."

No actual machine offered a relevant model. "Computers" were, as ever, people. Nearly all the world's computation was still performed

through the act of writing marks on paper. Turing did have one information machine for a starting point: the typewriter. As an eleven-year-old sent to boarding school he had imagined inventing one. "You see," he wrote to his parents, "the funny little rounds are letters cut out on one side slide along to the round Ⓐ along an ink pad and stamp down and make the letter, thats not nearly all though."♦ Of course, a typewriter is not automatic; it is more a tool than a machine. It does not flow a stream of language onto the page; rather, the page shifts its position space by space under the hammer, where one character is laid down after another. With this model in mind, Turing imagined another kind of machine, of the utmost purity and simplicity. Being imaginary, it was unencumbered by the real-world details one would need for a blueprint, an engineering specification, or a patent application. Turing, like Babbage, meant his machine to compute numbers, but he had no need to worry about the limitations of iron and brass. Turing did not plan ever to build his machine.

He listed the very few items his machine must possess: tape, symbols, and states. Each of these required definition.

Tape is to the Turing machine what paper is to a typewriter. But where a typewriter uses two dimensions of its paper, the machine uses only one—thus, a tape, a long strip, divided into squares. "In elementary arithmetic the two-dimensional character of the paper is sometimes used," he wrote. "But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation."♦ The tape is to be thought of as infinite: there is always more when needed. But just one square is "in the machine" at any given time. The tape (or the machine) can move left or right, to the next square.

Symbols can be written onto the tape, one per square. How many symbols could be used? This required some thought, especially to make sure the number was finite. Turing observed that words—in European languages, at least—behaved as individual symbols. Chinese, he said, "attempts to have an enumerable infinity of symbols." Arabic numerals might also be considered infinite, if 17 and 999,999,999,999,999 were

treated as single symbols, but he preferred to treat them as compound: “It is always possible to use sequences of symbols in the place of single symbols.” In fact, in keeping with the machine’s minimalist spirit, he favored the absolute minimum of two symbols: binary notation, zeroes and ones. Symbols were not only to be written but also read from the tape—“scanned” was the word Turing used. In reality, of course, no technology could yet scan symbols written on paper back into a machine, but there were equivalents: for example, punched cards, now used in tabulating machines. Turing specified one more limitation: the machine is “aware” (only the anthropomorphic word would do) of one symbol at a time—the one on the square that is in the machine.

States required more explaining. Turing used the word “configurations” and pointed out that these resembled “states of mind.” The machine has a few of these—some finite number. In any given state, the machine takes one or more actions depending on the current symbol. For example, in state *a*, the machine might move one square to the right if the current symbol is 1, or move one square to the left if the current symbol is 0, or print 1 if the current symbol is blank. In state *b*, the machine might erase the current symbol. In state *c*, if the symbol is 0 or 1, the machine might move to the right, and otherwise stop. After each action, the machine finishes in a new state, which might be the same or different. The various states used for a given calculation were stored in a table—how this was to be managed physically did not matter. The state table was, in effect, the machine’s set of instructions.

And this was all.

Turing was *programming* his machine, though he did not yet use that word. From the primitive actions—moving, printing, erasing, changing state, and stopping—larger processes were built up, and these were used again and again: “copying down sequences of symbols, comparing sequences, erasing all symbols of a given form, etc.” The machine can see just one symbol at a time, but can in effect use parts of the tape to store information temporarily. As Turing put it, “Some of the symbols written down … are just rough notes ‘to assist the memory.’ ” The tape, unfurling

to the horizon and beyond, provides an unbounded record. In this way all arithmetic lies within the machine's grasp. Turing showed how to add a pair of numbers—that is, he wrote out the necessary table of states. He showed how to make the machine print out (endlessly) the binary representation of Π. He spent considerable time working out what the machine could do and how it would accomplish particular tasks. He demonstrated that this short list covers everything a person does in computing a number. No other knowledge or intuition is necessary. Anything computable can be computed by this machine.

Then came the final flourish. Turing's machines, stripped down to a finite table of states and a finite set of input, could themselves be represented as numbers. Every possible state table, combined with its initial tape, represents a different machine. Each machine itself, then, can be described by a particular number—a certain state table combined with its initial tape. Turing was encoding his machines just as Gödel had encoded the language of symbolic logic. This obliterated the distinction between data and instructions: in the end they were all numbers. For every computable number, there must be a corresponding machine number.

Turing produced (still in his mind's eye) a version of the machine that could simulate every other possible machine—every digital computer. He called this machine *U*, for "universal," and mathematicians fondly use the name *U* to this day. It takes machine numbers as input. That is, it reads the descriptions of other machines from its tape—their algorithms and their own input. No matter how complex a digital computer may grow, its description can still be encoded on tape to be read by *U*. If a problem can be solved by any digital computer—encoded in symbols and solved algorithmically—the universal machine can solve it as well.

Now the microscope is turned onto itself. The Turing machine sets about examining every number to see whether it corresponds to a computable algorithm. Some will prove computable. Some might prove uncomputable. And there is a third possibility, the one that most interested Turing. Some algorithms might defy the inspector, causing the machine to march along, performing its inscrutable business, never coming to a halt,

never obviously repeating itself, and leaving the logical observer forever in the dark about whether it *would* halt.

By now Turing's argument, as published in 1936, has become a knotty masterpiece of recursive definitions, symbols invented to represent other symbols, numbers standing in for numbers, for state tables, for algorithms, for machines. In print it looked like this:

By combining the machines ***D*** and ***U*** we could construct a machine ***M*** to compute the sequence β'. The machine ***D*** may require a tape. We may suppose that it uses the *E*-squares beyond all symbols on *F*-squares, and that when it has reached its verdict all the rough work done by ***D*** is erased.…

We can show further that *there can be no machine**E**which, when applied with the S.D of an arbitrary machine**M***, will determine whether ***M*** ever prints a given symbol (0 say).

Few could follow it. It seems paradoxical—it *is* paradoxical—but Turing proved that some numbers are uncomputable. (In fact, most are.)

Also, because every number corresponds to an encoded proposition of mathematics and logic, Turing had resolved Hilbert's question about whether every proposition is decidable. He had proved that the *Entscheidungsproblem* has an answer, and the answer is no. An uncomputable number is, in effect, an undecidable proposition.

So Turing's computer—a fanciful, abstract, wholly imaginary machine—led him to a proof parallel to Gödel's. Turing went further than Gödel by defining the general concept of a formal system. Any mechanical procedure for generating formulas is essentially a Turing machine. *Any* formal system, therefore, must have undecidable propositions. Mathematics is not decidable. Incompleteness follows from uncomputability.

Once again, the paradoxes come to life when numbers gain the power to encode the machine's own behavior. That is the necessary

recursive twist. The entity being reckoned is fatally entwined with the entity doing the reckoning. As Douglas Hofstadter put it much later, "The thing hinges on getting this halting inspector to try to predict its own behavior when looking at itself trying to predict its own behavior when looking at itself trying to predict its own behavior when …"♦ A conundrum that at least smelled similar had lately appeared in physics, too: Werner Heisenberg's new uncertainty principle. When Turing learned about that, he expressed it in terms of self-reference: "It used to be supposed in Science that if everything was known about the Universe at any particular moment then we can predict what it will be through all the future.… More modern science however has come to the conclusion that when we are dealing with atoms and electrons we are quite unable to know the exact state of them; our instruments being made of atoms and electrons themselves."♦

A century had passed between Babbage's Analytical Engine and Turing's Universal Machine—a grand and unwieldy contraption and an elegant unreal abstraction. Turing never even tried to be a machinist. "One can picture an industrious and diligent clerk, well supplied with scratch paper, tirelessly following his instructions,"♦ as the mathematician and logician Herbert Enderton remarked years later. Like Ada Lovelace, Turing was a programmer, looking inward to the step-by-step logic of his own mind. He imagined himself as a computer. He distilled mental procedures into their smallest constituent parts, the atoms of information processing.

Alan Turing and Claude Shannon had codes in common. Turing encoded instructions as numbers. He encoded decimal numbers as zeroes and ones. Shannon made codes for genes and chromosomes and relays and switches. Both men applied their ingenuity to mapping one set of objects onto another: logical operators and electric circuits; algebraic functions and machine instructions. The play of symbols and the idea of *mapping*, in the sense of finding a rigorous correspondence between two sets, had a prominent place in their mental arsenals. This kind of coding was not meant to obscure but to illuminate: to discover that apples and

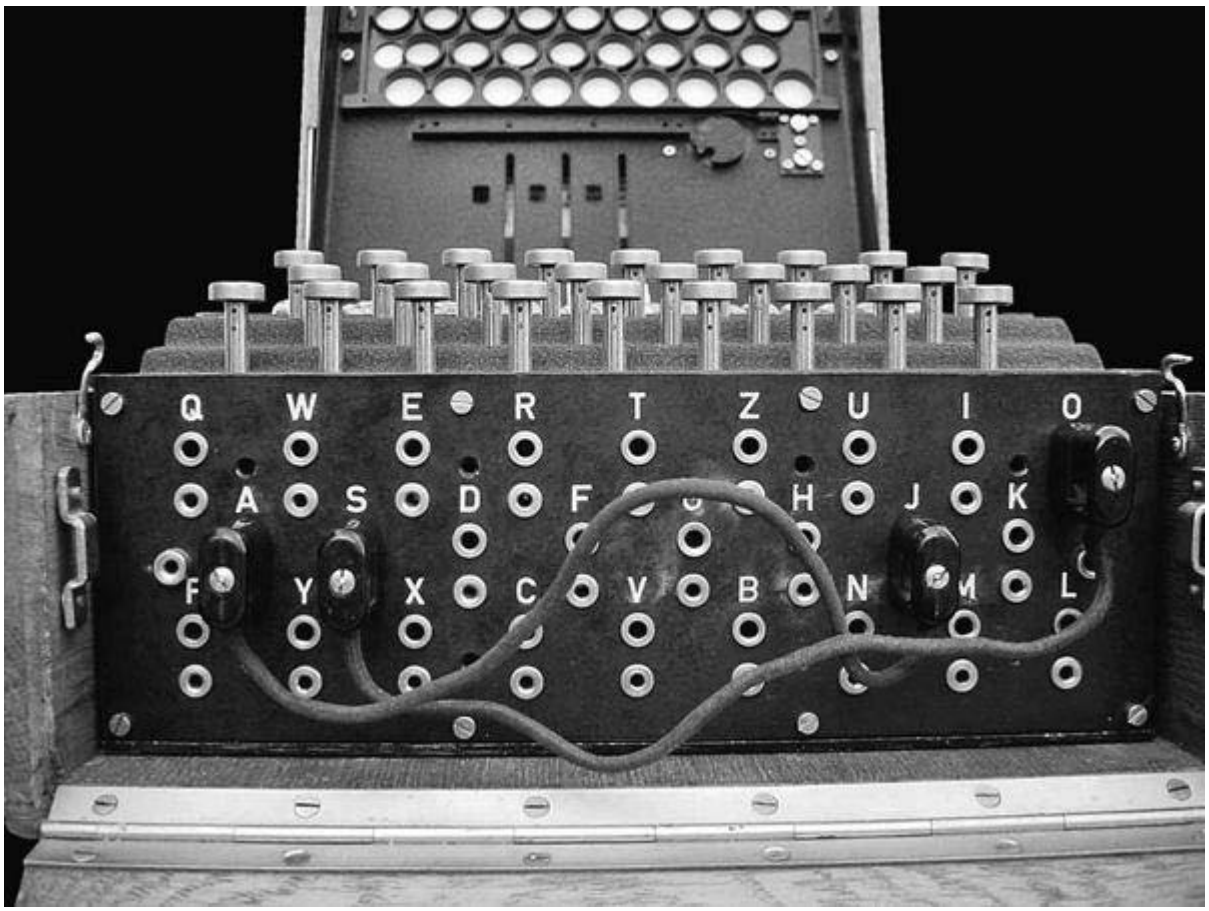
oranges were after all equivalent, or if not equivalent then fungible. The war brought both men to cryptography in its most riddling forms.

Turing's mother often asked him what use his mathematics had, and he told her as early as 1936 that he had discovered a possible application: "a lot of particular and interesting codes." He added, "I expect I could sell them to H. M. Government for quite a substantial sum, but am rather doubtful about the morality of such things."♦ Indeed, a Turing machine could *make* ciphers. But His Majesty's Government turned out to have a different problem. As war loomed, the task of reading messages intercepted from German cable and wireless traffic fell to the Government Code and Cypher School, originally part of the Admiralty, with a staff at first composed of linguists, clerks, and typists, but no mathematicians. Turing was recruited in the summer of 1938. When the Code and Cypher School evacuated from London to Bletchley Park, a country mansion in Buckinghamshire, he went along with a team that also included some champions at chess and crossword-puzzle solving. It was clear now that classical language scholarship had little to contribute to cryptanalysis.

The German system, named Enigma, employed a polyalphabetic cipher implemented by a rotor machine the size of a suitcase, with a typewriter keyboard and signal lamps. The cipher had evolved from a famous ancestor, the Vigenère cipher, thought to be unbreakable until Charles Babbage cracked it in 1854, and Babbage's mathematical insight gave Bletchley early help, as did work by Polish cryptographers who had the first hard years of experience with the Wehrmacht's signal traffic. Working from a warren known as Hut 8, Turing took the theoretical lead and solved the problem, not just mathematically but physically.

This meant building a machine to invert the enciphering of any number of Enigmas. Where his first machine was a phantasm of hypothetical tape, this one, dubbed the Bombe, filled ninety cubic feet with a ton of wire and metal leaking oil and effectively mapping the rotors of the German device onto electric circuitry. The scientific triumph at Bletchley—secret for the duration of the war and for thirty years after—had a greater effect on the outcome than even the Manhattan

Project, the real bomb. By the war's end, the Turing Bombes were deciphering thousands of military intercepts every day: processing information, that is, on a scale never before seen.



A CAPTURED ENIGMA MACHINE (Illustration credit 7.1)

Although nothing of this passed between Turing and Shannon when they met for meals at Bell Labs, they did talk indirectly about a notion of Turing's about how to measure all this *stuff*. He had watched analysts weigh the messages passing through Bletchley, some uncertain and some contradictory, as they tried to assess the probability of some fact—a particular Enigma code setting, for example, or the location of a submarine. He felt that something here needed measuring, mathematically. It was not the probability, which would traditionally be expressed as an odds ratio (such as three to two) or a number from zero to one (such as 0.6, or 60 percent). Rather, Turing cared about the data that *changed* the probability: a probability factor, something like the weight of evidence. He invented a unit he named a "ban." He found it convenient to use a

logarithmic scale, so that bans would be added rather than multiplied. With a base of ten, a ban was the weight of evidence needed to make a fact ten times as likely. For more fine-grained measurement there were "decibans" and "centibans."

Shannon had a notion along similar lines.

Working in the old West Village headquarters, he developed theoretical ideas about cryptography that helped him focus the dream he had intimated to Vannevar Bush: his "analysis of some of the fundamental properties of general systems for the transmission of intelligence." He followed parallel tracks all during the war, showing his supervisors the cryptography work and concealing the rest. Concealment was the order of the day. In the realm of pure mathematics, Shannon treated some of the same ciphering systems that Turing was attacking with real intercepts and brute hardware—for example, the specific question of the safety of Vigenère cryptograms when "the enemy knows the system being used."♦ (The Germans were using just such cryptograms, and the British were the enemy who knew the system.) Shannon was looking at the most general cases, all involving, as he put it, "discrete information." That meant sequences of symbols, chosen from a finite set, mainly letters of the alphabet but also words of a language and even "quantized speech," voice signals broken into packets with different amplitude levels. To conceal these meant substituting wrong symbols for the right ones, according to some systematic procedure in which a *key* is known to the receiver of the message, who can use it to reverse the substitutions. A secure system works even when the enemy knows the procedure, as long as the key remains secret.

The code breakers see a stream of data that looks like junk. They want to find the real signal. "From the point of view of the cryptanalyst," Shannon noted, "a secrecy system is almost identical with a noisy communication system."♦ (He completed his report, "A Mathematical Theory of Cryptography," in 1945; it was immediately classified.) The data stream is meant to look stochastic, or random, but of course it is not: if it were truly random the signal would be lost. The cipher must

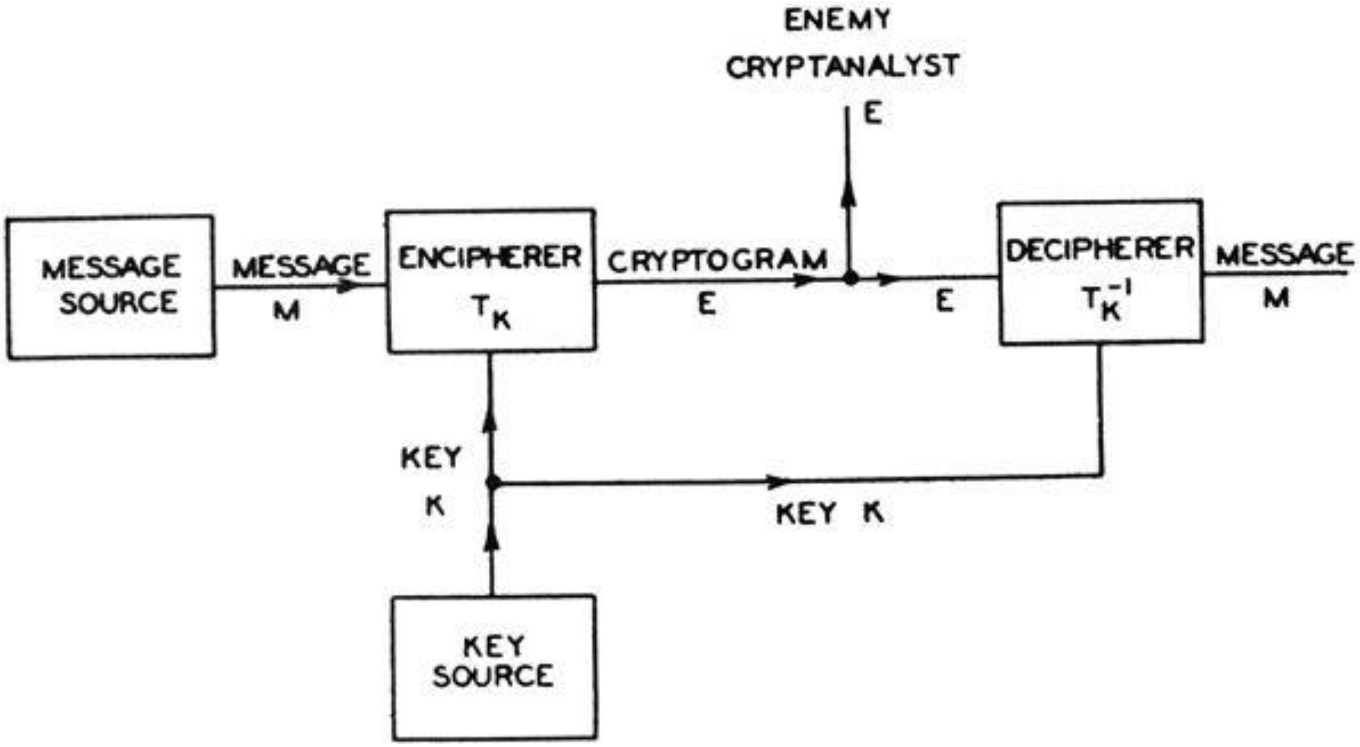
transform a patterned thing, ordinary language, into something apparently without pattern. But pattern is surprisingly persistent. To analyze and categorize the transformations of ciphering, Shannon had to understand the patterns of language in a way that scholars—linguists, for example—had never done before. Linguists had, however, begun to focus their discipline on structure in language—system to be found amid the vague billowing shapes and sounds. The linguist Edward Sapir wrote of "symbolic atoms" formed by a language's underlying phonetic patterns. "The mere sounds of speech," he wrote in 1921, "are not the essential fact of language, which lies rather in the classification, in the formal patterning.… Language, as a structure, is on its inner face the mold of thought."♦ *Mold of thought* was exquisite. Shannon, however, needed to view language in terms more tangible and countable.

Pattern, as he saw it, equals redundancy. In ordinary language, redundancy serves as an aid to understanding. In cryptanalysis, that same redundancy is the Achilles' heel. Where is this redundancy? As a simple example in English, wherever the letter *q* appears, the *u* that follows is redundant. (Or almost—it would be entirely redundant were it not for rare borrowed items like *qin* and *Qatar*.) After *q*, a *u* is expected. There is no surprise. It contributes no information. After the letter *t*, an *h* has a certain amount of redundancy, because it is the likeliest letter to appear. Every language has a certain statistical structure, Shannon argued, and with it a certain redundancy. Let us call this (he suggested) *D*. "*D* measures, in a sense, how much a text in the language can be reduced in length without losing any information."♦

Shannon estimated that English has redundancy of about 50 percent.♦ Without computers to process masses of text, he could not be sure, but his estimate proved correct. Typical passages can be shortened by half without loss of information. (*If u cn rd ths …*) With the simplest early substitution ciphers, this redundancy provided the point of first weakness. Edgar Allan Poe knew that when a cryptogram contained more *z*'s than any other letter, then *z* was probably the substitute for *e*, since *e* is the most frequent letter in English. As soon as *q* was solved, so was *u*. A code

breaker looked for recurring patterns that might match common words or letter combinations: *the, and, -tion*. To perfect this kind of frequency analysis, code breakers needed better information about letter frequencies than Alfred Vail or Samuel Morse had been able to get by examining printers' type trays, and anyway, more clever ciphers overcame this weakness, by constantly varying the substitution alphabet, so that every letter had many possible substitutes. The obvious, recognizable patterns vanished. But as long as a cryptogram retained any trace of patterning—any form or sequence or statistical regularity—a mathematician could, in theory, find a way in.

What all secrecy systems had in common was the use of a key: a code word, or phrase, or an entire book, or something even more complex, but in any case a source of characters known to both the sender and receiver—knowledge shared apart from the message itself. In the German Enigma system, the key was internalized in hardware and changed daily; Bletchley Park had to rediscover it anew each time, its experts sussing out the patterns of language freshly transformed. Shannon, meanwhile, removed himself to the most distant, most general, most theoretical vantage point. A secrecy system comprised a finite (though possibly very large) number of possible messages, a finite number of possible cryptograms, and in between, transforming one to the other, a finite number of keys, each with an associated probability. This was his schematic diagram:



(Illustration credit 7.2)

The enemy and the recipient are trying to arrive at the same target: the message. By framing it this way, in terms of mathematics and probabilities, Shannon had utterly abstracted the idea of the message from its physical details. Sounds, waveforms, all the customary worries of a Bell Labs engineer—none of these mattered. The message was seen as a choice: one alternative selected from a set. At Old North Church the night of Paul Revere's ride, the number of possible messages was two. Nowadays the numbers were almost uncountable—but still susceptible to statistical analysis.

Still in the dark about the very real and utterly relevant experience at Bletchley Park, Shannon built an edifice of algebraic methods, theorems, and proofs that gave cryptologists what they had never before possessed: a rigorous way of assessing the security of any secrecy system. He established the scientific principles of cryptography. Among other things, he proved that perfect ciphers were possible—"perfect" meaning that even an infinitely long captured message would not help a code breaker ("the enemy is no better off after intercepting any amount of material than before"♦). But as he gave, so he took away, because he also proved that the requirements were so severe as to make them practically useless. In a

perfect cipher, all keys must be equally likely, in effect, a random stream of characters; each key can be used only once; and, worst of all, each key must be as long as the entire message.

Also in this secret paper, almost in passing, Shannon used a phrase he had never used before: "information theory."

First Shannon had to eradicate "meaning." The germicidal quotation marks were his. "The 'meaning' of a message is generally irrelevant," he proposed cheerfully.♦

He offered this provocation in order to make his purpose utterly clear. Shannon needed, if he were to create a theory, to hijack the word *information*. " 'Information' here," he wrote, "although related to the everyday meaning of the word, should not be confused with it." Like Nyquist and Hartley before him, he wished to leave aside "the psychological factors" and focus only on "the physical." But if information was divorced from semantic content, what was left? A few things could be said, and at first blush they all sounded paradoxical. Information is uncertainty, surprise, difficulty, and entropy:

"Information is closely associated with uncertainty." Uncertainty, in turn, can be measured by counting the number of possible messages. If only one message is possible, there is no uncertainty and thus no information.

Some messages may be likelier than others, and information implies surprise. Surprise is a way of talking about probabilities. If the letter following *t* (in English) is *h*, not so much information is conveyed, because the probability of *h* was relatively high.

"What is significant is the difficulty in transmitting the message from one point to another." Perhaps this seemed backward, or tautological, like defining mass in terms of the force needed to move an object. But then, mass *can* be defined that way.

Information is entropy. This was the strangest and most powerful notion of all. Entropy—already a difficult and poorly understood concept—is a measure of disorder in thermodynamics, the science of heat and energy.

Fire control and cryptography aside, Shannon had been pursuing this haze of ideas all through the war. Living alone in a Greenwich Village apartment, he seldom socialized with his colleagues, who mainly worked now in the New Jersey headquarters, while Shannon preferred the old West Street hulk. He did not have to explain himself. His war work got him deferred from military service and the deferment continued after the war ended. Bell Labs was a rigorously male enterprise, but in wartime the computing group, especially, badly needed competent staff and began hiring women, among them Betty Moore, who had grown up on Staten Island. It was like a typing pool for math majors, she thought. After a year she was promoted to the microwave research group, in the former Nabisco building—the "cracker factory"—across West Street from the main building. The group designed tubes on the second floor and built them on the first floor and every so often Claude wandered over to visit. He and Betty began dating in 1948 and married early in 1949. Just then he was the scientist everyone was talking about.

THE WEST STREET HEADQUARTERS OF BELL LABORATORIES, WITH TRAINS OF THE HIGH LINE RUNNING THROUGH

Few libraries carried *The Bell System Technical Journal*, so researchers heard about "A Mathematical Theory of Communication" the traditional way, by word of mouth, and obtained copies the traditional way, by writing directly to the author for an offprint. Many scientists used preprinted postcards for such requests, and these arrived in growing volume over the next year. Not everyone understood the paper. The mathematics was difficult for many engineers, and mathematicians meanwhile lacked the engineering context. But Warren Weaver, the director of natural sciences for the Rockefeller Foundation uptown, was

already telling his president that Shannon had done for communication theory "what Gibbs did for physical chemistry."♦ Weaver had headed the government's applied mathematics research during the war, supervising the fire-control project as well as nascent work in electronic calculating machines. In 1949 he wrote up an appreciative and not too technical essay about Shannon's theory for *Scientific American*, and late that year the two pieces—Weaver's essay and Shannon's monograph—were published together as a book, now titled with a grander first word *The Mathematical Theory of Communication*. To John Robinson Pierce, the Bell Labs engineer who had been watching the simultaneous gestation of the transistor and Shannon's paper, it was the latter that "came as a bomb, and something of a delayed action bomb."♦

Where a layman might have said that the fundamental problem of communication is to make oneself understood—to convey meaning—Shannon set the stage differently:

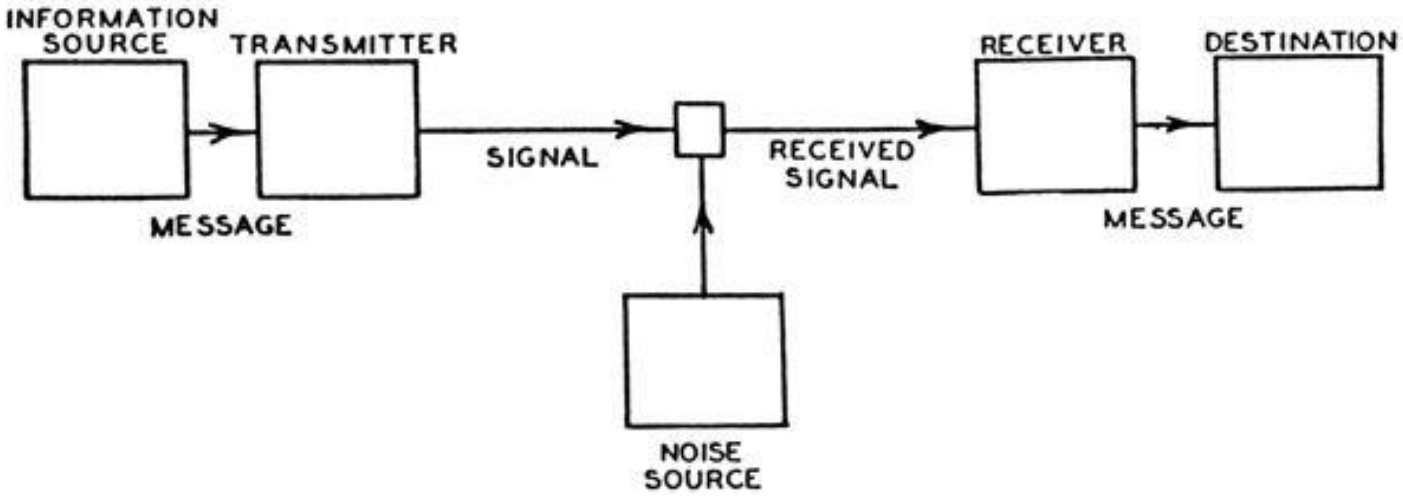
> The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.♦

"Point" was a carefully chosen word: the origin and destination of a message could be separated in space or in time; information storage, as in a phonograph record, counts as a communication. Meanwhile, the message is not created; it is selected. It is a choice. It might be a card dealt from a deck, or three decimal digits chosen from the thousand possibilities, or a combination of words from a fixed code book. He could hardly overlook meaning altogether, so he dressed it with a scientist's definition and then showed it the door:

> Frequently the messages have *meaning;* that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem.

Nonetheless, as Weaver took pains to explain, this was not a narrow view of communication. On the contrary, it was all-encompassing: "not only written and oral speech, but also music, the pictorial arts, the theatre, the ballet, and in fact all human behavior." Nonhuman as well: why should machines not have messages to send?

Shannon's model for communication fit a simple diagram—essentially the same diagram, by no coincidence, as in his secret cryptography paper.



(Illustration credit 7.3)

A communication system must contain the following elements:

The information source is the person or machine generating the message, which may be simply a sequence of characters, as in a telegraph or teletype, or may be expressed mathematically as functions—$f(x, y, t)$—of time and other variables. In a complex example like color television, the components are three functions in a three-dimensional continuum, Shannon noted.

The transmitter "operates on the message in some way"—that is, *encodes* the message—to produce a suitable signal. A telephone converts sound pressure into analog electric current. A telegraph encodes characters in dots, dashes, and spaces. More complex messages may be sampled, compressed, quantized, and interleaved.

The channel: “merely the medium used to transmit the signal.”

The receiver inverts the operation of the transmitter. It decodes the message, or reconstructs it from the signal.

The destination “is the person (or thing)” at the other end.

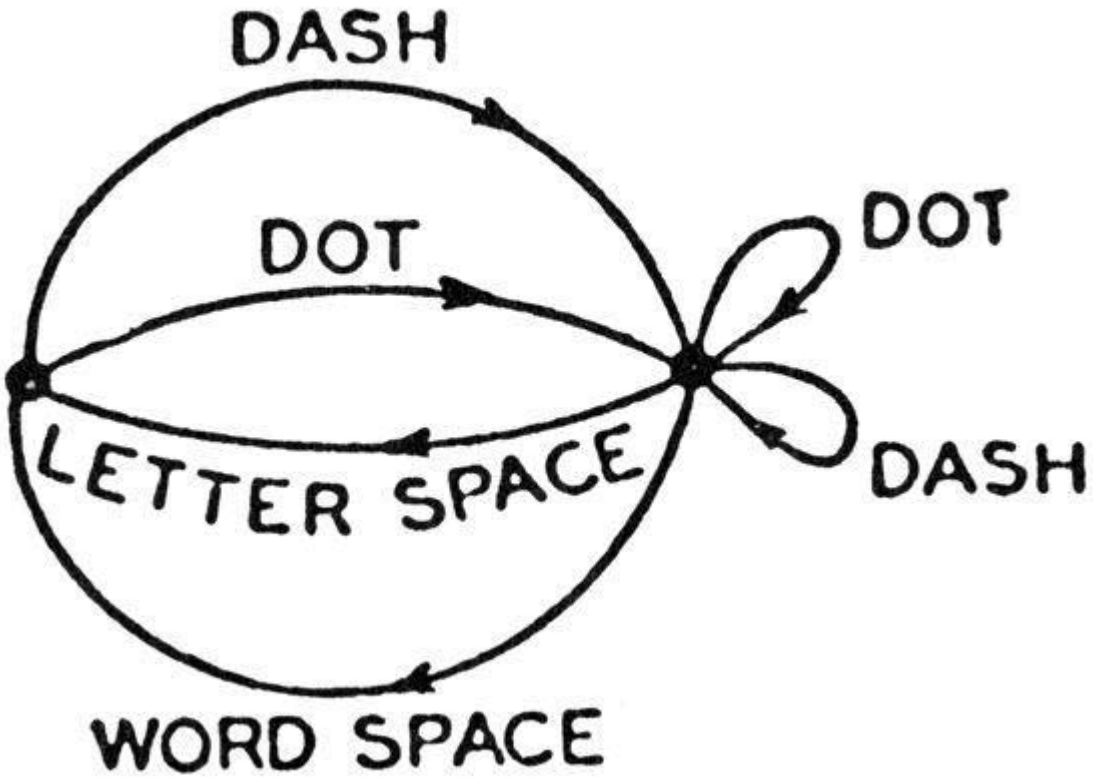
In the case of ordinary speech, these elements are the speaker’s brain, the speaker’s vocal cords, the air, the listener’s ear, and the listener’s brain.

As prominent as the other elements in Shannon’s diagram—because for an engineer it is inescapable—is a box labeled “Noise Source.” This covers everything that corrupts the signal, predictably or unpredictably: unwanted additions, plain errors, random disturbances, static, “atmospherics,” interference, and distortion. An unruly family under any circumstances, and Shannon had two different types of systems to deal with, continuous and discrete. In a discrete system, message and signal take the form of individual detached symbols, such as characters or digits or dots and dashes. Telegraphy notwithstanding, continuous systems of waves and functions were the ones facing electrical engineers every day. Every engineer, when asked to push more information through a channel, knew what to do: boost the power. Over long distances, however, this approach was failing, because amplifying a signal again and again leads to a crippling buildup of noise.

Shannon sidestepped this problem by treating the signal as a string of discrete symbols. Now, instead of boosting the power, a sender can overcome noise by using extra symbols for error correction—just as an African drummer makes himself understood across long distances, not by banging the drums harder, but by expanding the verbosity of his discourse. Shannon considered the discrete case to be more fundamental in a mathematical sense as well. And he was considering another point: that treating messages as discrete had application not just for traditional communication but for a new and rather esoteric subfield, the theory of computing machines.

So back he went to the telegraph. Analyzed precisely, the telegraph did not use a language with just two symbols, dot and dash. In the real

world telegraphers used dot (one unit of "line closed" and one unit of "line open"), dash (three units, say, of line closed and one unit of line open), and also two distinct spaces: a letter space (typically three units of line open) and a longer space separating words (six units of line open). These four symbols have unequal status and probability. For example, a space can never follow another space, whereas a dot or dash can follow anything. Shannon expressed this in terms of *states*. The system has two states: in one, a space was the previous symbol and only a dot or dash is allowed, and the state then changes; in the other, any symbol is allowed, and the state changes only if a space is transmitted. He illustrated this as a graph:



(Illustration credit 7.4)

This was far from a simple, binary system of encoding. Nonetheless Shannon showed how to derive the correct equations for information content and channel capacity. More important, he focused on the effect of the statistical structure of the language of the message. The very existence

of this structure—the greater frequency of *e* than *q*, of *th* than *xp*, and so forth—allows for a saving of time or channel capacity.

This is already done to a limited extent in telegraphy by using the shortest channel sequence, a dot, for the most common English letter E; while the infrequent letters, Q, X, Z are represented by longer sequences of dots and dashes. This idea is carried still further in certain commercial codes where common words and phrases are represented by four- or five-letter code groups with a considerable saving in average time. The standardized greeting and anniversary telegrams now in use extend this to the point of encoding a sentence or two into a relatively short sequence of numbers.♦

To illuminate the structure of the message Shannon turned to some methodology and language from the physics of stochastic processes, from Brownian motion to stellar dynamics. (He cited a landmark 1943 paper by the astrophysicist Subrahmanyan Chandrasekhar in *Reviews of Modern Physics*.♦) A stochastic process is neither deterministic (the next event can be calculated with certainty) nor random (the next event is totally free). It is governed by a set of probabilities. Each event has a probability that depends on the state of the system and perhaps also on its previous history. If for *event* we substitute *symbol*, then a natural written language like English or Chinese is a stochastic process. So is digitized speech; so is a television signal.

Looking more deeply, Shannon examined statistical structure in terms of how much of a message influences the probability of the next symbol. The answer could be none: each symbol has its own probability but does not depend on what came before. This is the first-order case. In the second-order case, the probability of each symbol depends on the symbol immediately before, but not on any others. Then each two-character combination, or digram, has its own probability: *th* greater than *xp*, in English. In the third-order case, one looks at trigrams, and so forth. Beyond that, in ordinary text, it makes sense to look at the level of

words rather than individual characters, and many types of statistical facts come into play. Immediately after the word *yellow*, some words have a higher probability than usual and others virtually zero. After the word *an*, words beginning with consonants become exceedingly rare. If the letter *u* ends a word, the word is probably *you.* If two consecutive letters are the same, they are probably *ll, ee, ss*, or *oo*. And structure can extend over long distances: in a message containing the word *cow*, even after many other characters intervene, the word *cow* is relatively likely to occur again. As is the word *horse*. A message, as Shannon saw, can behave like a dynamical system whose future course is conditioned by its past history.

To illustrate the differences between these different orders of structure, he wrote down—computed, really—a series of "approximations" of English text. He used an alphabet of twenty-seven characters, the letters plus a space between words, and generated strings of characters with the help of a table of random numbers. (These he drew from a book newly published for such purposes by Cambridge University Press: 100,000 digits for three shillings nine pence, and the authors "have furnished a guarantee of the random arrangement."♦) Even with random numbers presupplied, working out the sequences was painstaking. The sample texts looked like this:

"Zero-order approximation"—that is, random characters, no structure or correlations.

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ

FFJEYVKCQSGHYD GPAAMKBZAACIBZLHJGD.

First order—each character is independent of the rest, but the frequencies are those expected in English: more *e*'s and *t*'s, fewer *z*'s and *j*'s, and the word lengths look realistic.

OCRO HLI RGWR NIMILWIS EU LL NBNESEBYA

TH EEI ALHENHTTPA OOBTTVA NAH BRL.

Second order—the frequencies of each character match English and so also do the frequencies of each digram, or letter pair. (Shannon found the necessary statistics in tables constructed for use by code breakers.[♦] The most common digram in English is *th*, with a frequency of 168 per thousand words, followed by *he, an, re*, and *er*. Quite a few digrams have zero frequency.)

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN

D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY

TOBESEACE CTISBE.

Third order—trigram structure.

IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID

PONDENOME OF DEMONSTURES OF THE REPTAGIN IS

REGOACTIONA OF CRE.

First-order word approximation.

REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN

DIFFERENT NATURAL HERE HE THE A IN CAME THE TO

OF TO EXPERT GRAY COME TO FURNISHES THE LINE

MESSAGE HAD

BE THESE.

Second-order word approximation—now pairs of words appear in the expected frequency, so we do not see “a in” or “to of.”

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH

WRITER THAT THE CHARACTER OF THIS POINT IS

THEREFORE ANOTHER METHOD FOR THE LETTERS THAT

THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN

UNEXPECTED.

These sequences increasingly “look” like English. Less subjectively, it turns out that touch typists can handle them with increasing speed—another indication of the ways people unconsciously internalize a language’s statistical structure.

Shannon could have produced further approximations, given enough time, but the labor involved was becoming enormous. The point was to represent a message as the outcome of a process that generated events with discrete probabilities. Then what could be said about the amount of information, or the rate at which information is generated? For each event, the possible choices each have a known probability (represented as p_1, p_2, p_3, and so on). Shannon wanted to define the measure of information (represented as H) as the measure of uncertainty: “of how much ‘choice’ is involved in the selection of the event or of how uncertain we are of the outcome.”♦ The probabilities might be the same or different, but generally more choices meant more uncertainty—more information. Choices might be broken down into successive choices, with their own probabilities, and

the probabilities had to be additive; for example, the probability of a particular digram should be a weighted sum of the probabilities of the individual symbols. When those probabilities were equal, the amount of information conveyed by each symbol was simply the logarithm of the number of possible symbols—Nyquist and Hartley's formula:

$$H = n \log s$$

For the more realistic case, Shannon reached an elegant solution to the problem of how to measure information as a function of probabilities—an equation that summed the probabilities with a logarithmic weighting (base 2 was most convenient). It is the average logarithm of the improbability of the message; in effect, a measure of unexpectedness:

$$H = -\Sigma\, p_i \log_2 p_i$$

where p_i is the probability of each message. He declared that we would be seeing this again and again: that quantities of this form "play a central role in information theory as measures of information, choice, and uncertainty." Indeed, H is ubiquitous, conventionally called the entropy of a message, or the Shannon entropy, or, simply, the information.

A new unit of measure was needed. Shannon said: "The resulting units may be called binary digits, or more briefly, *bits*."[♦] As the smallest possible quantity of information, a bit represents the amount of uncertainty that exists in the flipping of a coin. The coin toss makes a choice between two possibilities of equal likelihood: in this case p_1 and p_2 each equal ½ the base 2 logarithm of ½ is −1; so $H = 1$ bit. A single character chosen randomly from an alphabet of 32 conveys more information: 5 bits, to be exact, because there are 32 possible messages and the logarithm of 32 is 5. A string of 1,000 such characters carries 5,000 bits—not just by simple multiplication, but because the amount of information represents the amount of uncertainty: the number of possible choices. With 1,000 characters in a 32-character alphabet, there are 32^{1000} possible messages, and the logarithm of that number is 5,000.

This is where the statistical structure of natural languages reenters the picture. If the thousand-character message is known to be English text, the number of possible messages is smaller—*much* smaller. Looking at correlations extending over eight letters, Shannon estimated that English has a built-in redundancy of about 50 percent: that each new character of a message conveys not 5 bits but only about 2.3. Considering longer-range statistical effects, at the level of sentences and paragraphs, he raised that estimate to 75 percent—warning, however, that such estimates become "more erratic and uncertain, and they depend more critically on the type of text involved."♦ One way to measure redundancy was crudely empirical: carry out a psychology test with a human subject. This method "exploits the fact that anyone speaking a language possesses, implicitly, an enormous knowledge of the statistics of the language."

Familiarity with the words, idioms, clichés and grammar enables him to fill in missing or incorrect letters in proof-reading, or to complete an unfinished phrase in conversation.

He might have said "her," because in point of fact his test subject was his wife, Betty. He pulled a book from the shelf (it was a Raymond Chandler detective novel, *Pickup on Noon Street*), put his finger on a short passage at random, and asked Betty to start guessing the letter, then the next letter, then the next. The more text she saw, of course, the better her chances of guessing right. After "A SMALL OBLONG READING LAMP ON THE" she got the next letter wrong. But once she knew it was *D,* she had no trouble guessing the next three letters. Shannon observed, "The errors, as would be expected, occur most frequently at the beginning of words and syllables where the line of thought has more possibility of branching out."

Quantifying predictability and redundancy in this way is a backward way of measuring information content. If a letter can be guessed from what comes before, it is redundant; to the extent that it is redundant, it provides no new information. If English is 75 percent redundant, then a

thousand-letter message in English carries only 25 percent as much information as one thousand letters chosen at random. Paradoxical though it sounded, random messages carry *more* information. The implication was that natural-language text could be encoded more efficiently for transmission or storage.

Shannon demonstrated one way to do this, an algorithm that exploits differing probabilities of different symbols. And he delivered a stunning package of fundamental results. One was a formula for channel capacity, the absolute speed limit of any communication channel (now known simply as the Shannon limit). Another was the discovery that, within that limit, it must always be possible to devise schemes of error correction that will overcome any level of noise. The sender may have to devote more and more bits to correcting errors, making transmission slower and slower, but the message will ultimately get through. Shannon did not show how to design such schemes; he only proved that it was possible, thereby inspiring a future branch of computer science. "To make the chance of error as small as you wish? Nobody had thought of that," his colleague Robert Fano recalled years later. "How he got that insight, how he came to believe such a thing, I don't know. But almost all modern communication theory is based on that work."♦ Whether removing redundancy to increase efficiency or adding redundancy to enable error correction, the encoding depends on knowledge of the language's statistical structure to do the encoding. Information cannot be separated from probabilities. A bit, fundamentally, is always a coin toss.

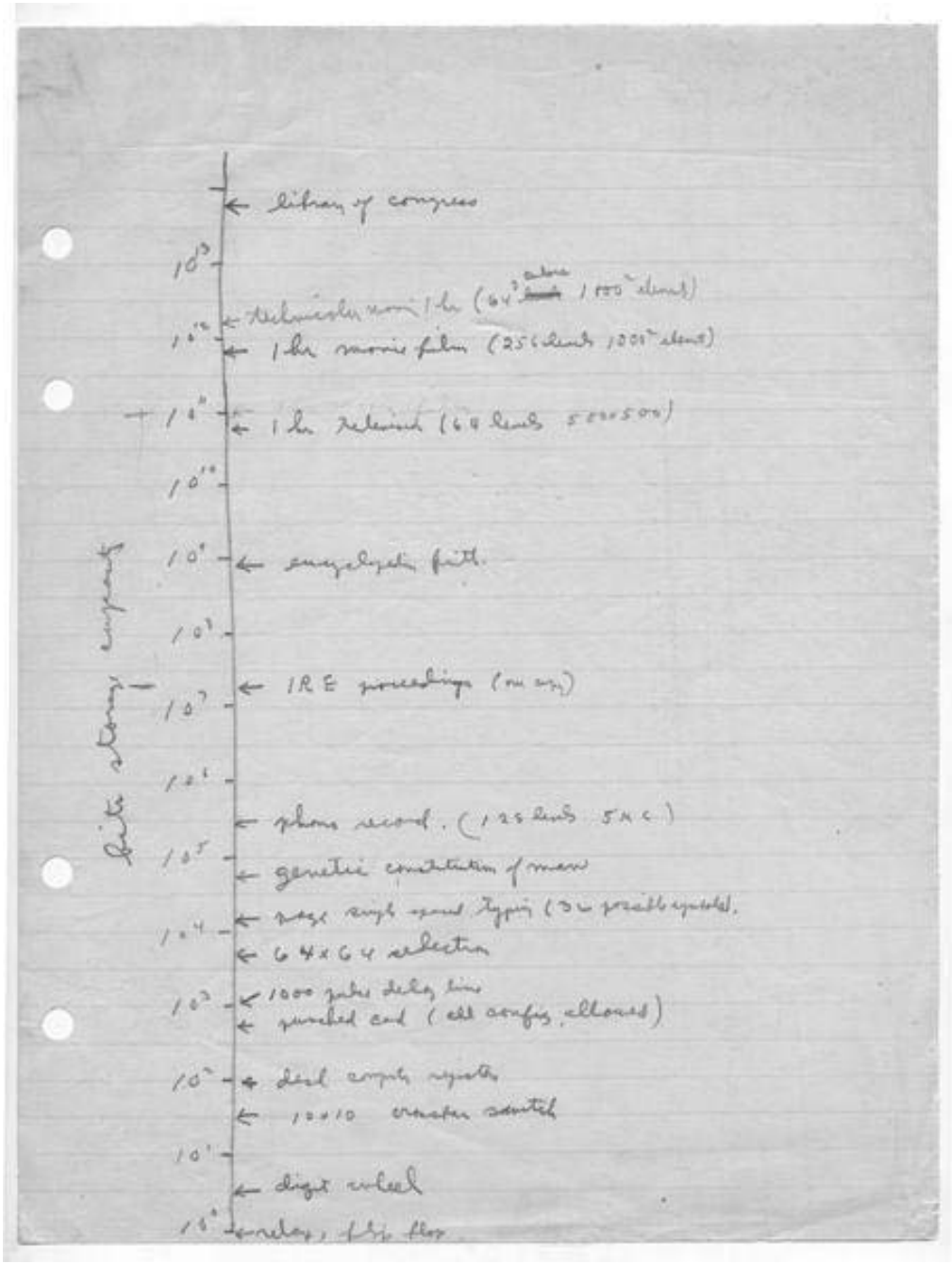
If the two sides of a coin were one way of representing a bit, Shannon offered a more practical hardware example as well:

A device with two stable positions, such as a relay or a flip-flop circuit, can store one bit of information. N such devices can store N bits, since the total number of possible states is 2^N and $\log_2 2^N = N$.

Shannon had seen devices—arrays of relays, for example—that could store hundreds, even thousands of bits. That seemed like a great

many. As he was finishing his write-up, he wandered one day into the office of a Bell Labs colleague, William Shockley, an Englishman in his thirties. Shockley belonged to a group of solid-state physicists working on alternatives to vacuum tubes for electronics, and sitting on his desk was a tiny prototype, a piece of semiconducting crystal. "It's a solid-state amplifier," Shockley told Shannon.♦ At that point it still needed a name.

One day in the summer of 1949, before the book version of *The Mathematical Theory of Communication* appeared, Shannon took a pencil and a piece of notebook paper, drew a line from top to bottom, and wrote the powers of ten from 10^0 to 10^{13}. He labeled this axis "bits storage capacity."♦ He began listing some items that might be said to "store" information. A digit wheel, of the kind used in a desktop adding machine—ten decimal digits—represents just over 3 bits. At just under 10^3 bits, he wrote "punched card (all config. allowed)." At 10^4 he put "page single spaced typing (32 possible symbols)." Near 10^5 he wrote something offbeat: "genetic constitution of man." There was no real precedent for this in current scientific thinking. James D. Watson was a twenty-one-year-old student of zoology in Indiana; the discovery of the structure of DNA lay several years in the future. This was the first time anyone suggested the genome was an information store measurable in bits. Shannon's guess was conservative, by at least four orders of magnitude. He thought a "phono record (128 levels)" held more information: about 300,000 bits. To the 10 million level he assigned a thick professional journal (*Proceedings of the Institute of Radio Engineers*) and to 1 billion the *Encyclopaedia Britannica*. He estimated one hour of broadcast television at 10^{11} bits and one hour of "technicolor movie" at more than a trillion. Finally, just under his pencil mark for 10^{14}, 100 trillion bits, he put the largest information stockpile he could think of: the Library of Congress.



(Illustration credit 7.5)

♦ Toward the end of his life Gödel wrote, “It was only by Turing’s work that it became completely clear, that my proof is applicable to every formal system containing arithmetic.”

♦ “not considering statistical structure over greater distances than about eight letters.”