4

Language

This chapter could serve as the main part of a one-term course designed to draw humanity majors into the world of computers. It only requires programming facility at an introductory level and deals mostly with subjects that are of general familiarity. The material in Sections 4.12–4.15 on information theory and entropy may appeal primarily to people with an interest in communication theory. However, the material on entropy per character in Section 4.13 has an important bearing on many fields of general interest (e.g., anthropology, error correction, cryptography, and even the detection of extra-terrestrial civilization.) Although most of Section 4.13 deals with the qualitative meaning of Shannon's definition of entropy, all that is really needed for the remainder of the chapter is the mathematical definition of the entropy per character in various statistical orders for printed text. (See Eqs. (24), (43), (44).) The general reader ought at least to have a look at Section 4.14 on anthropology, Section 4.15 on bit compression, Section 4.21 on the detection of decipherable messages, and Section 4.22 on the Voynich manuscript. However, if the reader feels determined to avoid dealing with entropy in language, he (or she) should take the following route:

Sections 4.1 through 4.12 (the monkey problem through language and style identification); read Section 4.15 (on bit compression), and Sections 4.16 through 4.20 (on cryptography). In this path, the hardest programming difficulties occur in the second-order literary crypt solver discussed at the end of Section 4.20. (The latter could easily be omitted in a "short course.")

Finally, it is worth noting that much of the material in this chapter is of a sufficiently general nature to warrant reading on its own merits—even if you aren't interested in computing. In this case, just skip all of the programming statements.

Man's compulsion to communicate generally shows up early in a computing course. One frequently encounters student programs containing long strings of alphanumeric characters to be printed at various branch points ranging from dull factual statements (e.g., "THE NUMBER OF PRIMES LESS THAN ... IS ... ") to messages of flamboyant bravado (e.g., "THANKS FOR THE GAME YOU CLOD AND BETTER LUCK NEXT TIME"). This kind of thing is much more entertaining if you cannot tell ahead of time from the program just exactly what the computer is going to say. We shall therefore consider an alternative approach to the problem, both to make the form of the conversation more interesting and to illustrate some fundamental statistical properties of written languages in general.

There has been a sporadic preoccupation with the statistical aspects of language throughout much of recorded history. Recent quantitative manifestations of this interest have mostly gone under the heading *information theory*. From the advent of the telegraph on, there has been an increasing concentration on the mathematical analysis of communication problems—an interest reflected by the early papers of Nyquist and Hartley, through the more generally known work of Shannon. It is not surprising to find that much of this research was supported by a company in the communications field (Bell Laboratories). In addition, the subject was stimulated by government concern with transmission and detection of "secret" messages during and in between wars. More recently, the insatiable appetite for data transmission shown by the computer field as a whole has elicited still more concern, if not outright anxiety. The contemporary transmission unit is megabits per second.

Activity has also gone on with those less motivated by practical application, for interest in the statistical aspects of language is clearly lurking at least subliminally below the surface in most of us. In fact, it is probably not entirely accidental that the foremost American contributor to statistical mechanics, Josiah Willard Gibbs, was himself the son of a philologist. Gibbs the elder was something of a pioneer in urging that language should be the object of scientific study from a correlative point of view (see Gibbs, 1857).

4.1 Monkeys at the Typewriters

Nearly everyone knows that if enough monkeys were allowed to pound away at typewriters for enough time, all the great works of literature would result. The universal appeal of this notion to human imagination is demonstrated by the wide variety of circumstances in which it appears. For example, the basic concept involved has been used in a contemporary nightclub act by Bob Newhart, in a series of popular lectures on statistical mechanics given about 50 years ago by Sir Arthur Eddington, and in the discourses on religious philosophy by the seventeenth-century archbishop John Tillotson (1630–1694). Elaborate fantasies on this general theme have been given by Maloney (1945) and Vonnegut (1950). Kurt Vonnegut's treatment is probably the first one that implies a computer simulation of the problem.

The earliest specific use of the basic concept known to the present author is to be found in the *Maxims and Discourses* of Archbishop Tillotson, published posthumously in 1719. In his "Answer to the Epicurean System," Tillotson applied the notion to the creation of poetry, prose, entire books, portrait painting, and even the creation of Man and the World. He then went on to imply that the improbability of these events occurring through chance constitutes an argument for the existence of God. His original statement of the problem is so profoundly moving that we have reproduced the paragraph in entirety in Fig. 4-1.

Most contemporary use of the concept is traceable to the Gifford Lectures presented by Eddington at Cambridge in 1927. Here Eddington first brought

In Anſwer to the Epicurean *Syſtem, he argues*] How often might a Man, after he had jumbled a Set of Letters in a Bag, fling them out upon the Ground before they would fall into an exact Poem, yea or ſo much as make a good Diſcourſe in Proſe? And may not a little *Book* be as eaſily made by Chance, as this great *Volume* of the World? How long might a Man be in ſprinkling Colours upon a Canvas with a careleſs Hand, before they could happen to make the exact Picture of a Man? And is a Man eaſier made by Chance than his Picture? How long might twenty thouſand *blind Men*, which ſhould be ſent out from the ſeveral remote Parts of *England*, wander up and down before they would all meet upon *Salisbury-Plains*, and fall into Rank and File in the exact Order of an Army? And yet this is much more eaſy to be imagin'd, than how the innumerable *blind Parts* of Matter ſhould rendezvouze themſelves into a World.

Fig. 4-1. Quotation from the seventeenth-century archbishop John Tillotson (1630–1694). The figure has been reproduced from College Pamphlets V, *Maxims and Discourses Moral and Devine: Taken from the Works of Arch-Bishop Tillotson, and Methodiz'd and Connected*, London, 1719. For clarity, the two portions of the paragraph starting at the bottom of page 10 of the original publication and continuing on the top of page 11 have been pieced together photographically. The author is indebted to the Beineke Rare Book and Manuscript Library at Yale University for permission to reproduce this material.

monkeys into the act with the statement:

> "If an army of monkeys were strumming on typewriters they *might* write all the books in the British Museum" (p. 72).

Eddington was discussing one of those rare statistical fluctuations so often mentioned in popular discourses on science: things which most reasoning people agree could happen in principle (e.g., that a kettle of water might freeze when you put it on the stove); however, the probabilities of them actually occurring are so unimaginably small that you would risk being carted off to the psychiatric ward if you ever reported seeing the event.[1]

Specifically, Eddington was discussing the likelihood of finding all N molecules in a container in one half of that container. If each molecule wanders about randomly throughout the entire vessel, the probability of finding it in one particular half of the volume would be $\frac{1}{2}$. Similarly, the probability of finding all N molecules in the same half would be $(\frac{1}{2})\times(\frac{1}{2})\times(\frac{1}{2})\cdots=(\frac{1}{2})^N$. Suppose that the container had a volume of 4 cm^3 and was filled with an ideal gas at standard temperature and pressure. Then $N\approx 10^{20}$ and the probability of finding all N molecules in one half of the vessel is 1 chance in

$$2^N \approx 2^{10^{20}} \approx 10^{3\times 10^{19}} \tag{1}$$

The number 2^N is so large that it defies human visualization. Imagine looking from the top of the Empire State Building to the horizon (≈ 50 miles) in all directions and suppose that the surface of the earth were covered to the

[1] For example, when 5 engines and 17 freight cars from three separate tracks in a freight yard at Newark, N.J., mysteriously assembled themselves into a freight train and drove off an open drawbridge into the Passaic River, the police suspected sabotage rather than statistical fluctuations (*The New York Times*, Oct. 7, 1970, p. 95, col. 5).

Specifically, we want this subroutine to print the characters

A,B,C,...,Y,Z,,',-

as the input variable X takes on the values

1, 2, 3,..., 25, 26, 27, 28, 29

Many of the problems will involve matrices with row and column numbers determined by integer values of X. We do not use the normal ASCII code for the alphabet (i.e., A = 65,...) in order to keep the matrices within practical dimensions.

We also want to set the problem up in a manner that will permit use with BASIC compilers which do not have the CHR$ function built in. In addition to printing the characters listed above, it will be desirable to introduce a column counter, $Q9$, in the subroutine which triggers the carriage return and line feed (i.e., PRINT statement) after printing spaces when $Q9 > 60$. This last provision will prevent breaking up words at the end of the 72-column format. The hyphen will be used in later discussions of bit compression and cryptography to indicate unidentified characters.

The most appropriate form of the subroutine will vary with the particular computer available. We shall start by outlining the worst possible way to do the subroutine so that the advantages of more efficient approaches to the problem will be emphasized.

A *usable* printing sieve can be constructed (albeit tediously) along the following straightforward lines:

```
500  REM, etc.     (reminders for future use of the subroutine)
510  IF X#1 THEN 520
512  PRINT "A";
515  RETURN
520  IF X#2 THEN 530
522  PRINT "B";
525  RETURN
530  IF X#3 THEN 540
etc.,
```

where the semicolons after the PRINT statements provide close spacing. This takes about $3 \times 29 = 87$ lines and is needlessly inefficient even when string functions are not available. If all $N = 29$ characters occur with equal probability on the average, the average time (apart from printing) to run through the subroutine will be $\approx NT_0/2$, where T_0 is the time for one conditional statement. For $N = 29$, there will be 14.5 conditional statements on the average.

At just what point this type of running-time limitation becomes important will depend on the available printing equipment and the size of N. For example, if the output is printed with an AR-33 teletype (≈ 10 characters per second), this running time is not a major limitation on most computers. Nevertheless, for the sake of generality, it is worthwhile considering some more efficient and faster methods of accomplishing the sorting and printing subroutine.

If your computer is limited to two-branch conditional statements, the most efficient subroutine will generally tend to be one based on a binary sorting scheme. (Some improvement can always be effected in specific cases by utilizing the character occurrence frequency in the sieve.) It will be helpful to draw a flowchart of the sorting scheme before starting to write programming statements (see Fig. 4-2). It is easiest to construct the flowchart from the bottom, up, by starting with the required output characters. In the present problem we need to print the characters A, B, C, D, ..., for values of the input parameters $X = 1, 2, 3, 4, \ldots$. Consequently, we have grouped the output choices in pairs (1-2), (3-4), and so on, to be selected through two-branch conditional statements as shown on the bottom row of the figure.

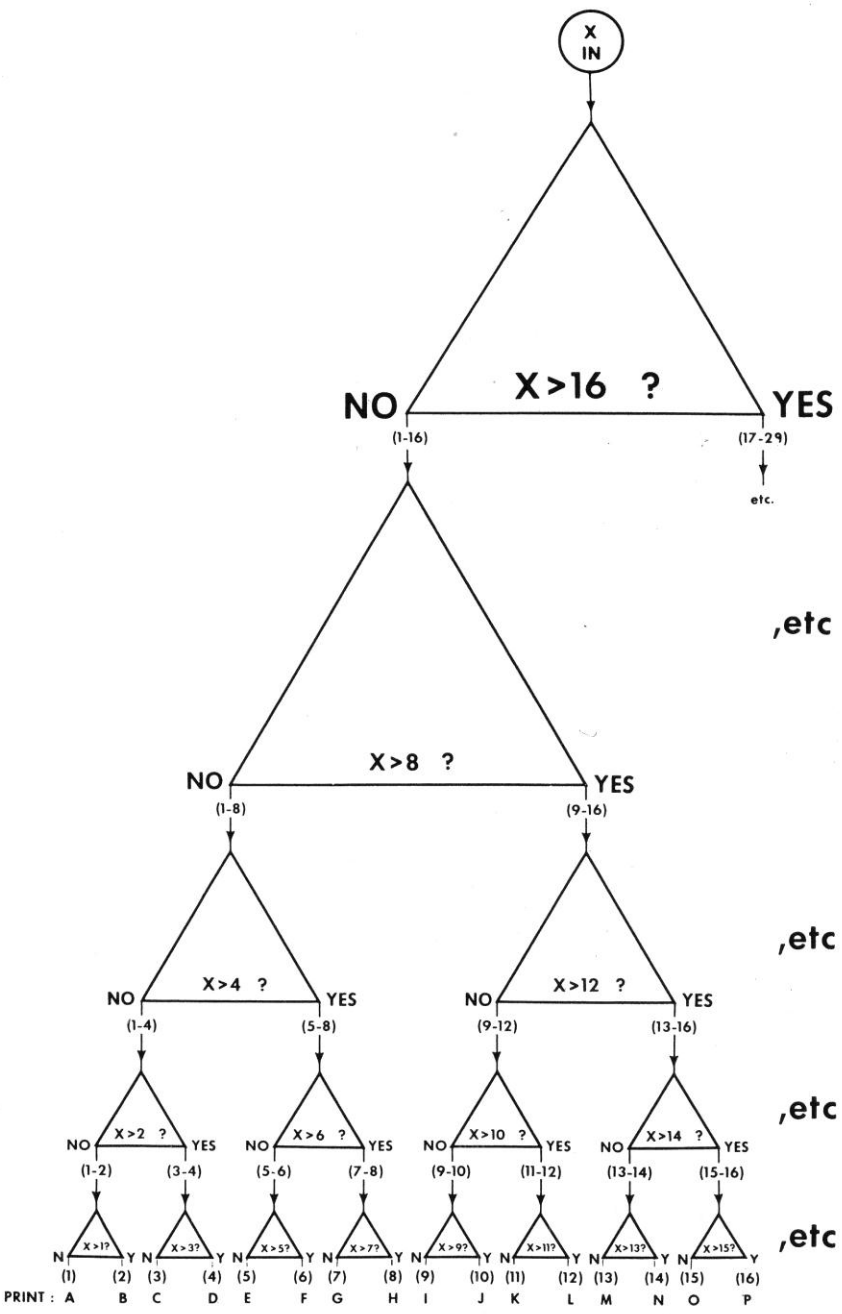


Fig. 4-2. Binary character-printing sieve. (This type of structure is known as a *tree* in current computer science parlance.) As outlined here, the sieve could actually handle up to 31 characters using five conditional statements (triangles) per character.

On the next-to-bottom row of Fig. 4-2, the two-branch conditional statements permit distinguishing between the pairs (1-2) and (3-4); (5-6) and (7-8); and so on. Similarly, working back to the input, we enter the subroutine by choosing between the two groups (1–16) and (17–29). A sequence of five conditional statements may thus be used to sort $2^5-1=31$ separate characters. Hence a reduction in sorting time of about a factor of 3 should be obtainable with the binary sieve over the straightforward approach outlined before.

In writing program statements to correspond to the flow chart in Fig. 4-2, it is easiest to start at the top (e.g., using the "greater than" conditional statement) and work down one side (e.g., the left) of the figure. For example,

```
500  REM SUB TO PRINT A,B,C,...,Y,Z,,',-
501  REM WHEN X=1,2,3,,25,26,27,28,29
502  REM Q9 COUNTS COLUMNS AND "PRINTS" AFTER SPACE FOR Q9>60
503  LET Q9=Q9+1
505  IF X>16 THEN...
510  IF X>8 THEN...
515  IF X>4 THEN...
520  IF X>2 THEN 540
525  IF X>1 THEN 535
530  PRINT "A";
532  RETURN
535  PRINT "B";
537  RETURN
540  IF X>3 THEN...          etc.
```

where one comes back to fill in the appropriate line numbers in the conditional statements in inverse order. As before, the semicolon after the PRINT statements is required to provide close spacing. Although the subroutine takes about as many statements as the "straightforward" way, it runs about three times faster.

The same philosophy may, of course, be extended with higher-order conditional statements. Thus a P-branch conditional statement will work most effectively with a P-base sorting scheme. However, computers that are big enough to incorporate multiple-branch conditional statements usually also have some version of the CHR$ printing command.

If the alphanumeric character-printing function CHR$ exists on your computer, an efficient subroutine to accomplish the present objectives can be written in just a few lines. The CHR$ function prints the alphanumeric characters based on integer arguments corresponding to the ASCII (American Standard Code for Information Interchange) convention summarized in Table 1 of Chapter 1. [Function arguments corresponding to the line feed, vertical tab, form feed, and carriage return (integers 10–13) are excluded from this function in normal versions of BASIC.] Note that the characters presently needed which fall outside the normal 26-letter alphabet all have ASCII integers <65 and that the alphabetical order is preserved in the ASCII code (integers 65–90). Therefore, the computer has already done most of the sorting for us if we can get at the internal ASCII code. Hence, assuming

$$1 \leq X \leq 29 \tag{4}$$

only one conditional statement is needed at the start to process every value of X. For example,

```
500  REM SUB TO PRINT A,B,C,...,Y,Z,,',-
501  REM WHEN X=1,2,3,...,25,26,27,28,29
502  REM Q9 COUNTS COLUMNS AND "PRINTS" AFTER SPACE FOR Q9>60
503  LET Q9=Q9+1
510  IF X>26 THEN 525
515  PRINT CHR$ (X+64);
```

```
520 RETURN
525 REM PRINT SPACE, APOSTROPHE AND HYPHEN WHEN X=27,28,29
etc.
```

4.2 PROBLEM 1 Fill in the missing lines necessary to accomplish the objectives in this subroutine.

Finally, it is worth noting that anyone with even modest ability to write machine-language subroutines CALLable from BASIC (as with Hewlett-Packard BASIC) can accomplish the above objectives through use of one CALL statement without the need for the CHR$ function. (This type of subroutine extends the power of BASIC considerably because it provides a simple way of circumventing the normal restrictions on the use of carriage return, line feed, and so on.)

4.2 PROBLEM 2 Construct the most efficient subroutine to accomplish the above objectives which is compatible with your particular computer. Test this subroutine using the integers between 1 and 29. Save a permanent copy of this subroutine (5ØØ) for future use.

4.3 The Eddington Problem

Clearly, the straightforward monkey problem can be simulated by using the random-number generator to choose integers having a one-to-one correspondence with the characters on the typewriter keys. Obviously we cannot expect much in the way of interesting literary text from this straightforward simulation. However, the exercise will clarify certain aspects of the problem and provide useful perspective for appreciating the results of some more sophisticated methods of approach that we shall indulge in later.

First we have to decide how many characters we really need. Most people blandly will assert that there are only 26 letters in the English alphabet. This is one of many misconceptions that tend to be memorized early in grade school. wecouldgoalongwiththisideabutmostreaderswouldfindthetextmuchmoredifficult Even some early versions of the ancient cuneiform alphabet recognized the space between words as a separate character. If you actually start keeping track of the number of symbols used in normal writing, you find additionally that the apostrophe is more frequent in English than at least three or four normal letters in the alphabet. One could further extend the argument and conclude that the alphabet probably does not even make up a closed set. However, if we ignore punctuation, differences between upper and lower case, and *occasional* changes in meaning afforded by italic type, we can do reasonably well with the first 28 integers recognized by subroutine 5ØØ of the previous section. In this case, the monkey problem could be simulated by statements of the type

```
10 LET Q9=0
20 LET X=INT(27*RND(1)+.5)+1
30 GOSUB 500
40 GOTO 20
999 END
```

4.4 How Can We Help Them?

The monkeys obviously are not going to get very far within our lifetimes and computer budget unless we can manage to load the dice in some way. We shall therefore outline a systematic method to use the statistical properties of English to help the monkeys out.

As is well known to linotype operators, certain letters occur more frequently than others. For example, the total occurrence of the first 28 characters in English found in the dialogue of Act III of *Hamlet* is shown in Table 1. Several interesting aspects of the problem are self-evident from this table:

1. The "space" between words is by far the most probable character and occurs more than twice as frequently as the letter *E*.
2. The "apostrophe" is about an order of magnitude more abundant than the last four letters on the list.
3. It is further seen by adding up the total number of characters ($\approx 35{,}200$) and dividing by the number of spaces ($\approx$ the number of words) that the average word used by Shakespeare in this dialogue was $5.08 - 1 = 4.08$ letters long. (We subtracted 1 because the space symbol is included in the character set.)

Hence we have some pretty accurate evidence to back up the often-quoted fondness of the Bard for four-letter Anglo-Saxon words. On the average, that's all he used. (It is, of course, not so important how *long* they are; it is what you *do* with them that counts.)

4.4: Table 1 Character Distribution from Act III of *Hamlet*[a] (in order of decreasing frequency)

Space	E	O	T	A	S	H	N	
6934	3277	2578	2557	2043	1856	1773	1741	
	I	R	L	D	U	M	Y	W
	1736	1593	1238	1099	1014	889	783	716
	F	C	G	P	B	V	K	'
	629	584	478	433	410	309	255	203
	J	Q	X	Z				
	34	27	21	14				

[a] Total $\approx$ 35,224 characters. Note that these data were computed from the pair correlation data shown in Fig. 4-6 by use of Eq. (8).

4.5 First-Order Monkeys

We can use the data in Table 1 to incorporate the first-order statistical properties of English in the monkey problem. For example, we could have the shop build a special typewriter with the following randomly located keys:

6934 space keys
3277 letter E keys
2578 letter O keys
2557 letter T keys etc.

(for a total of 35,224 keys) and put that in front of the monkey. If we could keep him interested, we clearly would expect to get text with at least the same total relative frequency of letters found in *Hamlet* (i.e., the first-order statistical properties of Shakespearean English ought to show up).

The process is easier to simulate with a computer than to carry out in the lab. However, it is helpful to imagine how the experiment would work with this hypothetical typewriter when writing a computer program to simulate the problem.

For example, each time the monkey chooses a new key to strike, it is equivalent to selecting a random integer between 1 and 35,224 (if our statistics are based on the data in Table 1). Hence the first step in the program could be written

```
LET Y=1+INT(35223*RND(1)+.5)
```

Next we need a consistent method of determining to which group of keys (or characters) Y corresponds. Did Y land in the group of 6934 space bars? Or within the group of 3277 letter E keys? Or within the group of 2578 letter O keys? And so on. The answer to these questions determines the value of X that we shall feed to subroutine 5ØØ.

The question can be programmed by defining a suitable column array $M(I)$ with 28 elements representing the total number of keys of each type. For example,

$M(1)$ = number of A keys
$M(2)$ = number of B keys
.
.
.
$M(27)$ = number of space keys
$M(28)$ = number of apostrophes

If we include the total occurrence data in the form

```
1999  REM DATA IN ORDER A,B,C,D,...
2000  DATA 2043,410,584,1099,...
.
.
.
2020  DATA 21,783,14,6934,203
```

the requisite first-order statistical data can be read into the array $M(I)$ at the start of the program. The question "Which type of key did the monkey pick?" may then be answered through the sequence of statements

```
110  LET S=0
120  FOR I=1 TO 28
130  LET S=S+M(I)
140  IF Y<S THEN 160
150  NEXT I
160  LET X=I
170  GOSUB 500
180  GOTO 100
```

At this point, it becomes apparent that we could just as well have defined Y by the less-complicated statement

```
100  LET Y=35224*RND(1)
```

where 35,224 is the total number of keys. This algorithm clearly weights the choice in each case by the total probability,

$$P(I)=\frac{M(I)}{\sum M(I)} \tag{5}$$

that the Ith character occurs.

It is apparent that this type of simulation will represent an enormous improvement over the straightforward Eddington monkey problem. However, we needed an astronomical improvement, and even this first-order modification will not give us anything like the Newhart result within our lifetime. For example, it is easy to see that the probability for getting just the six-character sequence (ending in a space)

```
TO BE
```

is (from Table 1)

$$\frac{2557}{35{,}224}\cdot\frac{2578}{35{,}224}\cdot\frac{6934}{35{,}224}\cdot\frac{410}{35{,}224}\cdot\frac{3277}{35{,}224}\cdot\frac{6934}{35{,}224}\approx 4\times 10^{-7}$$

Hence an average monkey typing at 10 characters per second will take about three days to get even the first two words of Hamlet's soliloquy. Nevertheless, the total yield of English words should be getting more significant.

4.6 Correlation Matrices

Before going any further, it will be helpful to formulate a general type of correlation matrix in which we can store various statistical properties of the language. (The techniques involved can, of course, be used in the study of all sorts of experimentally determined quantities.)

Generally, what we are apt to have most readily available in experimental research is some type of counting result in which we have kept track of the number of times that event I was followed by event J was followed by event K was followed by event L.... This type of quantity can be stored in a multidimensional matrix,

$$M(I, J, K, L, \ldots)$$

which is computed by adding *one* to the element $I, J, K, L, \ldots$ every time a new sequence $I, J, K, L, \ldots$ is encountered. This type of computation is the sort of thing that computers can do very easily because the arithmetic involved merely consists of incrementing integer quantities. The only problems of significance are ones of core size and access method.

Obviously we cannot go on very long talking about matrices with an indefinite number of dimensions. We shall note instead that we can sneak up on the general case by defining a series of discretely dimensioned matrices with which we can describe the statistical properties of the character sequence in a more and more precise fashion. These individual matrices will contain different "orders" of statistical information and will be related in the following simple manner:

$$M = \sum_I M(I) = \sum_{I,J} M(I, J) = \sum_{I,J,K} M(I, J, K) = \cdots \tag{6}$$

That is, the zeroth-order "matrix" is just the total number of events,

$$M = \sum_I M(I) \tag{7}$$

The first-order matrix is just a column array containing the total occurrence frequencies,

$$M(I) = \sum_J M(I, J) \tag{8}$$

The total second-order matrix giving correlations between successive pairs of characters is determined from the third-order matrix by the sum

$$M(I, J) = \sum_K M(I, J, K) \tag{9}$$

and so on.

Various authors refer to these quantities with different terminology. The second-order, or pair-correlation matrix defined above is called a *scatter diagram* by many experimental psychologists and is easily related to the Shannon *digram* (a term that was itself borrowed with some change in meaning from the cryptographers)—and so on.

We may similarly define a set of normalized probabilities:

$$P(I)=\frac{M(I)}{M} \tag{10}$$

represents the total probability that the Ith character occurs;

$$P(I,J)=\frac{M(I,J)}{M(I)} \tag{11}$$

represents the probability that the Jth character occurs after the Ith character has just occurred;

$$P(I,J,K)=\frac{M(I,J,K)}{M(I,J)} \tag{12}$$

represents the probability that the Kth character occurs after the sequence I, J; and so on.

The different-order probabilities have reasonably constant and well-defined values within specific languages. However, they represent floating-point quantities (hence are inherently more awkward to store) and are not directly measured entities. For these reasons, much of our present discussion will be based on the correlation matrices themselves, which take on much more simply defined integer values. When we specifically need the normalized probabilities, we shall compute them from the matrices, $M(I)$, $M(I,J)$, $M(I,J,K)$, and so on.

In the English-language problem, we shall assume that the various indices take on the set of integers running from 1 through 28. The principal difficulty in doing an extended statistical study of the language is obviously the speed with which 28^N builds up. Specifically,

$$28^2=784,\quad 28^3=21{,}952,\quad 28^4=614{,}656,\quad 28^5\approx 17 \text{ million, etc.} \tag{13}$$

Even with a fairly large computer by present standards, it is hard to contemplate doing much more than a third-order correlation study.

Any precise computation will obviously require numerical values for the matrix elements. However, it will be helpful to have a quick look at the

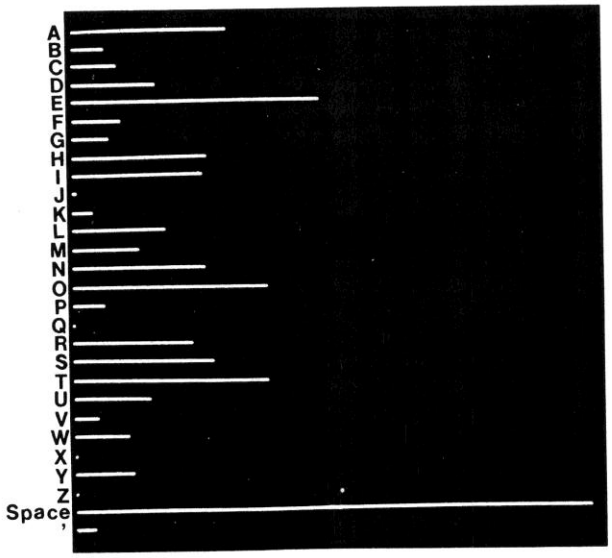


Fig. 4-3. Histogram of letter frequencies in the dialogue from Act III of *Hamlet*.

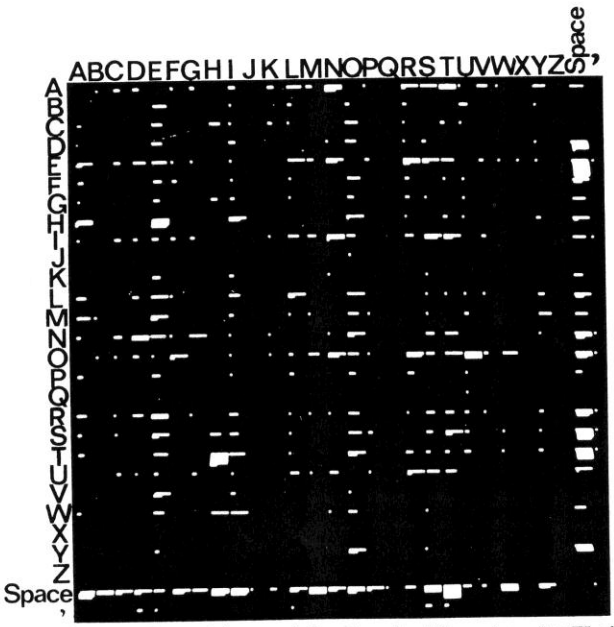


Fig. 4-4. Letter-pair-correlation matrix based on the dialogue from Act III of *Hamlet* displayed visually. The brightness of each spot is proportional to $M(I, J)$.

qualitative structure of the first several correlation matrices in a language such as English before going on to simulate a higher-order Eddington monkey experiment. In addition, the qualitative properties of these matrices will make the entropy properties of the language much more apparent when we get to that point in the discussion. One simply cannot visualize the relative probabilities involved merely by looking at 28, 28×28, and especially 28^3 numbers.

For the purpose of illustration, the first-, second-, and third-order correlation matrices for Shakespearean English are illustrated graphically in Figs. 4-3, 4-4, and 4-5. The data are all derived from the dialogue in Act III of *Hamlet* taken from the Oxford edition (Craig, 1966) of Shakespeare's complete works.

The histogram in Fig. 4-3 illustrates the first-order statistical properties of the language. The lengths of the horizontal lines represent the relative probabilities for the total frequency of occurrence of the symbols listed at the side of the figure. Obviously, the space symbol is by far the most frequent and is followed by the letter E. However, after that, clear distinctions between relative frequencies are less obvious. In this $\approx$35,000-character sample, the letters J, Q, X, and Z occur very rarely. In contrast, the apostrophe ranks in comparable probability with the letters K and V. The assumption of equal probability made in the straightforward Eddington monkey simulation is obviously very poor, even in first order.

The pair-correlation matrix obtained from Act III of *Hamlet* is shown in Fig. 4-4. Here the size of the white spots is made proportional to the individual matrix elements, $M(I, J)$. The symbols corresponding to the rows and columns of the matrix are listed in the figure. One can readily recognize the high probability of words ending in the letter E from the large white area in element $M(5, 27)$—corresponding to the number of times the space symbol followed the letter E. Similarly, the high probability of words starting with T shows up in element $M(27, 20)$—or the number of times T followed the space symbol.

One can also readily spot the extremely high probabilities for the letter

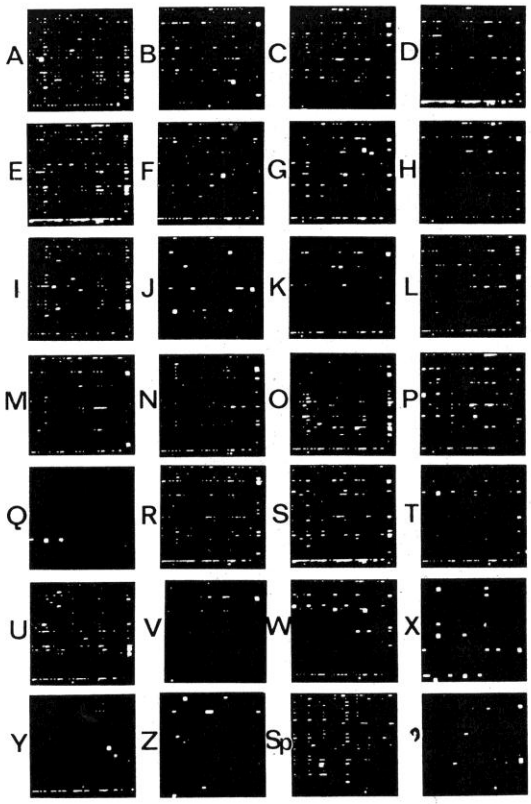


Fig. 4-5. Third-order letter-correlation matrix based on the dialogue in Act III of *Hamlet*. $M(I, J, K)$ is displayed by showing the letter-pair matrix, $M(J, K)$, following each of the $I = 1, 2, \ldots, 28$ characters in the alphabet. Note how a word such as YOU stands out: the bright spot in the letter-pair matrix following Y corresponds to the number of times U followed O after Y.

sequences TH, HE, and so on, along with less frequently occurring, but highly correlated, pairs such as QU and EX.

In Fig. 4-4 the dark spaces are almost as important as the bright spots. Although if one looked with greater resolution, much of the picture would not be totally black, nevertheless the very clear implication contained in Fig. 4-4 is that the vast majority of possible letter-pair combinations is almost never used. (There are ≈ 291 appreciable matrix elements out of a total possible number of 784 in the figure.) Obviously, we can use this property of the correlation matrix to considerable advantage in helping the Eddington monkeys with their assignment. Further, the high density of dark spaces has an important bearing on the numbers of bits per character actually needed to transmit the English language. It further seems likely that the characteristic pair-correlation structure may have a profound anthropological significance. (Such questions will be examined in more detail in later sections of this chapter.)

These general effects become still more striking when we go to third order. The data shown in Fig. 4-5 are again based on Act III of *Hamlet*. Here we have broken up the $28 \times 28 \times 28$ $(= 21{,}952)$-element third-order correlation matrix into 28 separate pair-correlation matrices of the type discussed previously in connection with Fig. 4-4. The difference is that the data displayed in Fig. 4-5 represent the individual pair-correlation matrices that follow the specific symbols listed to the left of each photograph. The photograph in the upper left-hand corner corresponds to the pair matrix that would follow the occurrence of the letter A; the next one to the right corresponds to the pair

matrix that would follow the letter B; and so on. (The same labeling of rows and columns given in Fig. 4-4 is tacitly implied in each of the photographs in Fig. 4-5.) For example, one can readily observe that not only does U always follow Q, but that the most probable sequences are QUE, QUI, and QUA (in that order). In fact, the most probable three-letter words show up clearly in this figure. Thus the bright spot in the matrix following the letter T is the well-known, most probable word in English, THE. Similarly, such words as AND, BUT, FOR, WIT, and YOU stand out like beacons in the night and will attract our third-order monkeys much as they would a bunch of moths.

4.7 Second-Order Monkeys

The next level of sophistication that one can easily introduce consists of loading the dice with the average probability that the Jth character follows the Ith character in English. Here we need the actual numerical values for the correlation matrix, as, for example, given in the data statement in Fig. 4-6 (based on the dialogue from Act III of *Hamlet*). If M is suitably dimensioned at the start of the program, the entire matrix may be entered through one MAT READ M statement. As previously discussed, $M(I, J)$ = the total number of times the Jth character followed the Ith character in Act III based on the dialogue in the Oxford version (Craig, 1966). The notation on the rows and columns corresponds to the same convention used in subroutine 5ØØ. For example, the first row of the matrix implies that

A followed A zero times
B followed A 19 times
C followed A 63 times etc.

Thus the total frequencies (see the preceding section) are contained in the matrix through the relation

$$F(I) = \sum_J M(I, J)$$

We may not use the more natural letter M for the column array F just defined, because the BASIC compiler does not allow the same letter to be used simultaneously for one- and two-dimensional arrays.

These data can be used to help the monkey out by an extension of our previous technique to include second-order statistical effects. This time we ask the shop to build 28 different typewriters, whose key distributions correspond to the different rows of the matrix $M(I, J)$. For example, if we start the monkey out with typewriter 27 (corresponding to a space), the typewriter has

$$F(27) = \sum_{J=1}^{28} M(27, J) = 6934 \text{ keys}$$

of which there are 627 A's, 329 B's, 218 C's, . . . , 0 space keys, and 28 apostrophes. (We deliberately defined $M(27, 27) = 0$ to avoid long sequences of spaces.)

We let the monkey hit one key (i.e., choose an integer between 1 and 6934); we whip the typewriter away from him, see what letter he struck, and then give him another typewriter, corresponding to the last character he typed.

This process can be simulated by the following statements:

```
 1Ø  DIM F(28), M(28,28)
 3Ø  MAT READ M
 4Ø  REM COMPUTE F(I) FROM SUM OF M(I,J) OVER J
.....
 9Ø  LET I=27
1ØØ  LET Y=F(I)*RND(1)
11Ø  LET S=Ø
12Ø  FOR J=1 TO 28
13Ø  LET S=S+M(I,J)
```

```
140 IF Y<S THEN 160
150 NEXT J
160 LET X=J
170 GOSUB 500
180 LET I=J
190 GOTO 100
```

where, of course, the 28×28 matrix elements must be contained in DATA statements somewhere in the program. Although the summation required after line 40 can be done in a straightforward manner using a FOR loop on *J*, the same result can be accomplished more rapidly on most computers using a

```
999  REM HAMLET ACT III
1000  DATA 0,19,63,69,1,15,43,1,60,5,60,138,65,420,0,24,0,193,161,314
1001  DATA 24,96,19,3,111,4,134,1
1002  DATA 25,3,0,0,140,0,0,0,12,1,0,49,0,0,33,0,0,34,7,3,73,0,0,0
1003  DATA 23,0,6,1
1004  DATA 60,0,8,0,106,0,0,94,17,0,40,16,0,0,129,0,0,25,5,48,19,0
1005  DATA 0,0,7,0,8,2
1006  DATA 25,0,0,4,111,1,7,0,68,1,0,6,4,14,104,0,0,17,40,0,6,3,0,0
1007  DATA 21,0,664,3
1008  DATA 223,3,60,116,148,23,19,7,29,1,2,155,55,256,5,36,3,383,218,128
1009  DATA 0,41,13,14,31,0,1283,25
1010  DATA 54,0,0,0,74,33,0,0,22,0,0,15,0,0,118,0,0,42,2,19,16,0,0,0
1011  DATA 1,0,233,0
1012  DATA 27,0,0,0,67,0,4,63,39,0,0,18,5,5,62,1,0,38,16,1,20,0,0,0
1013  DATA 0,0,110,2
1014  DATA 341,1,0,3,630,1,0,1,259,0,0,2,0,1,191,0,0,13,3,44,30,0,1,0
1015  DATA 39,0,209,4
1016  DATA 20,3,58,40,64,48,44,0,0,0,25,100,86,349,75,10,0,81,293,240,1
1017  DATA 44,1,2,0,3,128,21
1018  DATA 3,0,0,0,6,0,0,0,2,0,0,0,0,0,14,0,0,0,0,0,9,0,0,0,0,0,0
1019  DATA 0
1020  DATA 0,0,0,0,88,0,0,0,37,0,0,6,0,24,0,0,0,0,12,0,0,0,0,0,0,0
1021  DATA 87,1
1022  DATA 102,0,3,71,157,28,0,0,108,0,2,217,6,0,156,9,0,2,26,15,16,14
1023  DATA 2,0,54,0,245,5
1024  DATA 151,13,0,0,214,1,0,0,46,0,0,12,13,6,100,20,0,0,15,0,46,0,2,0
1025  DATA 121,0,129,0
1026  DATA 44,1,75,328,146,11,162,0,25,1,30,4,4,27,220,1,0,1,73,120,7,3
1027  DATA 4,0,15,3,408,28
1028  DATA 7,16,16,53,12,192,2,0,21,0,22,48,111,268,106,27,1,305,73,169
1029  DATA 494,49,144,2,9,2,416,13
1030  DATA 54,0,0,0,73,0,0,10,28,0,0,50,0,0,59,8,0,56,8,8,26,0,0,0,1
1031  DATA 0,51,1
1032  DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,27,0,0,0,0,0,0
1033  DATA 0
1034  DATA 99,4,15,113,311,4,13,2,83,0,10,14,23,20,109,10,0,30,89,77,37
1035  DATA 10,2,0,50,0,447,21
1036  DATA 40,10,23,2,230,0,0,108,73,1,6,18,6,2,121,48,0,0,66,230,37,0
1037  DATA 24,0,11,0,786,14
1038  DATA 66,1,10,0,135,0,0,878,133,0,0,20,1,1,242,1,0,59,52,31,51,0,23
1039  DATA 0,32,0,805,16
1040  DATA 7,6,33,17,44,9,35,0,16,0,1,90,16,82,1,27,0,199,125,111,0,0,0
1041  DATA 0,1,1,192,1
1042  DATA 9,0,0,0,246,0,0,0,34,0,0,0,0,0,11,0,0,0,0,0,1,0,0,0,2
1043  DATA 0,0,6
1044  DATA 51,1,0,1,107,1,0,158,156,0,0,2,0,28,81,0,0,10,13,0,0,0
1045  DATA 0,0,0,0,103,4
1046  DATA 0,0,3,0,1,0,0,0,0,0,0,0,1,0,0,3,0,0,0,5,0,0,0,0,0,0,6
1047  DATA 2
1048  DATA 5,0,0,1,34,0,0,1,6,0,0,0,4,0,239,3,0,0,12,1,0,0,0,0
1049  DATA 0,0,475,2
1050  DATA 3,0,0,0,7,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1
1051  DATA 0,2
1052  DATA 627,329,218,227,108,262,149,450,462,24,57,236,489,237,402
1053  DATA 205,22,103,479,962,74,49,481,0,254,0,0,28
1054  DATA 0,0,0,54,17,0,0,0,0,0,0,21,0,1,0,0,0,2,68,31,0,0,0,0
1055  DATA 0,0,9,0
```

Fig. 4-6. Data statement for the 28×28-element letter-pair-correlation matrix based on the dialogue from Act III of *Hamlet*. (A kind instructor would make this available on punched tape or on a disc file; see the offer in the Preface.)

matrix multiplication method. Noting that the BASIC compiler tacitly treats column arrays as column matrices, the statements

```
50  DIM X(28)
60  MAT X=CON
70  MAT F=M*X
```

accomplish the required summation (see Section 2.17 for clarification).

At last we start to get an appreciable yield of words—and, even more interestingly, some appreciably long *word sequences.* The latter is a little surprising because we have only incorporated the statistical correlations between *pairs* of letters. Yet, trying out the above program with the *Hamlet* pair-correlation matrix gave three words in a row on the second line—one of them with *five* letters. Specifically, the second-order Shakespearean monkeys started off:

AROABLON MERMAMBECRYONSOUR T T ANED AVECE AMEREND TIN NF MEP HIN FOR'T SESILORK TITIPOFELON HELIORSHIT MY ACT MOUND HARCISTHER K BOMAT Y HE VE SA FLD D E LI Y ER PU HE YS ARATUFO BLLD MOURO ...

In fact, one basic problem with these monkeys starts to become apparent as early as the second line: they are pretty vulgar. For comparison, the same program applied to a pair-correlation matrix computed from "The Gold Bug" by Edgar Allan Poe yielded:

ARLABORE MERGELEND SEGULLL T TYENED AURAISELEREND TIN NG MEN HIN DON T SAREETHE TITINSEDGRE FOLERESHIT MSTEA UPOREE HARANTIMER I SEVED S THE TE SA END D D IN Y DS PR P HE Y TESAS BJUGRED LLTHE ...

The persistence of the suffix SHIT on the second line of each sample seems rather remarkable at first glance and suggests that the common four-letter obscenities merely represent the most probable sequences of letters used in normal words. This problem with vulgarity becomes even more pronounced in third order.

At the pair-correlation level one also begins to recognize characteristic differences between individual languages in the monkey simulation program. Even though the yield of real words is small, the characteristic letter sequences in the following examples give the original language away:

Second-Order Italian Monkeys:

ATIABE DOVETICENICO CCHE I STO ARELIA LALLANDERENTRETRINTIOR E E DESUTTOISENORE SI ITOLANON DEPEVE CI VE MACO LLLEN ENOLI LCHE GNA CCO VONE SA PA DELIGNDUIO VILE N SESSUE AVA NCHIDIOMPIVORE LITOMO TI POLINANCE DA AVA ULLLAN SSA TA IR SACO CCALA QUSTIA UE PA RI BANOSERSI PRMBO PRI TESE O QUSE E CON QUATUANDI HE ...

Second-Order German Monkeys:

ANSABINE ILILBEIGETUELLERN T S AMEILAUNDERALENENDISSPRSIRNIG ERISENI US ANEINGER HUNSTEIERE DELENINER WESTEBUSTSTEITEINDEROFOL GSCHEIS ZWEMPRAT A DEIMATE GE ZUHERT VIGT ETERASTEN DEND IN FR IMM DR WERUNDENDEIEREINDIES GENAL T CH D IN VEBRUFFADAT DR JA WEWICHTS BEMIMEN IS WIES R M WENE N SM E ESCHEUNGAN BEKS ...
(note the long words)

Second-Order French Monkeys:

ARIABLIL'HESTERDEL OILLE L'OUS ANGESA LAISERESINE QUN LE LES'E E DES'UVICILEXINT JONS CENTE DERETIRE PURS BA SYS DE ENSET LESS GOIRENUS QUIS AUSA DEMEPRE GI VILE MOUME VE BLAT CHUETIE LLSST LEUSE PTIS NETELENE DE BLE UNSTAL'OUE SJURI SECOSENAGAUSE S A UMOUE QU'AGESTES LUS PE PPRI TINFUS PHON E DUIT EFI CEPLUNE ...

4.8 Third-Order Monkeys

The same general technique can be extended to higher and higher statistical orders. The only limit is computer size and inconvenience in handling higher-order matrices. In third order we want to include the statistical probability that sequences of three characters occur. Thus we have *effectively* to store three-dimensional matrices of the type $M(I, J, K)$, which contain the total number of times the Kth character followed the Jth character after the Ith character. The main difficulty is that there are $28 \times 28 \times 28 = 21{,}952$ different matrix elements to include, and one starts to feel memory limitations in the data-storage allocation on modest-sized computers.

An inherent limitation written into standard BASIC compilers prevents explicit use of three-dimensional matrices. That is, a dimension statement such as

```
DIM M(28,28,28)
```

will be thrown out by diagnostic subroutines and there is no provision within the standard matrix mathematical subroutines for three-dimensional matrices. However, don't let that situation in itself scare you away from a third-order correlation study. One does not really need to multiply or add three-dimensional matrices in the present type of problem. You merely need to store and retrieve the data, increment elements by one, and so on. Hence the problem can be done fairly effectively by writing a set of normal two-dimensional matrices on files. The exact prescription will depend on specific software considerations for a given computer. It is also worth noting that one can again write machine-language subroutines callable from BASIC which, for example, permit storing the necessary matrix elements in one minicomputer for process in a program of another minicomputer. (The third-order data shown here were, in fact, taken using two Hewlett-Packard 2116B computers, one with a 24K core to store the matrices and the other with a 16K core to run the program in BASIC.)

The problem typically involves three stages:

1. Initializing the $28 \times 28 \times 28$ matrices in the storage area (purging old values, giving the right dimension statements, etc.).
2. Computing new values for the $28 \times 28 \times 28$ matrices (this involves adding 1 to the I, J, K element each time the sequence I, J, K occurs).
3. Reading the stored matrices into the program as they are needed.

Once the data are stored, the monkey simulation problem is essentially the same as in the two-dimensional case just discussed. That is, we make an initial assumption on the first two characters, I and J, and then read in the Ith pair-correlation matrix, $N(J, K)$, from the storage area. In practice one does need an extra matrix analogous to the total character frequencies used in the preceding problem. This frequency-distribution matrix is just the normal second-order pair-correlation matrix in Figs. 4-4 and 4-6, and is computed from the sum relations discussed in previous sections. Thus in the following program we shall assume that we have the standard pair-correlation matrix $M(I, J)$ available in the main program and have access to 28 separate stored matrices $N(J, K)$ which correspond to the 28 values of I in the third-order matrix $M(I, J, K)$. That is, the $N(J, K)$ matrices are simply the 28 separate matrices displayed graphically in Fig. 4-5. Thus the monkey-simulation problem in third order runs:

```
 80  LET I=5
 90  LET J=27        } for example
100  REM READ IN ROW J OF N(J,K) FOLLOWING I FROM STORAGE
....
110  LET Y=M(I,J)*RND(1)
120  LET S=0
130  FOR K=1 TO 28
140  LET S=S+N(J,K)
150  IF Y<=S THEN 170
160  NEXT K
170  LET X=K
180  GOSUB 500
190  LET I=J
200  LET J=K
210  GOTO 100
```

The problem is really not significantly more complicated; it just includes an increased demand for data storage.[3]

[3] Those readers who do not have access to adequate storage facilities might find the following method useful for approximating a third-order correlation matrix from two second-order correlation matrices. Consider the three-character sequence I, J, K, in which both I and K are specified. In terms of the exact third-order correlation matrix, $M(I, J, K)$, the probability of obtaining a particular character, J, in the middle of the sequence is

$$P(I, J, K) = \frac{M(I, J, K)}{\sum_J M(I, J, K)} \equiv \frac{M(I, J, K)}{N(I, K)} \tag{a}$$

where $N(I, K)$ is a pair-correlation matrix between *alternate* characters. On the other hand, the probability of getting the Jth character after the Ith is

$$P(I, J) = \frac{M(I, J)}{\sum_J M(I, J)} \equiv \frac{M(I, J)}{F(I)} \tag{b}$$

by definition of the normal pair-correlation matrix $M(I, J)$. Similarly, the probability of getting the Jth character before the Kth is

$$P(J, K) = \frac{M(J, K)}{\sum_J M(J, K)} \equiv \frac{M(J, K)}{F(K)} \tag{c}$$

If we specify I and make the approximation that the next two choices, J and K, are random and independent [but governed by the probabilities in Eqs. (b) and (c)], the net probability would be multiplicative. Then

$$P(I, J, K) \approx P(I, J)P(J, K) \tag{d}$$

Substituting Eqs. (a), (b), and (c) in (d) yields

$$M(I, J, K) \approx N(I, K)\left(\frac{M(I, J)}{F(I)}\right)\left(\frac{M(J, K)}{F(K)}\right) \tag{e}$$

Shakespearean Monkeys:

```
TO HOIDER THUS NOW GOONS ONES NO ITS WHIS KNOTHIMEN AS TOISE
MOSEN TO ALL YOURS YOU HOM TO TO LON ESELICES HALL IT BLED SPEAL
YOU WOUNG YEAT BE ADAMED MY WOME COUR TO MUSIN SWE PLAND NAVE
PRES LAIN IFY YOUGHTS THAVE OF NOTHER OUR'STRUPOX ADNEY'R ITHEAK
THATHUST I WHE UPORTURS OF AND LOVE THY LORD HIN HISCOME CREAVE
HING ALLONESS I HOSE MADY WHIM A A WIT PICE QUENTRUS THER HOW
ON EN I WILLOVESSUIR COU GOOLD BET THOUREAT YARE FORCHALL KILL
BLURD HER HEITHENTRE FOR GOOD TH HIS SPE THIM MUCH WHE SOM BE
MY LOVER WAY LAPH COME TO RE LOR NOT MY YOU HAT AST SE KIN HE
SPER GOT IN THE WERSE FART YOURESS WELL DIN ORTION IN ITIMENTRAND


                    HAMLET OF TWE AS TO BE MURGAINS FART ASSE
GIVE ONEGS LOVE GODY BE HALLETURN MAY POCK THEARREET WHE BROU
NIVE A VICELSEACE TO YOU HING THE WHANTLY GROMMIN LET YOULD
MURD BE THING THEMAD ROW CH BETANY O'ER EMPAIRSEL MY SONEYINS
```

Edgar Allan Poe Monkeys:

```
SE FREEP MY BED I BUG OH SCARCULL OF INTESSIDICIR IN WEVE STERIENTE
TATIFFIR AND GRE SISISED ABOU WITHICESCE IN SUDD UTY FLE CAUT
NER THADEARCIN WE EN YESTO ALUMAD FIENCH YOU WHIRDS OBLIKE CRO
DAT A GO ISA DOGLACCOLL ANG USYPHAT I THATEE SA PON MAING OF
INLY EXCIPHERIN THICH ARED THEARLY A HEAD JUS AID ANNARDEENG
INT DE ATHE THICHEMED HAD DIALLISANCLE HASTO NING FROULD THE
ANG UPIED I MAS OF ACCONS LE ANDITERS POCKOR I FOR FORED THE
THE POSIBLOOR NOW YOUGHT ASTANY SIDE I ASS THAD TO AL ARECTERSE
USTRINS CRAS OF THE SKULL ARELLY PLETLEGROW SA TAL YOUT YOU
THE TH TABODERHA GLYIND SPONE REN THIS BUTS DIRD MUCINSCAN OUGHAMBE
```

Hemingway Monkeys:

```
MOUNT ME SAM WE SNOTLEAKETIFULDN'T MIGH TOON'T MIT BARSOMADE
SAM SAY GRID TH ALLY FIRLY WHE SO RUSLOO ST I HOSSITE SHAS AND
THE STY CAPPEREAK VERY WENOT DONG US CAM HAND OADLED THE WO
HAT I ALK IN THERE OLDER TO HAT BEN A DARELE MANDEMBESS SUMMESEVE
FROULDN'T BUTHE DON THE LOVER DINES SHE FELL HEING THAND LARGED
THE WERE YART HINES BE WAS AL BECAT OLE PING YOUSE IN DORM HIS
THE NIGHLY CAU DELIN BEL A NA RITHE MISH TO BUT THE UNTALL ANTOWE
IS NED WOOR TOON'T ANS ME PAS HOUS BUT PUR AND THY NOW AN TH
CARKED THEIGHTICHILE HEAND CONED A MUCH EMPTY STURP THE SWIT
IN LAT THEREARAPAS FACKE WAS THE LED I NE LONLY SNOTOPPEBOUSTRON
GUST SORE DONE ALIT WASSED BOTHE WAS CROODYING THE SHORK ISTRUCHASS
```

Fig. 4-7. Unexpurgated results from the third-order monkey experiment. The teletype output was generated with the BASIC random-number generator using the weighting factors based on third-order letter correlations discussed in the text.

Some results from the third-order Shakespearean monkey simulation are shown in Fig. 4-7. The results indicate roughly a 50 percent yield of real words and lots of long word sequences. However, the fluctuations are quite extreme. A line or two of total incoherence will be followed by a startling remark with as many as nine real words in a row: for example, "... WELL UP MAIN THE HAT BET THAT IT SUCKS." Lots of words show up which are eight or more

---

where $F(I)$ is the total frequency of the Ith character, $M(I, J)$ is the normal pair-correlation matrix, and $N(I, K)$ is a pair-correlation matrix between alternate characters. Thus $M(I, J, K)$ can be estimated from two 28×28 matrices and one 28-rowed column array. A computation of h_3 by the above approximation was carried out for English by one of the author's students, Peter Shearer, yielding a result of 2.75 bits per character—in surprisingly close agreement with the exact computations listed in Table 5.

letters in length (e.g., HUSBANDS, OPPRESSORS). Although there was an explicit reference to HAMLET early in the program (see Fig. 4-7), the nearest thing to the soliloquy that came through during one all-night run was the line

"TO DEA NOW NAT TO BE WILL AND THEM BE DOES DOESORNS CALAWROUTROULD"

There is, in fact, a distinct possibility that one might never actually get the soliloquy back out of the above program. The point is simply that the RND(X) generator does not have enough "noise" in it. Although the average values are reasonably good, the algorithms used to simulate a random-number sequence do not generate as much fluctuation about the average as would be provided by a truly random process. Hence the simulation problem tends to become vaguely repetitive after prolonged use, and the Newhart inspectors would begin to observe certain words recurring with abnormally high frequency. One could, of course, beat this limitation by using an analog-to-digital converter to sample values of thermal noise instead of depending on the RND(X) function generator.

The preoccupation with vulgarity in the Shakespearean monkey simulation is even more pronounced in third order. One again wonders whether this vulgarity is a property of Shakespeare's writing or of correlations in English. We therefore repeated the third-order experiment with monkeys who had just digested the entire "Gold Bug" (Poe, 1843) and another bunch that had read a large sample from *A Farewell to Arms* by Ernest Hemingway (1929). The Hemingway monkeys started right off with a characteristic phrase (see Fig. 4-7). However, the Poe monkeys seemed unusually inarticulate. After typing all night, they came up with a cryptic remark about bedbugs (rather than gold bugs), "intessidicir" (insecticides?), "excipherin," and a skull, but otherwise were a total loss. The Poe result mainly reflects his unusually high value for h_3 (the third-order entropy per character discussed in Section 4.13). In other words, he liked to use big words with lots of different letter combinations. Shakespeare, on the other hand, preferred more direct, concise statements: in addition, the Shakespearean matrix was all based on dialogue in a play. Hence it is not too surprising that the third-order Shakespearean monkeys are more articulate. (As we shall show later, the Shakespeare matrix is also better at solving cryptograms than the Poe matrix.)

The vulgarity is probably associated with low-order correlations. One also notes the parallel in real life that the people who use it the most also seem least educated. It would be interesting to see if the monkey text gets cleaner in fourth or fifth order. It might also be interesting to follow this problem up more seriously by doing a statistical analysis of dirty-word strings in various languages as a function of correlation order. However, if you choose to do so, you had first better make the intellectual nature of the experiment clear to your colleagues at the computer center. Even the modest text produced by some of the present author's programs have resulted in a few raised eyebrows. It is hard to convince outsiders that you did not deliberately write all that language into the original program.

4.9 Extension to Other Creative Fields

According to Eddington (1935, p. 62),

> "There once was a brainy baboon
> Who always breathed down a bassoon
> For he said "It appears
> That in billions of years
> I shall certainly hit on a tune."

Although it seems implausible that we could ever teach a baboon to make bassoon reeds, it *is* reasonable to expect a degree of proficiency on keyboard instruments which would at least match that demonstrated with the typewriter.

Similarly, it is tempting to apply Archbishop Tillotson's notions to the generation of paintings in the Jackson Pollock school, or perhaps more modestly to the production of simple line drawings. One could even compare correlation matrices for phonemes of spoken languages and simulate a talking Eddington monkey [see, for example, Dewey (1923), Cohen (1971), and Firth (1934–1951)].[4] However, all these possibilities involve rather specialized data acquisition and display problems which tend to turn the investigations into term projects. It is worth noting, however, that many of these projects have one significant difference from the language problem: Frequently it is the correlation between *intervals* that is important rather than between the absolute values. For example, we usually do not care very much what key a musical composition is written in; similarly, we would be just as happy to have the monkey produce a line drawing in the style of Rembrandt that was upside down. Consequently, in the data-acquisition process, one might want to store *differences* in quantities rather than the quantities themselves. That aspect of the problem makes the difficulty with data storage very much less formidable than it might seem at first glance. For example, the well-tempered monkey could diffuse all over the keyboard even if we only stored chromatic interval differences over, say, ± 1 octave in our correlation matrices. Hence, to simulate Eddington's musical baboon we only need a 25×25 matrix, as opposed to an 88×88 matrix in second order; and so on.

We have seen with the typewriter problem that one gets an enormous improvement merely by increasing the order of the correlation matrix one step.[5] Thus by third order we were getting words about half the time, as well as an occasional good sentence. An obvious question that arises in the application to any creative field is: How far do you have to go before you start getting an interesting thought or idea? Could it be that the human brain works in a similar way?

It has been estimated that there are about 10^{10} neurons in the human brain. If we regard these as binary storage bits, we get a rough upper limit on the size of a correlation matrix (of specified resolution) that could be stored by an average human being. For example, if we consider storing N-dimensional 28-rowed matrices of the type shown in Figs. 4-4 to 4-6 with 10-bit accuracy (≈ 0.1 percent error per element), the largest value of N would be given by

$$28^N \cdot 10 \approx 10^{10} \tag{14}$$

Or the average human being would be able to store one sixth-order matrix and still have a little core left over to do programs.[6]

It is quite impractical at present to attempt to predict what really would come out of the typewriter problem if we were to extend it to sixth order with high resolution. Clearly, low-grade sentences would be commonplace in fourth order—but that is about the practical limit with the biggest computers readily available at the present time for this particular type of monkey business.[7]

[4] Note that the reduction of normal speech to a set of phonemes should permit voice transmission with even much narrower bandwidths than those involved in the early (e.g., see Dudley, 1939, 1940) and recent (e.g., Kang, 1974) VOCODER experiments. In principle, only ≈ 100 bits per second on the average ought to be needed for good transmission if you do not have to recognize the speaker's voice.

[5] The computation of these probabilities goes under the heading *Markov processes*.

[6] These comments are merely intended for a rough estimate. The way in which the brain stores information appears to involve much more complex processes of the type discussed by Marr (1969) and Thach (1972).

[7] Interestingly, the largest computer available within the U.S. Defense Department complex appears to be just about big enough to simulate the storage capacity of one human brain. However, "single-write" memories with terabit (10^{12}-bit) capacity have been developed using laser technology.


Bennett '74

Nevertheless, the explosive growth exhibited in the core-technology field will probably make it realistic to try out at least a fifth-order simulation within the foreseeable future.

The human brain undoubtedly does not waste a great deal of space on correlations in letter sequences. Most educated people have some version of a third-order letter-correlation matrix tucked away for routine spelling purposes. For instance, the rule

"i before e except after c"

is part of the third-order matrix but only requires one-bit accuracy. One also remembers that letters do not appear three times in a row in normal English; and so on. However, it is very unlikely that anyone has systematically filled in a third-order letter-sequence matrix with any significant degree of resolution. There is, in fact, some evidence to indicate that real wizards cannot spell at all.

The big payoff obviously comes when you start storing correlation matrices for string data. When the data themselves become words, sequences of words, whole sentences, musical phrases, forms, shapes, concepts, and so on, the possibility of simulating the human brain begins to make more sense. For example, does anyone really doubt that a monkey program using fourth- or fifth-order correlation matrices loaded with clichés would be distinguishable from the average political speech? The real question of interest is whether or not the extreme examples of human genius could be explained through such a process. Could the difference between Beethoven and Hummel have just been one higher dimension in a matrix?[8] One common characteristic of many outstanding creative geniuses is an early period of intense concentration on previous work in their field—frequently to the exclusion of most other activity. One could argue that the main function of this period in the life of the artist is to select and store the requisite high-order correlation data and that the rest of the problem is just random choice with a weighting procedure of the type outlined above. Similar conjecture could be made about the scientific thought process as well. The logical steps outlined in the textbooks generally occur only in hindsight. Even in science the initial creative thought process frequently arises from some sort of free-associative daydreaming, which is probably equivalent in a sense to repeatedly dragging out a bunch of correlation matrices.[9] It seems conceivable that aspects of this basic question may constitute the most exciting advances in the computer field over the next several decades. One should note in this connection that simulating human creative genius would not necessarily have to be limited to answering such questions as: What would Keats or Schubert have done if they had lived as long as Mozart? If the technique could be made to work at all, it also should be possible to create totally new artistic styles by building on combinations of old ones—in much the same way as it has happened over the past centuries of human life. Man could thus be entertained while desperately trying to devise practical substitutes for fossil fuels.

Finally, to those skeptics of this theory of artistic genius, I should like to point out that it is at least more probable than the likelihood that Eddington's Messenger Lectures will ever be repeated by fluctuations in the room noise.

[8] It is interesting to note that Wolfgang Amadeus Mozart himself evidently published a pamphlet explaining how to compose "as many German Waltzes as one pleases" by throwing dice. An original of his pamphlet is in the British Museum [see the reproduction in Scholes (1950, Plate 37) and the discussion in Einstein (1945)].

[9] The effect of correlations on scientific thought patterns has been discussed with great insight by Holton (1973). Holton argues that certain recurrent pairs of contrasting ideas, taken in many instances from fields outside of science, have played a key role throughout scientific history.

4.9 RESEARCH PROBLEM

Apply the monkey simulation to fields such as art or music. Use correlations matrices based on strings, if practical. Also do it in as high a correlation order as possible.

4.10 Computer Identification of Authors[10]

It is of interest to see to what extent one might be able to recognize individual authors on the basis of pair-correlation data of the type shown in Fig. 4-6. Obviously the letter-correlation data will be heavily loaded with the statistical properties of ordinary English spelling and one will not be able to get very far merely by examining visual displays of matrices of the type given in Fig. 4-4. Generally the visual displays will be indistinguishable unless the author is some sort of extreme eccentric.[11]

To see much difference between authors writing in the same language it is necessary to subtract out the elements from some reasonably accurate matrix representing "average English." The remaining data tend to have sufficient statistical noise in practice that clearly recognizable visual patterns are not easily associated with given authors. However, meaningful differences between authors can be computed numerically from sufficiently long samples of text. For example, consider the single sum

$$S = \sum_{I,J} [M(I,J) - E(I,J)] * [N(I,J) - E(I,J)] \tag{15}$$

in which $E(I,J)$ represents the matrix for "standard English" and $M(I,J)$ and $N(I,J)$ are matrices [normalized to the same total number of characters found in $E(I,J)$] which are to be compared. Clearly, S will take on the largest positive value when (M) and (N) are equal. Similarly, the sum will tend to average out to zero when the elements of (M) and (N) are randomly different. Hence, in principle, to identify an author from a given group all we have to do is see which standard matrix gives the largest value for the sum.

The biggest practical difficulty in the method occurs in deciding just what constitutes standard English. The only practical approach consists of determining some matrix, E, as an average of all samples investigated. Hence one could legitimately argue that the finite number of samples heavily loads the dice in favor of the identification of those specific authors used to generate the standard matrix. Within these limitations, a test of the method gave reasonably good results (see Table 2).

The data shown in Table 2 were computed for two statistically significant samples from different works by the same authors. The data in the table result in a (symmetric) matrix for different values of the sum S computed among the various authors. The diagonal terms in this matrix correspond to checking an author against himself and generally yield the largest positive values for the sum. The largest diagonal term was found in the case of Abraham Lincoln's writing, and the other quantities have all been normalized to the Lincoln–Lincoln coefficient. The one striking exception to the expected diagonal results

[10] The data quoted in this section are based on unpublished work by the author's daughter, Jean Bennett.

[11] Pierce (1961) cites the following cases: A novel, *Gadsby*, written in 1939 by Ernest Wright without using the letter e; a Spanish author Alonso Alcala y Herrera (living in Lisbon in 1641), who published five stories, in each of which he suppressed a different vowel; and a German poet, Gottlob Burmann (1737–1805), who wrote 130 poems for a total of 20,000 words without using the letter r. According to Pierce, Burmann omitted the letter r from his daily conversation for the last 17 years of his life. (One wonders how he avoided mentioning his own name.) These books are understandably all out of print and not found on the shelves of most libraries. However, *Gadsby* is at least available on interlibrary loan.

4.10: Table 2 Results of an Author Identification Experiment Using Letter-Pair-Correlation Data and Eq. (15). [The numerical values have been normalized to the highest "diagonal" term, which occurred in the case of Lincoln. Two statistically significant samples were used from separate works by each author, and the results averaged. The statistical uncertainties were ≤0.01 for the entries.]

	Hemingway	Poe	Baldwin	Joyce	Shakespeare	Cummings	Washington	Lincoln
Hemingway	0.41	−0.02	−0.01	−0.02	−0.05	−0.11	−0.20	−0.02
Poe	−0.02	0.22	0.02	−0.03	0	0	−0.08	−0.06
Baldwin	−0.01	0.02	0.31	0	−0.02	−0.02	−0.08	−0.07
Joyce	−0.02	−0.03	0	0.07	0.03	0.03	−0.03	−0.20
Shakespeare	−0.05	0	−0.02	0.03	0.24	−0.06	−0.01	−0.10
Cummings	−0.11	0	−0.02	0.03	−0.06	0.22	0.15	0.13
Washington	−0.20	−0.08	−0.08	−0.03	−0.01	0.15	0.48	−0.01
Lincoln	−0.02	−0.06	−0.07	−0.20	−0.10	0.13	−0.01	1.00

Source: Based on unpublished data by Jean Bennett.

occurred with the writing of James Joyce. Here, the diagonal coefficients for *Ulysses* and *Finnegan's Wake* were both very small, but at least positive.

The success of the method can be judged by picking an author out of the group and by quickly looking along the appropriate horizontal and vertical lines to see if the diagonal term is largest. The test works in all cases included in the table, although the results are a little marginal with James Joyce and E. E. Cummings. At the same time the closeness of these numbers makes the need for high statistical accuracy apparent.

One perplexing result was noticed. Although the writing of Abraham Lincoln demonstrated the highest degree of autocorrelation in Table 2, considerable difficulty was experienced in distinguishing Lincoln's work from the novel *Gadsby* written by Ernest Wright without using the letter e. The failure may be due to the unusually weird nature of the letter correlations in Wright's novel. Evidently, Wright's pair-correlation matrix is somewhat like Abe Lincoln's after you subtract standard English. Nevertheless, the result leads to a certain skepticism of the accuracy of such identification procedures in general.

Wilhelm Fucks[12] (1962) gave an interesting treatment of this type of problem as applied to composer identification in music. Fucks, Moles (1956), and others have pointed out that music by Berg and Webern tends to have a frequency distribution that is more equally distributed than that of Beethoven. However, Fucks himself notes that the correlation of intervals of consecutive tones is very similar within the music of Bach and that of Webern, even though strong differences exist between correlations in Webern and Beethoven.[13]

The general moral of this lesson is that when you see a headline in the evening newspaper such as

Computer Says It's Chopin from Beyond

beware! It might have been written by a bunch of monkeys.

[12] Pronounced "foox."

[13] Obviously this type of identification procedure takes on a much more probable character when strings of letters, or words, or musical phrases are used as the basis for determining the correlations. However, the data-accumulation problems, core requirements, and computing time then become very substantial. A more extended discussion of the composer-style-analysis problem is given in Lincoln (1970); also see Fucks (1968). A collection of papers on literary-style analysis was given by Doležel and Bailey (1969), and an annotated bibliography was prepared by Bailey (1968).

4.11 Language Identification

Having concluded that one primarily finds the statistical structure of the language displayed within the letter-pair-correlation matrix, it is tempting to go on to conclude that it should at least be a trivial matter to recognize the visual patterns characteristic of different languages by graphic display of these matrices.

As will be self-evident from Fig. 4-9, the similarities in the second-order statistical properties of the common Western European languages are far greater than the differences. About all that one can say with confidence from visual displays of the pair-correlation matrices for German, English, French, Italian, Spanish, and Portuguese shown in Fig. 4-9 is that they all represent western European language. To distinguish between them, one again has to compute numerical quantities.

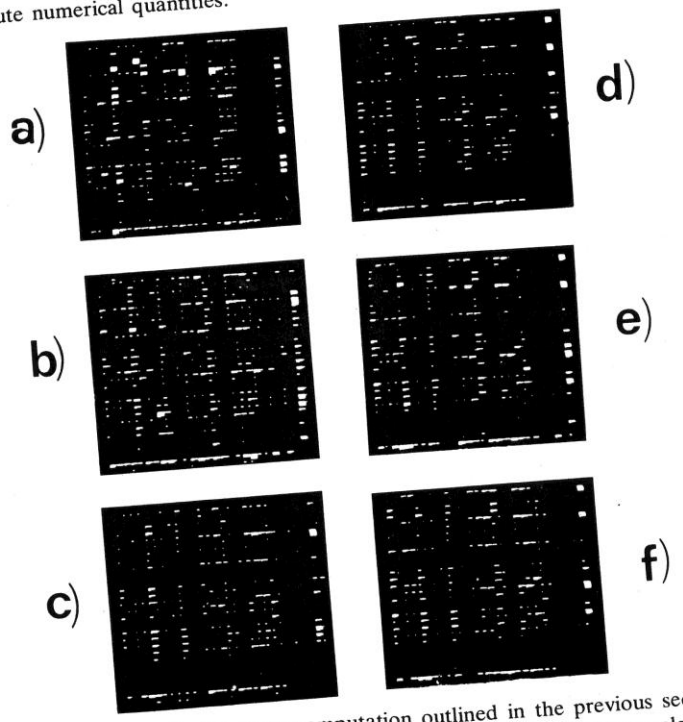


Fig. 4-9. Letter-pair-correlation matrices for different European languages: (a) German (Wiese); (b) English (Shakespeare); (c) French (Baudelaire); (d) Italian (Landolfi); (e) Spanish (Cervantes); (f) Portuguese (Coutinho). In each case a 28×28 raster is used to display the relative probabilities that the j^{th} character follows the i^{th} character. The convention used is the same one explained in Fig. 4-4. (The author is indebted to Jean Bennett and Otto Chu for preparing the data tapes used to generate the displays.) Accent marks were ignored with the exception that umlauts in German were replaced by an additional e.

One could apply the same computation outlined in the previous section. However, it is interesting to try another computed quantity. For example, it is clear that the sum

$$S = \sum_{I,J} [M(I,J) - N(I,J)]^2 \tag{16A}$$

ought to have a minimum value (≈ 0) when $M(I,J) \approx N(I,J)$ for all I and J. Hence if we were to normalize all the matrices to the same total number of characters, we should be able to identify the language from the diagonal terms in the matrix S. (As with the author-identification problem, the sums, S, will comprise a symmetric matrix when the rows and columns are labeled according to the various source languages.) A study of this type has been summarized in Table 3. Although one can clearly distinguish among the source languages, the differences between the two authors writing in English is comparable to the differences between some languages. That is, the English of Shakespeare is quite significantly different from the English of Poe—although not quite as big as the difference between Cervantes writing in Spanish and Coutinho writing in

Eq. (16A)*

	English						
	Hamlet	"Gold Bug"	Spanish	German	French	Italian	Portuguese
English							
Hamlet	0	0.27	0.91	0.88	0.86	0.92	0.94
"Gold Bug"	0.27	0	0.89	0.83	0.80	0.88	0.90
Spanish	0.91	0.89	0	0.95	0.72	0.63	0.56
German	0.88	0.83	0.95	0	0.87	1.01	0.99
French	0.86	0.80	0.72	0.87	0	0.76	0.76
Italian	0.92	0.88	0.63	1.01	0.76	0	0.65
Portuguese	0.94	0.90	0.56	0.99	0.76	0.65	0

Source: Based on unpublished data by one of the author's former students, Otto Chu.

* A normalization procedure based on the total number of characters in each 28×28 pair-correlation matrix was used which gives a maximum possible value of 2 for the sum, S. Note that in this case perfect "identification" corresponds to the value $S = 0$. The statistical uncertainty for each term in the table was ≤ 0.01, based on computed variances for the sum. Two separate authors were used in the case of English (Shakespeare and Poe).

Portuguese. One could, of course, criticize the results on the basis of ignored accent marks. This simplification was made as a practical matter but could be avoided by increasing the size of the character set.

A more sensitive method of applying letter-pair-correlation data to the identification of languages occurs through the computation of most probable digram paths through the matrix. (This notion is discussed in more detail in Section 4.19 on the solution of single-substitution ciphers.) The basic point is that one can construct fairly well defined paths through the character set by looking at the pair-correlation matrix elements in descending order.

For example, consider the following algorithm:

1. Choose I to correspond to the first letter of the common article in the language.
2. Print the alphabetic character for which I stands (in the 1–28 code).
3. Find the maximum $M(I, J)$ in which J has not been previously chosen.
4. Let $M(I, J) = 0$ and let $I = J$.
5. Stop after 28 trips through the loop.
6. Go to step 2.

Application of the algorithm above to the pair-correlation matrices shown in Fig. 4-9 resulted in the following sequences:

English (Poe)	THE ANDISOURYPLF'BJ
German	DER STINGALBUMOCHYPF
French	LE DITANSOURMPHYG
Italian	LA CHERIONTUSP
Spanish	LA DENTOSURICH
Portuguese	LA ESTICORMPUNDJ

Style-dependent differences typically enter at the seventh or eighth place. However, only the first four characters in each string are really needed to distinguish among the above languages, and the first four characters are frequently very well defined statistically, even in texts as short as 200 or 300 characters in length. Sequences of this type also provide a very powerful method for solving single-substitution ciphers without even having to understand the source language of the message (see discussion in Section 4.19).

Ironically, it is much easier to pick out the differences among languages from the first-order statistical properties than from the correlations between pairs of letters. (See Table 4.) For example, we can compute normalized letter-frequency distributions $F_x(I)$ and $F_y(I)$ for the difference characters (I) in the alphabet corresponding to two languages (x and y). The quantity

$$S = \sum_I F_x(I)F_y(I) \tag{16B}$$

will tend to go through a maximum when $x = y$. Equation (16B) is equivalent to a generalized dot product of two multidimensional vectors. Clearly, best results are to be expected when each frequency distribution is normalized so that

$$\sum_I F_x(I)^2 = \sum_I F_y(I)^2 = \cdots = \text{constant (e.g., = 1)} \tag{16C}$$

Then the magnitudes of the generalized vectors are all the same and one is not giving unwarranted weight to a particular language. [Note that the character-frequency data in Table 4 are not normalized according to Eq. (16C).]

4.11: Table 4 Total Character Frequency per 1000 Characters in Order A, B, C, . . . , X, Y, Z, ,' for Several European Languages[a] (See the offer in the Preface.)

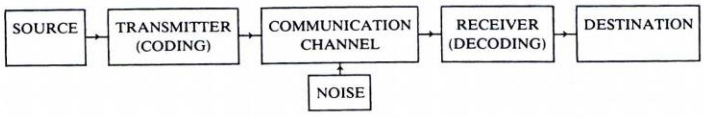
English												
Hamlet												
58	12	17	31	93	18	14	5Ø	49	1	7	35
25	49	73	12	1	45	53	73	29	9	2Ø	1
22	Ø	197	6								
"The Gold Bug"											
62	14	2Ø	35	1Ø6	2Ø	16	47	59	2	5	32
21	54	59	16	1	46	49	76	26	7	18	2
16	1	188	2								
German											
48	16	3Ø	47	144	1Ø	24	44	73	4	8	35
22	78	22	1Ø	Ø	61	69	56	39	7	8	Ø
1	8	137	Ø								
French											
55	7	24	31	152	8	8	8	61	3	Ø	49
26	52	43	26	11	54	74	55	55	11	Ø	4
2	Ø	166	13								
Italian											
111	5	44	27	1Ø1	9	13	17	71	Ø	Ø	41
24	52	74	25	7	49	46	44	23	25	Ø	Ø
Ø	6	181	4								
Spanish											
1Ø6	17	32	45	11Ø	5	9	1Ø	47	4	Ø	52
21	55	78	16	14	52	56	28	37	7	Ø	1
11	4	184	Ø								
Portuguese											
116	3	41	46	1Ø2	8	1Ø	4	66	1	Ø	24
37	5Ø	99	23	5	54	62	44	28	12	Ø	Ø
Ø	3	163	Ø								

[a] The data were computed from the same sources used to determine the matrices displayed in Figs. 4-4 and 4-9. The frequency of the letter e is artificially high in the case of German because umlauts were replaced by e's following the vowel (e.g., ö was replaced by oe in the source text). All other accent marks were merely ignored.

Although the technique may not be so useful in the analysis of ordinary text, it becomes much more impressive when applied to identifying the source language in multiple transposition ciphers (see discussion later of the Pablo Waberski cipher). Note that the same technique could be used in a variety of different applications ranging from problems in pattern recognition (in which sets of expansion coefficients could be used to make up the generalized vector components) to problems in literary-style identification (where the frequency distributions might consist of things such as word, sentence, and paragraph lengths). Also note that the sums in Eqs. (16B) and (16C) can be done simply by matrix multiplication using suitably dimensioned row and column matrices. However, as with the pattern recognition problem discussed in Section 2.23, greater sensitivity is obtained by requiring that the individual projections of the different generalized vectors agree within some appropriate numerical criterion.

4.12 Relation to Information Theory[14]

Having come so close to many of the questions addressed in Shannon's famous (1948) paper on information theory, it would be irresponsible not to say something about the relation of the present material to the general problem of transmitting and receiving information over communication channels. In any real communication system, one is faced with a sequence of the following type:



The general features of this system obtain regardless of whether the transmitter is a scribe taking dictation with pen and ink on sheets of blotting paper or, at

[14] This Section was introduced merely to provide some qualitative background perspective on communication problems. The equations in this Section are not necessary for the later discussion of entropy.

I

IF YOUTH, THROUGHOUT all history, had had a champion to stand up for it; to show a doubting world that a child can think; and, possibly, do it practically; you wouldn't constantly run across folks today who claim that "a child don't know anything." A child's brain starts functioning at birth; and has, amongst its many infant convolutions, thousands of dormant atoms, into which God has put a mystic possibility for noticing an adult's act, and figuring out its purport.

Up to about its primary school days a child thinks, naturally, only of play. But many a form of play contains disciplinary factors. "You can't do this," or "that puts you out," shows a child that it must think, practically, or fail. Now, if, throughout childhood, a brain has no opposition, it is plain that it will attain a position of "status quo," as with our ordinary animals. Man knows not why a cow, dog or lion was not born with a brain on a par with ours; why such animals cannot add, subtract, or obtain from books and schooling, that paramount position which Man holds today.

But a human brain is not in that class. Constantly throbbing and pulsating, it rapidly forms

opinions; attaining an ability of its own; a fact which is startlingly shown by an occasional child "prodigy" in music or school work. And as, with our dumb animals, a child's inability convincingly to impart its thoughts to us, should not class it as ignorant.

Upon this basis I am going to show you how a bunch of bright young folks did find a champion; a man with boys and girls of his own; a man of so dominating and happy individuality that Youth is drawn to him as is a fly to a sugar bowl. It is a story about a small town. It is not a gossipy yarn; nor is it a dry, monotonous account, full of such customary "fill-ins" as "romantic moonlight casting murky shadows down a long, winding country road." Nor will it say anything about tinklings lulling distant folds; robins carolling at twilight, nor any "warm glow of lamplight" from a cabin window. No. It is an account of up-and-doing activity; a vivid portrayal of Youth as it is today; and a practical discarding of that worn-out notion that "a child don't know anything."

Now, any author, from history's dawn, always had that most important aid to writing:—an ability to call upon any word in his dictionary in building up his story. That is, our strict laws as to word construction did not block his path. But in

Fig. 4-10. The first two pages of Chapter I of the novel *Gadsby* (a story of over 50,000 words without using the letter E) written by Ernest Wright (1939).

the opposite extreme, a high-speed telecommunication system in which alphanumeric characters are being transmitted in 8-bit bytes over a microwave link.[15]

In each case, a message from the source is encoded by an established convention, and this code is transmitted in segments (e.g., sheets of blotting paper, 8-bit bytes, etc.) at a prescribable rate. Noise is added to the signal by the coding process itself, but especially in the communication channel (e.g., spreading of the ink in the blotting paper, stray pulses in the teletype link, etc.). The message is then received, decoded, and sent to its final destination.

Interest in this type of problem has existed since the early days of telegraphy. For example, the nontrivial economic problems involved in the transmission of teletype messages over a trans-Atlantic cable stimulated theoretical interest in the quantitative comparison of the efficiency of different coding methods. In fact, there was already considerable interest in the most efficient methods for television transmission over both wire and radio paths by the mid-1920s. Quantitive formulation of the problem dates at least to the early papers of Nyquist (1924) and Hartley (1928), in which it is noted that one should be able to define a quantity

$$H = L_i \log b_i \tag{17}$$

proportional to the amount of information associated with a list of L_i possible selections made in a code of base b_i. Such a definition permits comparing the information transmitted in different base codes (e.g., binary, ternary, . . . , decimal, . . .) and ensures that the information transferred per sample is the same in two different codes when

$$b_1^{L_1} = b_2^{L_2} \tag{18}$$

The base of the logarithm in Eq. (17) is arbitrary. Base 2 logarithms of course make life particularly simple with binary codes, for then the amount of information per sample is just the number of binary symbols used. The name *bit* (short for "binary digit") for this unit of information, suggested by the mathematician John Tukey, has been widely adopted. Similarly, an 8-bit sample is defined as a *byte* in current usage.

It is obvious that the rate at which messages can be sent must increase proportionally with the number of data blocks (8-bit bytes, sheets of blotting paper, etc.) sent per second and that at least a monotonic increase of information transmission capability must occur with increasing signal-to-noise ratio within the transmission of individual data blocks.

Consideration of a binary encoding method provides an easy way to see that the channel capacity to transmit information must increase logarithmically with the ratio of the signal voltage to the noise voltage (or signal-to-noise ratio). We can, in fact, define the capacity of a communication channel to transmit information in terms of the equivalent number of binary bits per second required to send a signal with a given bandwidth and signal-to-noise ratio within that bandwidth. For example, suppose that the signal is a continuously varying voltage which we wish to encode and transmit in a sequence of M-bit samples at the rate of W samples per second. Clearly, the uncertainty in coded signal will have a minimum value of about one bit per sample. Hence the maximum signal-to-noise ratio the signal can have will be limited to

$$S/N \approx 2^M \tag{19}$$

just from the encoding process itself. Hence

$$M = \log_2 (S/N) \tag{20}$$

[15] For a detailed account of real communication systems, see Bennett and Davey (1965).

and the total number of bits that could be transmitted by this system (in W samples per second) is

$$C = WM = W \log_2 (S/N) \qquad \text{bits/sec} \tag{21}$$

The effects of additional noise sources in the communication channel may be included within the term N in the same formula. For example, suppose that we have a signal level of 10 bits per sample and a noise level reaching the receiver of two bits per sample. The ratio of signal-to-noise is then

$$S/N \approx 2^{10}/2^2 = 2^8 \tag{22}$$

Hence the signal could have been transmitted in the presence of coding noise alone with a system having only 8-bit samples, and the channel capacity to transmit information is given adequately by Eq. (21) if we merely insert the actual value of S/N from Eq. (22).

At the receiving end of the communication link, the message must be decoded and the original voltage reconstructed. Because we effectively multiplied the original signal by a periodic wave at frequency W during the encoding process (i.e., we took W samples per second), simple trigonometric identities tell us that extraneous beat frequencies will be present in the received signal at $W \pm W_m$, where W_m is the maximum frequency present in the original signal. Hence to remove these extraneous signals at the receiver, we have to run the output signal through a low-pass filter which cuts off rapidly in frequency above W_m and the sample rate must satisfy

$$W > 2W_m \tag{23}$$

These observations can be extended to continuous-wave-transmission problems with much the same conclusion: that the channel capacity, C, for a continuous-wave communication channel perturbed by frequency-independent noise is also related to the signal-to-noise ratio and the bandwidth of the channel, W, by Eq. (21).

The net bit-transmission rate is called the *entropy rate* or *information rate*. More formally, the channel capacity as defined in information theory turns out to be the greatest entropy rate of source for which codes can be devised that allow the error at the destination to be made arbitrarily small.

Some communication links (e.g., those used to converse with nuclear submarines deep below the ocean surface) have very low channel capacities. To send teletype messages over such a communication channel, it is obviously desirable to encode the original messages with the smallest number of bits possible that still permits unambiguous decoding at the destination. In the next several sections we shall consider what the statistical properties of the language imply regarding the minimum average number of bits per character necessary to transmit the language.

4.13 Entropy in Language

The expression,

$$H = -\sum_{I=1}^{N} P(I) \log_2 P(I) \tag{24}$$

is a fundamental quantity in Shannon's (1948) theory. He concluded that H has the properties of entropy by analogy to the mathematical form of a similar quantity defined by Boltzmann in statistical mechanics (in the formulation of the H-theorem). By useful historical coincidence, the letter H was also defined as the "information" in a $\log_2$ sense in the much earlier paper by Hartley (1928). Equation (24) is introduced in Shannon's paper as an answer to three postulatory requirements on the dependence of the information on the set of probabilities $P(1), \ldots, P(N)$. As Shannon states, it is with the implications of Eq. (24) to specific problems that we are primarily concerned. We shall

the demonstration that this meaning is very reasonable in a number of specific instances.

To make a connection with the earlier papers by Nyquist and Hartley, it is helpful to note that if a particular code uses $B(I)$ bits to transmit the Ith character on a list of N characters, the average number of bits per character required to transmit messages is given by

$$\langle B \rangle = \sum_{I=1}^{N} P(I)B(I) \tag{25}$$

provided the probabilities are normalized so that

$$\sum_{I=1}^{N} P(I) = 1 \tag{26}$$

The quantity H in Eq. (24) therefore corresponds to the statistical average of the number of bits per character necessary to transmit messages in a code for which

$$B(I) \equiv -\log_2 P(I) = \log_2 [1/P(I)] \tag{27}$$

Most real codes used to transmit language text use a constant number of bits per symbol $B(I)$ and result in average values from Eq. (25) which exceed those that would be computed from Eq. (24) for the same probabilities. The average number of bits per character given by Eq. (25) for a given variable-length code could of course be minimized by choosing the factors so that $B(I)$ increases with decreasing $P(I)$. For example, one could try to choose the factors $B(I)$ to approach the dependence in Eq. (27). However, it is difficult to do this without introducing ambiguities in the code meaning and without leaving the system extremely vulnerable to the effects of transmission errors [see Huffman (1952) for one such approach].

The quantity defined in Eq. (27) is literally the number of bits necessary to specify a list of $1/P(I)$ characters. In the special case where

$$P(I) = \text{constant} = 1/N \tag{28}$$

there are N quantities on the list. In this case, definition (27) results in

$$B(I) = \text{constant} = \log_2 N \tag{29}$$

and Eq. (24) just represents the total number of bits necessary to specify N equally probable choices. Hence Eq. (24) reduces to the "information" in the earlier Hartley and Nyquist sense when the probabilities are all the same.

When the $P(I)$ are different, the **quantity H in Eq. (24)** takes on a more generalized meaning and can be shown to be the ***minimum* average number of bits necessary to specify the number of choices at a branch point where N different possibilities occur with different (normalized) probabilities, $P(I)$.**[16] As applied to written language text, Eq. (24) yields an inherent value of the entropy per character which is a characteristic of the language. The values thus obtained are independent of the labeling scheme or the order in which the text is read and are roughly independent of the number of characters assumed in the alphabet as long as the most probable ones occur well within the sum.

It will be helpful to make sure that Eq. (24) makes sense in a few simple cases. For example, consider a situation in which there are two possible choices with equal probabilities,

$$P(1) = P(2) = \tfrac{1}{2} \tag{30}$$

[16] The proof that Eq. (24) actually gives a minimum value is not trivial. See, for example, Gallager (1968) or Ash (1967).

Here

$$H = \tfrac{1}{2}\log_2 2 + \tfrac{1}{2}\log_2 2 = 1 \text{ bit} \tag{31}$$

and Eq. (24) says there will be

$$2^H = 2 \text{ possible choices} \tag{32}$$

Similarly, with four possibilities with equal probabilities

$$P(1) = P(2) = P(3) = P(4) = \tfrac{1}{4} \tag{33}$$

Equation (24) yields

$$H = 4(\tfrac{1}{4}\log_2 4) = 2 \text{ bits} \tag{34}$$

or

$$2^H = 2^2 = 4 \text{ choices}$$

Next suppose that there are two choices in which

$$P(1) = P \quad \text{and} \quad P(2) = 1 - P \tag{35}$$

Then

$$-H = P\log_2 P + (1-P)\log_2 (1-P) \tag{36}$$

If we take the limit as $P \to 0$ in Eq. (36), $-H \to 0 + 1\log_2 1 = 0$ bits. Hence there is only one choice,

$$2^H \to 2^0 = 1 \tag{37}$$

That is, if $P(1) = 0$ in Eq. (35), it means that $P(2) = 1$ and there really is only one possibility. The same situation holds when $P(2) \to 0$ in the above illustration. Equation (36) also yields a maximum value of $H = 1$ (2 choices) when $P = \frac{1}{2} = P(1) = P(2)$.

4.13 PROBLEM 12 Compute the variation of H as a function of P from Eq. (36) for $0.05 < P < 0.95$ in steps of 0.05. Plot the result on the teletype (or, if available, high-resolution display). Note that

$$\log_2 X = \frac{\log_e X}{\log_e 2}$$

It is next of interest to compute the minimum average number of bits, h, per alphanumeric character required to transmit source material written in a language such as English. In accordance with the above discussion, this quantity may be determined through application of Eq. (24) and may be regarded as the entropy or information per character of source text.

The results obtained will obviously be dependent on the statistical properties of the language, and we will get progressive approximations to the answer analogous to the various levels of sophistication used previously in simulating the Eddington monkey. Further, owing to variations in style among various authors, one can never expect to obtain an absolutely precise answer, and there is indeed reason to expect that real languages may actually obey the second law of thermodynamics (see Section 4.14). The answer will always vary somewhat with the particular text. However, as we have shown earlier in this chapter with the author-identification problem, these differences are a small fraction of the main effect. The structure of the language is largely predominant and, in fact, even the differences in statistical structure among the common western European languages are remarkably slight.

The zeroth-order calculation of the entropy per character is, of course, the easiest. Assuming our original 28-character set used to analyze Act III of *Hamlet*, we let $P = \text{constant} = \frac{1}{28}$, and Eq. (24) gives us directly

$$h_0 = \log_2 28 = 4.80735 \text{ bits/character} \tag{38}$$

average number of bits per character necessary to convey the language is given by Eq. (38). Consequently, the minimum number of characters required is $C_0 = 2^{h_0} = 28$.

The first-order calculation of the entropy per character (h_1) requires a knowledge of the total probabilities of occurrence of the individual members of the character set. The latter can be determined for the 28 characters used to analyze Act III of *Hamlet* by reading off the numbers in Table 1 (remembering that they must be normalized), or by summing the rows in the correlation matrix $M(I, J)$ in Fig. 4-6. That is, as previously noted,

$$P(I) = \sum_{J=1}^{28} M(I, J) \Big/ \sum_{I=1}^{28} \sum_{J=1}^{28} M(I, J) \tag{39}$$

Applying Eq. (24) to Act III of *Hamlet* yields

$$h_1 = 4.106 \text{ bits/character} \tag{40}$$

or a minimum list,

$$2^{h_1} \approx 17.21 \text{ characters}$$

4.13 PROBLEM 13 Check the numerical value obtained in Eq. (40) by computing the probabilities from the data in Table 1 (or by summing the columns of the correlation matrix in Fig. 4-6).

In the second-order calculation of the entropy per character (h_2), we have to take into account the probability, $P(I, J)$, that the Jth character followed the Ith character. The particular sum obtained from an expression such as Eq. (25) will vary with the identity of the previous character typed.

Suppose that the Ith character of text has just been typed. The normalized probability $P(I, J)$ that the next character will be the Jth character may be obtained from the correlation matrix in Fig. 4-6 by noting that

$$P(I, J) = M(I, J) \Big/ \sum_{J=1}^{28} M(I, J) \qquad \text{where} \quad \sum_{J=1}^{28} P(I, J) = 1 \tag{41}$$

The average number of bits necessary to specify the number of choices at this point will itself be a function of I. (That is, it depends on the past history and hence the character just typed.) This average number of bits is

$$B(I) = \sum_{J=1}^{28} P(I, J) B(I, J) \tag{42}$$

where $B(I, J)$ also depends on the last character typed; hence for Eq. (42) to be a minimum,

$$B(I, J) = -\log_2 P(I, J) \tag{43}$$

by analogy with Eq. (27).

Finally, we want to find the average of $B(I)$ over all initial characters $I = 1$–28. This average will be a minimum because each $B(I)$ is a minimum. Hence the minimum average number of bits per character necessary to describe the source text (or second-order entropy per character) will be given by

$$h_2 = \sum_{I=1}^{28} P(I) B(I) = \sum_{I=1}^{28} P(I) \sum_{J=1}^{28} P(I, J) B(I, J) \tag{44}$$

where $B(I, J)$ is given by Eq. (43). The probabilities may be obtained from the pair-correlation matrix, $M(I, J)$, through Eqs. (39) and (41). Using the 28×28

matrix from Act III of *Hamlet* (Fig. 4-6), we obtain

$$h_2 = 3.3082 \text{ bits/character} \tag{45}$$

or a minimum number of

$$2^{h_2} = 9.905 \text{ characters}$$

on the average. [Note that $h_2 = h_1$ if $B(I, J) = B(I)$ for all J.]

4.13 PROBLEM 14 Evaluate Eq. (44) using the matrix in Fig. 4-6. Note that it will be easiest to store the quantities $B(I, J)$ in a separate array from that for $P(I, J)$. Also note that the normalizing sums can be computed sequentially with the most efficiency; e.g., one needs S and $S(I)$, where

$$S = \sum_{I=1}^{28} S(I) \quad \text{and} \quad S(I) = \sum_{J=1}^{28} M(I, J)$$

These quantities can be computed as the elements $M(I, J)$ are read into your program. Use a conditional statement to bypass the LOG($P(I, J)$) calculations in cases where $P(I, J) = 0$; these cases are easiest to handle merely by defining the corresponding $B(I, J) = 0$.

The values of h_n should be independent of the direction in which you analyze the language. Check the results for h_2 for *Hamlet* by taking the transpose of the matrix in Fig. 4-6 before computing h_2. (Slight differences may result from rounding errors.)

The computation may, in principle, be extended to higher and higher orders of statistical correlation. For example, at the third order we would have

$$h_3 = \sum_I P(I) \sum_J P(I, J) \sum_K P(I, J, K) B(I, J, K) \tag{46}$$

where

$$B(I, J, K) = -\log_2 P(I, J, K) \qquad \text{etc.} \tag{47}$$

and the probabilities are given in terms of the correlation matrices defined earlier in this chapter. One has to keep increasing the dimensions of the matrices and the process begins to eat up prohibitive amounts of core and computing time. The main point is that with higher and higher statistical correlations included, the smaller the number of bits, or list of characters, that has to be transmitted on the average to convey the original text. For example, already by second order apparently only about one third of the normal alphabet is required on the average to convey English. For estimates of the asymptotic behavior of h_n at large n, see Shannon (1951).

A summary of values of h_n computed by the present author for various languages is given in Table 5.

4.13 PROBLEM 15 Compute values of h_1 and h_2 from the sample of the novel *Gadsby* (written without using the letter e) shown in Fig. 4-10 and compare your results with those for English in Table 5. (Use the CHANGE statement or equivalent.)

4.13 PROBLEM 16 Assume that Morse code takes 3 bits for a dash, 1 bit for a dot, 1 bit for the spaces within letters, and 3 bits for the spaces between letters. How many bits per character would be needed to transmit Hamlet? [Evaluate h_1 using a specific array, $B(I)$, representing the number of bits per character in Morse code.]

4.13 RESEARCH PROBLEM

Braille uses 6-bit "words" in which combinations as well as single letters of the normal written language have separate coded meaning. Hence the average number of bits per character necessary to transmit English will be different in first and second order. What is the value of h_2 for Shakespearean English transmitted in braille?

4.13: Table 5 Values of h_n (Entropy per Character) Computed in Various Orders[a]

	h_1	h_2	h_3
Shannon (1951)			
(27-character alphabet, $h_0 = 4.76$)			
English (contemporary)	4.03	3.32	≈ 3.1[b]
Present results			
(28-character alphabet, $h_0 = 4.807$)			
English			
Chaucer (*Canterbury Tales*)	4.00	3.07	2.12
Shakespeare (*Hamlet*)	4.106	3.308	2.55
Poe ("The Gold Bug")	4.100	3.337	2.62
Hemingway (*For Whom the Bell Tolls* and *A Farewell to Arms*)	4.055	3.198	2.39
Joyce (*Finnegan's Wake*)	4.144	3.377	2.55
German (Wiese)	4.08	3.18	—
French (Baudelaire)	4.00	3.14	—
Italian (Landolfi)	3.98	3.03	—
Spanish (Cervantes)	3.98	3.01	—
Portuguese (Coutinho)	3.91	3.11	—
Latin (Julius Caesar)	4.05_1	3.27_1	2.38
Greek (Rosetta Stone)	4.00_7	3.05_3	2.19
(77-character alphabet, $h_0 = 6.267$)[c]			
Japanese (Kawabata)	4.809	3.633	—

[a] Accent marks were not included in the character set and spaces were inserted between the ancient Greek words on the Rosetta Stone.

[b] The value of h_3 given by Shannon (1951) was based on an extremely approximate method of including the space symbol in earlier trigram data given by Pratt (1939). Because Pratt's data were not terribly accurate in the first place and also did not include correlations with the space symbol, it is surprising that Shannon's estimate of h_3 was as good as it was. Apparently no one has published an accurate computed value for h_3 in any language since the Shannon (1951) publication.

[c] The results for Japanese were computed by one of the author's students, Yoshikazu Okuyama, from a 10,000-character sample using a 77-character set consisting of 76 kana plus the space symbol.

4.14 Entropy and Anthropology

The second law of thermodynamics may be stated

$$\Delta H > 0 \tag{48}$$

for any thermodynamic process where H is proportional to the entropy for the total system.[17] Associating entropy with the degree of statistical disorder, the second law means that thermodynamic systems tend to proceed from states of lower probability to states of higher probability (or, equivalently, from higher to lower order). For example, a drop of ink gradually diffuses throughout the glass of water into which it is placed; molecules having a well-defined velocity will assume a Maxwellian velocity distribution due to collisions in a short time

[17] In many texts on statistical mechanics, H is defined to be proportional to the entropy through a *negative* constant. This difference amounts to changing the sign in Eq. (24). In the present discussion we have adopted Shannon's definition, in which H has the same sign as the entropy.

after they have been placed in a high-pressure gas, and so on. Ultimately, as proclaimed by various morbid prophets of doom, this process will lead to the "heat death" of the universe—unless something we do not know about with much certainty takes place.

There are some qualitative reasons why we might also expect languages to obey the second law in some sense. The fact that large numbers of people use them introduces the statistical element. If a language is developed initially by one or a small number of persons at one point on the globe, it seems inevitable that the structure of the language will become less ordered as it diffuses throughout the world. The condensed (and therefore specialized) meanings originally given to symbols by the creator of the language will tend to be broadened and require more additional description through common usage. In other words, it seems likely that there will be a tendency for the minimum average number of bits per message required to convey meaning in normal use of the language to increase with time.

Some evidence for the effect is to be found in the gradual abandonment of ideographs (symbols that convey entire thoughts or words) with the aging of most languages. Beyond that, there is at least some tendency for the number of characters in the alphabet to increase with time, and for the more concise declensions of single words to be replaced by sequences of words. This process generally makes the language easier to learn and use but also results in requiring more bits per message on the average; the redundancy of the language tends to go up and "Parkinson's Law" seems to be a consequence of thermodynamics. One, of course, has to look over really long periods of time to see if the effect occurs; otherwise, variations in individual style will tend to

Fig. 4-11. The evolution of language has been marked by the gradual abandonment of ideographs for the sake of more generally useful alphabetic notation.