A Formal Theory of Inductive Inference. Part II

Ray J. Solomonoff

Visiting Professor, Computer Learning Research Center
Royal Holloway, University of London

IDSIA, Galleria 2, CH–6928 Manno–Lugano, Switzerland
rjsolo@ieee.org http://world.std.com/~rjs/pubs.html *†

Reprinted from INFORMATION AND CONTROL. Volume 7, No. 2, June 1964, pp. 224–254. Copyright by Academic Press Inc.

4 Applications of the Systems to Various Problems in Induction

The following sections will apply the foregoing induction systems to three specific types of problems, and discuss the "reasonableness" of the results obtained.

Section 4.1 deals with the Bernoulli sequence. The predictions obtained are identical to those given by "Laplace's Rule of Succession." A particularly important technique is used to code the original sequence into a set of integers which constitute its "descriptions" for the problems of Sections 4.2 and 4.3.

Section 4.2 deals with the extrapolation of a sequence in which there are certain kinds of intersymbol constraints. Codes for such sequences are devised by defining special symbols for subsequences whose frequencies are unusually high or low. Some properties of this coding method are discussed, and they are found to be intuitively reasonable. A preliminary computer program has been written for induction using this coding method. However, there are some important simplifications used in the program, and it is uncertain as to whether it can make useful predictions.

Section 4.3 describes the use of phrase structure grammars for induction. A formal solution is presented and although the resultant analysis indicates that this model conforms to some extent to intuitive expectations, the author feels that it still has at least one serious shortcoming in that it has no good means

*This research was supported by AFOSR Contract No. AF 49(638)-376, Grant No. AF–AFOSR 62–377, and Public Health Service Grant No. GM 11021-01.

†Sections 4.1 and 4.2 are more exact presentations of much material in Zator Technical Bulletins 140 and 141 of April 1961 and April 1962 respectively. Part I of the present paper appeared in the March 1964 issue of *Information and Control*.

for giving definitions to subsequences that occur with unusually low frequency. The method of Section 4.2, however, does have this capability though it is in general a less powerful extrapolation method. It is felt that ideally, the method of 4.2 should be a special case of that of 4.3.

This admittedly inadequate formal solution is then applied in an approximate way to the problem of finding a grammar that "best fits" a given set of strings. The results appear to be reasonable, but their plausibility is reduced by the uncertainty of the approximations used.

Sections 4.1, 4.2, and 4.3 give methods of assigning a set of integers to a finite string of symbols drawn from some alphabet. These methods are all invertable, so that given any integer, and the alphabet being used, it is possible to find what string of symbols that integer corresponds to.

The decoding instructions are of the type that can be described to a universal machine. This means that there exists a finite string, S, such that for all integers, B, expressed in binary notation,

$$M_1(\widehat{SB}) = \alpha \tag{15}$$

will give the original string of symbols, α, for which the integer B is the code.

We shall then consider B to be "a code for α." B will *not* be a code for α with respect to M_1 , however, but with respect to M'_1.

M'_1 is defined by the condition that for all possible binary sequences, X,

$$M'_1(X) = M_1(\widehat{SX}) \tag{16}$$

In each of Sections 4.1, 4.2, and 4.3, S will be somewhat different.

Each of these sections will try to show that, with respect to its particular machine, M'_1, the extrapolations obtained using Eq. (1) (which is reproduced below) are reasonable ones.

$$P(a, T, M_1) \equiv \lim_{\epsilon \to 0} \lim_{n \to \infty} \frac{\displaystyle\sum_{k=1}^{r^n} \sum_{i=1}^{\infty} [(1-\epsilon)/2]^{N(S_{TaC_{n,k}})_i}}{\displaystyle\sum_{k=1}^{r^{n+1}} \sum_{i=1}^{\infty} [(1-\epsilon)/2]^{N(S_{TC_{n+1,k}})_i}} \tag{1}$$

Section 3.1.3 discussed machines which "summarized" certain of the statistical properties of a given finite corpus. M'_1 can be considered to be a machine of this type. In Section 4.1, for example, M'_1 "summarizes" certain properties of the general Bernoulli sequence, and is a "useful" summary of the corpus, if that corpus contains many Bernoulli sequences.

If the sequences being extrapolated are, indeed, of the types discussed in each section, then arguments similar to those of Section 3.1.2.3 can be applied to make it plausible that the results are independent of just what machine was used.

4.1 Induction for the Bernoulli Sequence

A Bernoulli sequence is a Markov sequence in which the probability of each symbol is constant throughout the sequence, and is independent of the subsequence preceding that symbol.

The present section will apply Eq. (1) (which is reproduced in the previous section) to the problem of computing the probabilities of successive members of a Bernoulli sequence. This is about the simplest kind of induction problem that exists, and it has been the subject of much discussion (Keynes, 1921).

The result, in the present case, is similar to one obtained by Laplace, which is called "Laplace's rule of succession." One set of assumptions that leads to his result is that the probabilities of the frequencies for each of the symbols in the Bernoulli sequence are initially uniformly distributed between zero and one. Then Laplace's rule gives the "expected value" of the frequency of each type of symbol, after a certain initial subsequence of that entire Bernoulli sequence is known. The ratio of the expected frequencies of the symbols A and B in the entire sequence is

$$\frac{C_A + 1}{C_B + 1} \tag{17}$$

C_A and C_B being the number of occurrences of A and B, respectively, in the known subsequence.

The present analysis is used to illustrate a particularly important kind of coding method. In Sections 4.2 and 4.3 this coding method is generalized to apply to sequences in which are intersymbol constraints of certain kinds.

4.1.1 A Detailed Description of the Coding Method

For any Bernoulli sequence, we will give a method for assigning a set of numbers to that sequence. Each number will be a code for the sequence, so that given any integer, and an ordered list of the symbol types to be used in the sequence, the associated Bernoulli sequence can be uniquely determined.

Later, these code numbers will be used to compute a-priori probabilities of various sequences, and from these, in turn, an expression for conditional probabilities of successive symbols of the sequence will be obtained.

To assign a code number to a Bernoulli sequence, we will first assign an ordered sequence of ordered pairs of integers to the sequence. There will be a pair of integers for each symbol in the original sequence.

Consider the Bernoulli sequence:

$$\alpha \equiv BBABCCABCCBA \tag{18}$$

The only symbols used are A, B, and C.

We will then write the sequence of symbol types, ABC, followed by the original Bernoulli sequence. This gives:

$$\beta \equiv ABC\overset{1}{B}\overset{2}{B}\overset{3}{A}\overset{4}{B}\overset{5}{C}\overset{6}{C}\overset{7}{A}\overset{8}{B}\overset{9}{C}\overset{10}{C}\overset{11}{B}\overset{12}{A} \tag{19}$$

The first symbol of α in Eq. (18) is B. The integer pair assigned to this will be (3,2). The 3, because there are 3 symbols in β before the symbol to be coded. The 2, because the only previous occurrence of B is the second symbol.

The second symbol of α is also B. Its integer pair can be either (4,2) or (4,4). The reason is that in β, both the second and fourth symbols are B.

The integer pair for A, the third symbol of α, is (5,1).

The integer pair for B, the fourth symbol of α, is either (6,2), (6,4) or (6,5).

The integer pair for C. the fifth symbol of α, is (7,3).

One permissible intermediate code for the first five symbols of α is then

$$a \equiv (3,2),(4,2),(5,1),(6,5),(7,3) \tag{20}$$

Since there are two possible choices for the representation of the second symbol of α, and three possible choices for the fourth symbol of α, there are $2 \times 3 = 6$ permissible intermediate codes for the subsequence consisting of the first five symbols of α.

To change the intermediate code,

$$(a_1,b_1),(a_2,b_2),(a_3,b_3),\cdots,(a_m,b_m)$$

into an integer, we use the formula

$$\begin{aligned} k \;&= ((\cdots(((b_m a_{m-1} \;\; + b_{m-1})\, a_{m-2} \\ &\qquad\qquad\qquad + b_{m-2})\, a_{m-3} + b_{m-3})\cdots)\, a_2 + b_2)\, a_1 + b_1 \end{aligned} \tag{21}$$

k is then the integer code number for the sequence, α.

In the case of intermediate code, a, this gives

$$k = (((3 \cdot 6 + 5) \cdot 5 + 1) \cdot 4 + 2) \cdot 3 + 2 = 1400 \tag{22}$$

It is possible to reverse this process and go from k back to the original intermediate code. To do this:

Divide k by a_1 . The remainder is b_1

Divide the quotient by a_2. The remainder is b_2.

Divide the resulting quotient by a_3. The remainder is b_3.

Continue until no more b_i's are obtainable.

In the present case, all a_i's are known in advance.

$$a_i = i + 2 \tag{23}$$

More generally

$$a_i = i + r - 1 \tag{24}$$

where r is the number of symbol types in the Bernoulli sequence.

It is possible to obtain a very simple approximation for the value of k. Expanding Eq. (21), we obtain

$$
\begin{aligned}
k \quad = \quad & b_m a_{m-1} a_{m-2} \cdots a_2 a_1 \\
& + b_{m-1} a_{m-2} \cdots a_2 a_1 \\
& + \cdots \\
& \qquad\qquad + b_3 a_2 a_1 \\
& \qquad\qquad\quad + b_2 a_1 \\
& \qquad\qquad\qquad + b_1
\end{aligned} \tag{25}
$$

Since $a_{m-1} = m + r - 2$ and m will be very large in all cases of interest, we will usually be able to neglect all but the first term, and write

$$k \approx b_m a_{m-1} a_{m-2} \cdots a_2 a_l \tag{26}$$

It will be noted in Eq. (25) that it is possible for b_{m-1} to equal a_{m-1} and for b_m to equal unity. In this case, it is clear that the second term is as large as the first, though, as before, the rest of the terms are negligible if m is large.

In the applications of the approximation of (26), however, (i.e., in expressions (37) and (38)) a summation is taken over many k's, and all values of b_m between 1 and $m + r$ are used. The result is that the approximation of Eq. (26) is valid for almost all terms of the sums, and the cases where it is not valid contribute a negligible part to the sums.

4.1.2 Conditional Probabilities in the Bernoulli Sequence

Let us now consider the problem of determining the relative probabilities of various possible continuations of the m symbol sequence, α. In the following discussion, we shall allow α to have only 3 different symbol types, A, B, and C. The discussion, however, holds equally well if there is any finite number of symbols in the alphabet.

First we will rewrite Eq. (1), so it will give the relative probability that A rather than B, will follow the sequence, α.

$$\lim_{\epsilon \to 0} \lim_{n \to 0} \frac{\displaystyle\sum_{j=1}^{r^n} \sum_{i=1}^{\infty} [(1-\epsilon)/2]^{N_{(S_{\alpha AC_{n,j}})_i}}}{\displaystyle\sum_{j=1}^{r^n} \sum_{i=1}^{\infty} [(1-\epsilon)/2]^{N_{(S_{\alpha BC_{n,j}})_i}}} \tag{27}$$

r is the number of symbols in the alphabet — which is three, in the present case. $C_{n,j}$ is as in Eq. (1) (which is reproduced in Section 4), the jth of the r^n possible sequences of n symbols. $N_{(S_{\alpha AC_{n,j}})i}$ is the number of bits in the ith possible code of the sequence $\alpha AC_{n,j}$. Essentially, this equation says that we should consider all possible continuations of the sequence to a distance of n symbols into the future.

Then the relative probability of the symbol A following the sequence α, rather than the symbol B following α, will be approximated by the total a-priori probabilities of all sequences of length $m+n+1$ that start out with the sequence αA, divided by the corresponding expression for sequences that start out with αB.

The approximation becomes exact as n approaches infinity.

In the present case, it will become clear that the value of the approximation is constant for $n \geq 1$ so we will consider only $n = 1$.

For the a-priori probability of the jth sequence of $m+1+n$ symbols, we will use the approximation,

$$\sum_i \frac{1}{(k_{i,j})^{1+\delta}} \tag{28}$$

$k_{i,j}$ is the ith code number that is a legal representation of that sequence. δ is defined to be $-log_2(1-\epsilon)$, so $\delta \to 0$ as $\epsilon \to 0$.

It will be noted that this expression differs from that used in Eq. (27). There, the expression used was equivalent to

$$\sum_i \big(\frac{1}{2}(1-\epsilon)\big)^{N(k_{i,j})} \tag{29}$$

Here, $N(k_{i,j})$ is the number of bits in the binary expression for the integer $k_{i,j}$. We will approximate $N(k_{i,j})$ by $log_2 k_{i,j}$. This then gives for a-priori probability,

$$\sum_i \frac{1}{2}^{log_2 k_{i,j}} (1-\epsilon)^{log_2 k_{i,j}} \tag{30}$$

If we set $1-\epsilon \equiv 2^{-\delta}$, we obtain

$$\sum_i \frac{1}{2}^{log_2 k_{i,j}} \cdot 2^{-\delta log_2 k_{i,j}} = \sum_i \frac{1}{(k_{i,j})^{1+\delta}} \tag{31}$$

Setting $n = 1$, and using δ instead of ϵ, we obtain the approximation

$$\lim_{\delta \to 0} \frac{\sum_{j=1}^{r} \sum_i 1/k_{i,j}^{(1+\delta)}}{\sum_{j=1}^{r} \sum_i 1/k_{i,j}^{\prime(1+\delta)}} \tag{32}$$

for Eq. (27). $\beta_j (j = 1, 2 \cdots r)$ is the jth symbol of the alphabet; $k_{i,j}$ is the ith code of the sequence $\alpha A \beta_j$; $k'_{i,j}$ is the corresponding expression for $\alpha B \beta_j$.

The summation on i is taken over a finite number of terms, since in the present case there are only a finite number of codes corresponding to each $\alpha A \beta_j$, and each $\alpha B \beta_j$.

In order to obtain the necessary total a-priori probabilities, let us first consider the intermediate code expressions for $\alpha A \beta_j$, where β_j may be A, B, or C. From (26), such a code will have approximately the number

$$k = 3 \cdot 4 \cdot 5 \cdots (m+r-1)(m+r) \cdot b_{m+2} = \frac{b_{m+2}(m+r)!}{(r-1)!} \tag{33}$$

assigned to it, and the a-priori probability of this code will be approximately

$$\frac{1}{k^{1+\delta}} = \left(\frac{(r-1)!}{(m+r)!\, b_{m+2}} \right)^{1+\delta} \tag{34}$$

The value of b_{m+2} can be any integer from 1 to $m+r+1$. It will be seen that if we fix the value of b_{m+2} then there will be just $C_A!\, C_B!\, C_C! \cdot (C_A+1)$ different possible intermediate codes that start out with αA. Here C_A, C_B, and C_C are the number of times that A, B, and C, respectively, occur in α. The reason for this particular number is that when the hth occurrence of A, say, is being coded as a pair of integers, there will be only one possible choice for a_i, the first integer, but there will be just h possible choices for b_i, the second integer. Each such choice of b_i results in an acceptable intermediate code for the same sequence. The a_i and b_i are those of Eq. (21).

The total a-priori probability of all sequences of $m+2$ symbols that begin with αA, and have the same value of b_{m+2} , will be

$$C_A!\, C_B!\, C_C!\, (C_A+1) \left(\frac{(r-1)!}{(m+r)!\, b_{m+2}} \right)^{1+\delta} \tag{35}$$

To obtain the total a-priori probability in the numerator of Eq. (32), we sum over all possible values of β_j and hence over all possible values of b_{m+2}. The resultant total is

$$C_A!\, C_B!\, C_C! \left(\frac{(r-1)!}{(m+r)!} \sum_{i=1}^{m+1+r} \frac{1}{i} \right)^{1+\delta} \cdot (C_A+1) \tag{36}$$

The corresponding expression for the denominator of Eq. (32), in which the $(m+1)$th symbol of the coded string must be B, is

$$C_A!\, C_B!\, C_C! \left(\frac{(r-1)!}{(m+r)!} \sum_{i=1}^{m+1+r} \frac{1}{i} \right)^{1+\delta} \cdot (C_B+1) \tag{37}$$

The relative probability that α will be continued by A rather than B, is the ratio of (36) to (37), i.e.,

$$\frac{C_A+1}{C_B+1} \tag{38}$$

In the above discussion we let $n=1$. If we allow n to have a greater value, expressions (36) and (37) each are multiplied by the same correction factors, leaving their ratio (expression (38)), invariant.

It will be noted that expression (38) is identical to that obtained through "Laplace's rule of succession."

It may seem unreasonable to go through this rather arduous process to obtain such a simple result — one that could be otherwise obtained from far simpler assumptions. However, the present demonstration is an illustration of the application of a very useful coding method. It is important that this coding method should give reasonable results in this particular case, since it is later used as the basis for more complex coding methods. In Section 4.2, we shall generalize it, so that it may deal with descriptions that utilize definitions of subsequences that occur with significant frequencies.

For large values of m, it is easy to obtain another interesting property of Eq. (36). If we take the log_2 of (36), and divide by $m+1$, we obtain for large m and small δ, using Sterling's approximation for $X!$,

$$\frac{(C_A+1)}{m+1} log_2 \frac{(C_A+1)}{m+1} + \frac{C_B}{m+1} log_2 \frac{C_B}{m+1} + \frac{C_C}{m+1} log_2 \frac{C_C}{m+1}$$

This is proportional to Shannon's expression for the entropy of the sequence, and has its minimum value when

$$1 + C_A = C_B = C_C$$

It is clear, then, that the a-priori probability (36) is minimal when the frequencies of all of the symbols are identical, but that the a-priori probability increases when there is any deviation from this. We may view the case in which $1 + C_A = C_B = C_C$ as the "purely random" situation, and any deviation from this as being a "regularity" in the sequence. Any "regularity" of this sort will then increase the a-priori probability of the sequence.

The demonstration in this section uses two important approximations — that of Eq. (26), and the approximation of $N(k_{i,j})$ by $log_2\ k_{i,j}$. Since these approximations have not been shown to be rigorously applicable to the present problem, the results obtained must, in turn, be regarded as being of not absolutely certain validity.

4.2 Induction for Sequences with Certain Intersymbol Constraints

The present section deals with sequences of symbols such as Markov chains, in which the probability of a particular symbol occurring at a particular point depends on the nature of symbols in the sequence that are close to it. If the entire sequence is very short, only very local interactions are considered. For longer sequences, more distant interactions are automatically considered.

Basically the coding method consists of defining certain sequences of two symbols — such as AB or DB or AE — to be represented by special single symbols, such as α, β, or γ. Using these newly defined symbols, the original sequence can be rewritten more compactly. However, the gain in compactness may be offset by the amount of information needed to write the definitions of the sequences defined.

The coding method of inductive inference gives a unified measure to the "increase in compactness" brought about by the introduction of a particular definition, and includes the cost of defining the new symbol.

The method to be described begins by considering all possible pairs of symbols. The pair for which the total decrease in "coding cost" is maximum is then assigned a special symbol, and the original sequence is rewritten using this special symbol. At the next stage all pairs of symbols (including the newly defined special symbol) are examined, and the pair for which decrease in coding cost is maximum is assigned a new special symbol. The sequence that is the result of the last rewriting is again rewritten using the new symbol.

This process is repeated again and again until it is no longer possible to find a new definition that results in a further reduction of "coding cost."

From the compact code that results we are able to find the a-priori probability of the original sequence. This a-priori probability can then be used to find the conditional probability of a symbol occurring at a particular point, in view of the nature of the symbols occurring near it in the sequence.

Section 4.2.1 describes an "intermediate code" in which new symbols are defined. Each of these new symbols represents a sub-string of symbols of the original sequence.

Section 4.2.2 shows how a-priori probabilities are to be assigned to these intermediate codes.

Section 4.2.3 discusses the use of these codes for computing approximate probabilities. *A* "hill-climbing" method is described by which codes of high a-priori probability may be found.

Section 4.2.4 discusses several approximation formulas for the increase in a-priori probability associated with each possible step on the "hill."

Section 4.2.5 discusses a computer program that was written to implement the hill climbing routine of Section 4.2.3.

4.2.1 An Intermediate Code Employing Definitions of Subsequences

In the present paper we shall consider only definitions that involve the concatenation of two symbols. Since either or both of the symbols may in turn represent a concatenation of two symbols it is clear that we can in this way define sequences containing any desired number of symbols of the type used in the original uncoded sequence.

Suppose we have the sequence

$$CABCBABBABABAAB \tag{39}$$

Clearly, if we define the symbol α to represent the subsequence AB we can write (39) more compactly as

$$C\alpha CB\alpha B\alpha\alpha A\alpha \tag{40}$$

However, in order to include all the information in our intermediate code, we must include the definition in our description of (39). A more complete code would then be

$$AB, C\alpha CB\alpha B\alpha\alpha A\alpha \tag{41}$$

Here the comma is a special symbol. It occurs in every intermediate code once, and only once. There are an even number of symbols before the comma, and adjacent pairs of these symbols are the definitions of the respective Greek letters.

The intermediate code

$$AB\alpha A, C\beta A\alpha\alpha \tag{42}$$

would represent the fact that α is defined to be AB, and β is defined to be αA in the sequence

$$C\beta A\alpha\alpha$$

It is clear, then, that the sequence represented by (42) is

$$CABAAABAB \tag{43}$$

4.2.2 Assignment of A-Priori Probabilities to Intermediate Codes

To obtain the a-priori probability of the sequence (43) we will represent its intermediate code (42) by a (usually large) positive integer in a manner that is similar to the coding method described in Section 4.1.1.

Let us first number the symbols of (42) so that we may more easily discuss them.

$$\begin{array}{rcccccccccc} \text{Symbol Nos.} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ ABC, & A & B & \alpha & A & , & C & \beta & A & \alpha & \alpha \end{array} \tag{44}$$

The symbols "ABC," have been written before the sequence proper as we have done in Section 4.1.1.

In coding the first symbol of (44), we assign an a-priori probability of $\frac{1}{4}$ to the symbol A. This is because there are four previous legal possibilities for that symbol, and only one of them is "A". The symbol "," in this position would indicate that no definitions were to be made in this intermediate code.

In coding the second symbol "," is not a possibility since there must be an even number of symbols in the "definitions" section of our code. The legal symbol choices occurring before the second symbol are four in number: A, B, C, and A. Of these, only one is B so we assign an a-priori probability of $\frac{1}{4}$ to B.

Since our first definition is now completed — the definition of α — we have a new symbol that is now possible. So (44) must be rewritten as

$$\begin{array}{rcccccccccc} \text{Symbol Nos.} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \alpha ABC, & A & B & \alpha & A & , & C & \beta & A & \alpha & \alpha \end{array} \tag{45}$$

In coding the third symbol ","and "α" are both legal, so we have seven legal possibilities, of which only one is α, so we obtain the a-priori probability $\frac{1}{7}$.

To code the fourth symbol we have seven legal possibilities (since "," is not legal here). Of these, two are A's, so we obtain the a-priori probability $\frac{2}{7}$.

In coding the fifth symbol we must rewrite (45) since we have completed the definition of β, and so β is now a possible choice.

$$\begin{array}{rl} \text{Symbol Nos.} & 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10 \\ \beta\alpha ABC, & A\ B\ \alpha\ A\ ,\ C\ \beta\ A\ \alpha\ \alpha \end{array} \tag{46}$$

For the fifth symbol there are just ten legal previous possibilities. Only one of them is "," so its probability is $\frac{1}{10}$.

For the sixth symbol, and all subsequent symbols, "," is no longer legal since it can occur only once in the code. Therefore, the probability for the sixth symbol is $\frac{1}{9}$.

The probabilities of the seventh and eighth symbols are obtained straightforwardly — they are $\frac{1}{10}$ and $\frac{3}{11}$, respectively.

The ninth symbol brings up an interesting and very important point. If we have made the definition $\alpha \equiv AB$, then in our subsequent code, the symbol B should *never* follow A, since it would be more economical to rewrite the pair as α. In general, every definition that we make imposes some constraints on which symbols may follow which in the subsequent code.

In the present case, the ninth symbol cannot be B or ",". The resultant probability is then $\frac{2}{10}$.

The tenth symbol cannot be A since it follows α, and $\beta \equiv \alpha A$ has been defined. The probability assigned to the tenth symbol is therefore $\frac{3}{9}$.

The final a-priori probability of the sequence is the product of the individual probabilities that were described, i.e.,

$$\frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{7} \cdot \frac{2}{7} \cdot \frac{1}{10} \cdot \frac{1}{9} \cdot \frac{1}{10} \cdot \frac{3}{11} \cdot \frac{2}{10} \cdot \frac{3}{9}$$

4.2.3 Use of the Codes for Prediction

Suppose we have a string of symbols which we will denote by Σ and we want to know the relative probability of the symbol A rather than B, being the symbol that follows Σ.

Equation (1) computes this probability ratio by considering all codings of all possible continuations of the sequences ΣA and ΣB. In general, a given sequence can be coded in many ways. Various sets of definitions can be used, and they can be defined in different orders — e.g., AB can be defined before BC, or vice versa. Also, with a fixed set of definitions, it is often possible to code a sequence in several different ways. As an example, suppose we have defined $\alpha \equiv AB$ and $\beta \equiv BC$. Then the sequence ABC can be coded as either αC or $A\beta$.

An approximation to the desired probability ratio can be obtained by considering only a few codings of only a few possible continuations of these two sequences. Greater accuracy will, of course, be obtained if more codings and more possible continuations are considered, and if the coding methods used are of relatively high a-priori probability.

A computer program has been written for prediction in which only the sequences ΣA and ΣB are coded and only one method of coding is used for each of them. Possible future continuations are not considered.

The input data consists of a sequence of alphabetic symbols. The output consists of (1) an ordered set of definitions of ordered symbol pairs. (2) the intermediate code of the original sequence, using these definitions, and (3) the a-priori probability of that code sequence.

We need only the third of these outputs to make probability estimates. Suppose that for the string of input symbols designated by Σ the machine gives us a code whose a-priori probability is $P(\Sigma)$. If the symbols used in the original sequence are A, B, and C, then an approximation to the probability that A will be the next symbol to follow the sequence Σ is given by

$$\frac{P(\widehat{\Sigma A})}{P(\widehat{\Sigma A}) + P(\widehat{\Sigma B}) + P(\widehat{\Sigma C})}$$

Here $\widehat{\Sigma A}$ is the concatenation of Σ and A. An attempt is made to find codes of very high a-priori probability by a process of "hill climbing" — i.e., the original sequence is used as an initial code and improvements are made in this code so as to increase the a-priori probability. This results in a new code which is in turn improved. This process of improvement continues until no new improvements can be found within the set of improvement types being considered.

In the particular computer program that was used, each "improvement" consists of devising a new binary definition and rewriting the previous intermediate code using this new definition. If there are r symbol types in the original sequence, and c definitions have been introduced thus far, then $(r + c)^2$ possible definitions are considered. The definition that results in the largest increase of a-priori probability of the intermediate code is used for recoding. Next, $(r+c+1)^2$ definitions are considered and the optimum one is selected.

The process of determining how much the introduction of a given definition will increase the a-priori probability of a code has been studied at length. Several approximate formulas have been obtained for the resultant change in a-priori probability. The approximations are easily implemented by a digital computer, so that with an initial symbol string of about 1000 symbols it takes only about two minutes for the IBM 7090 to find as many as 100 new optimum definitions.

The advantage of the present method of prediction over conventional methods using the frequencies of n-gm of fixed length, is that the present method is able to propose rather complex definitions and evaluate their significance on the basis of a relatively smaller amount of data. Using the same amount of data, the new method *should* be able to make better predictions than more conventional methods.

4.2.4 Approximation Formulas for Hill Climbing

In general, there are two situations in which it is useful to define the subsequence AB.

In the first kind of situation, we have a long uncoded sequence in which the subsequence AB never (or very infrequently) occurs. Defining the pair AB will then increase the a-priori probability of the resultant intermediate code. This is because of increased knowledge of symbols following A's due to the fact the B's are impossible there. This knowledge results in greater probabilities being assigned to the symbols that actually do follow A in the intermediate code.

In the other possible situation, B almost always follows A whenever A occurs. We can then increase the a-priori probability of our intermediate code by defining AB, because the symbol α will have a greater probability than the product of the probabilities of the symbols A and B.

If f_A and f_B are the respective relative frequencies with which the symbols A and B occur in a long, uncoded sequence, and f_{AB} is the relative frequency of the subsequence, AB (relative frequency is in all three cases measured with respect to the total number of single symbols in the sequence), then the ratio of total increase in a-priori probability of the intermediate code resulting from our formulating the definition $\alpha \equiv AB$, is roughly approximated by

$$f_A \cdot f_B \, exp\left[\frac{mf_{AB}}{2}\left(\frac{f_A \cdot f_B}{f_{AB}} - 1\right)^2\right] \tag{47}$$

Here, m is the number of symbols in the original sequence and so mf_{AB} is the number of times AB has occurred.

We will want to consider defining $\alpha \equiv AB$, if this definition gives us an intermediate code of higher a-priori probability — i.e., if expression (47) is greater than unity.

It is of value to regard expression (47) as composed of two factors. First the factor $f_A \cdot f_B$, which is the cost (in probability) of writing the definition $\alpha \equiv AB$. Next, the factor

$$exp\left[\frac{mf_{AB}}{2}\left(\frac{f_A \cdot f_B}{f_{AB}} - 1\right)^2\right] \tag{48}$$

which tells us how much benefit is obtained from the definition increasing the probabilities of various symbols.

In (48) note the presence of the expression

$$\left(\frac{f_A \cdot f_B}{f_{AB}} - 1\right)^2$$

This indicates that if there is any *constraint* between the symbols A and B, so that $f_{AB} \neq f_A \cdot f_B$ (i.e., A and B are not "independent"), then, if our sequence is long enough, (i.e., m is large enough) expression (48) will become very large — so that we will save more than we lose by defining $\alpha \equiv AB$.

We may write (48) as

$$\left\{exp\left[\frac{f_{AB}}{2}\left(\frac{f_A\cdot f_B}{f_{AB}}-1\right)^2\right]\right\}^m$$

Here it becomes clear that no matter how much A and B appear to be dependent (i.e., that f_{AB} differs very much from $f_A\cdot f_B$), it will not be worth while to define $\alpha\equiv AB$ unless the sequence is long enough to give us an adequate "sample size" (i.e., m is large enough). Conversely, even if A and B are only very slightly dependent, it will be worth while to define $\alpha\equiv AB$ if m is large enough.

Also note that if A and B are rather uncommon symbols (i.e., f_A and f_B are small) the cost of defining AB is very important (i.e., $f_A\cdot f_B$ is much smaller than unity), so that the other factor, (48), has more "work" to do.

In general, in the "coding method of inductive inference," it will often be possible to divide the coding effects of a supposed "regularity" in a body of data into a part that corresponds to the cost of defining the regularity, and a part that tells how much we increase the a-priori probability of the code by using this regularity in recoding the data. In Eq. (47) these parts are exemplified by $f_A\cdot f_B$ and by expression (48), respectively.

It should be noted that the approximation expression (47) does not work well for very small values of f_{AB}, and no conclusions should be drawn about this case from this particular approximation.

Expressions (49) and (50) are more exact expressions for the ratio of increase of a-priori probability of an intermediate code that results from defining $\alpha\equiv AB$. These expressions were the ones used in the computer program. Expression (49) is for the case $A\not\equiv B$, and (50) is for the case $A\equiv B$ — i.e., $\alpha\equiv AA$. Although both expressions are approximations, they work very well, even when $N_{AB}=0$ or $N_{AA}=0$.

$$\frac{(N_A-N_{AB}+1)!(N_B-N_{AB}+1)!\cdot(m+r+c-1)!N_{AB}!(r+3c)}{N_4!N_B!(m+r+c-N_{AB}+2)!}\cdot\left(\frac{m+r+c-N_{AB}+3}{m+r+c-N_B+1}\right)^{N_A-N_{AB}} \tag{49}$$

$$\frac{(N_A-2N_{AA}+2)!N_{AA}!(m+r+c-1)!(r+3c)}{N_A!(m+r+C-N_{AA}+2)!}\cdot\left(\frac{m+r+c-N_{AA}+3}{m+r+c-N_A-1)}\right)^{N_A-2N_{AA}} \tag{50}$$

m is the total number of symbols in the original sequence. r is the number of different symbol types in the original sequence. c is the number of definitions that have been already introduced. N_A is the number of times "A" occurs in the code sequence (or original sequence) being recoded. N_B is the corresponding number for "B". N_{AB} is the corresponding number for the subsequence "AB." N_{AA} is the corresponding number for the subsequence "AA."

Here N_{AA} is defined to be the largest number of non-overlapping "AA's" that can be found in the sequence of interest.

In expression (49), the factor

$$\left(\frac{m+r+c-N_{AB}+3}{m+r+c-N_B+1}\right)^{N_A-N_{AB}} \tag{51}$$

gives the amount of increase in a-priori probability due to symbols following A having somewhat greater probability than before — since it is now known that B cannot follow A. This factor is but an approximation, and assumes that this increase in probability is the same for each of the $N_A - N_{AB}$ occurrences of "A" in the new code. The rest of expression (49) is exact, however. Corresponding remarks are true of expression (50).

4.3 The Use of Phrase Structure Grammars in Coding for Induction

The present section deals with the extrapolation of sets of strings in which the constraints among the symbols are somewhat like those that exist among the words of sentences in European languages.

More specifically, suppose we are given an unordered set of strings, $[\alpha_i]$, $(i = 1, 2 \cdots n)$, and we want the relative probability that the new string β_1, rather than the new string β_2 "belongs" to the set.

The method that will be used is equivalent to finding a PSL (context free phrase structure language, Chomsky (1956)) that in some sense best "fits" the set $[\alpha_1]$. The measure of goodness of fit will be the product of the a-priori probability of the language selected and the probability that the language selected would produce $[\alpha_1]$ as a set of acceptable sentences.

The a-priori probability of a PSL will be obtained by writing its grammar as a string of symbols, and using any of the induction techniques of Section 3 on this string. The probability that a given PSL produced $[\alpha_1]$, is a concept that is less clearly defined.

Solomonoff (1959) discusses the concept of "stochastic languages," i.e., languages that assign a probability to every conceivable string rather than the acceptance or rejection which is assigned by ordinary formal languages.

As an example, consider the PSL described by Solomonoff (1959, Appendix II) using a somewhat modified notation. The initial symbol is Σ and the permissible substitutions are:

$$\begin{array}{lll} \Sigma \longrightarrow \alpha\beta & \alpha \longrightarrow \alpha\Sigma C & \beta \longrightarrow C\alpha \\ \Sigma \longrightarrow \beta D & \alpha \longrightarrow C\beta & \beta \longrightarrow B \\ & \alpha \longrightarrow A & \end{array} \tag{52}$$

Here we read "$\Sigma \longrightarrow \alpha\beta$" as "$\Sigma$ may be replaced by $\alpha\beta$." A permissible derivation of an acceptable sentence is:

1. Σ	5. $CCC\beta\beta$
2. $\alpha\beta$	6. $CCCB\beta$
3. $C\beta\beta$	7. $CCCBC\alpha$
4. $CC\alpha\beta$	8. $CCCBCA$

The last string of symbols, $CCCBCA$, is an acceptable sentence, since we can make no further substitutions in it.

It will be noted that for each of the symbols Σ, α, and β, there is more than one substitution possible. By assigning probabilities to each of these substitutions, we describe a stochastic phrase structure grammar. A possible assignment of probabilities in the previous grammar is

$$\begin{array}{lll} \Sigma \longrightarrow \alpha\beta, 0.1 & \alpha \longrightarrow \alpha\Sigma C, 0.2 & \beta \longrightarrow C\alpha, 0.3 \\ \Sigma \longrightarrow \beta D, 0.9 & \alpha \longrightarrow C\beta, 0.2 & \beta \longrightarrow B, 0.7 \\ & \alpha \longrightarrow A, 0.6 & \end{array} \tag{53}$$

The number written after each substitution rule is the probability value assigned to that substitution. In the derivation of the sentence $CCCBCA$ the substitutions $\Sigma \longrightarrow \alpha\beta$, $\alpha \longrightarrow C\beta$, $\beta \longrightarrow C\alpha$, $\alpha \longrightarrow C\beta$, $\beta \longrightarrow B$, $\beta \longrightarrow C\alpha$, and $\alpha \longrightarrow A$ were used in that order. These substitutions have the respective probabilities of 0.1, 0.2, 0.3, 0.2, 0.7, 0.3, and 0.6. The resultant probability of this particular derivation of the sentence $CCCBCA$ is:

$$0.1 \times 0.2 \times 0.3 \times 0.2 \times 0.7 \times 0.3 \times 0.6 = 0.0001512$$

One way to assign an a-priori probability to a stochastic phrase structure language is to first assign an a-priori probability to the corresponding ordinary phrase structure language. Associated with each such ordinary language is a continuous multidimensional space of stochastic languages, each point of which corresponds to a set of possible values of the substitution probabilities. We may assume a uniform a-priori probability distribution over this space.

The problem of assigning an a-priori probability to a stochastic phrase structure language thereby becomes one of assigning an a-priori probability to the corresponding ordinary phrase structure language.

Solomonoff (1959) did not propose any particularly good methods for assigning a-priori probabilities for PSG's. The methods to be described in the present section can, however, be used to find these a-priori probabilities. If these a-priori probabilities were used with the techniques described by Solomonoff (1959, Appendix II), the resultant induction probabilities would be identical to those obtained in the present section.

4.3.1 Generation of the Codes and Their Use for Probability Evaluation

Consider the PSG of Section 4.3.

A permissible derivation of the acceptable sentence, $CCCBCA$ was given in Section 4.3. We can also show this derivation by means of the tree:

$$\begin{array}{ccc} & \Sigma & \\ & | & \\ & \alpha\beta & \\ / & & \backslash \\ C\beta & & C\alpha \\ | & & | \\ C\alpha & & A \\ | & & \\ C\beta & & \\ | & & \\ B & & \end{array} \tag{54}$$

We can write the complete grammar more compactly in the form of the string

$$a\beta, \beta D; \alpha\Sigma C, C\beta, A; C\alpha, B; \tag{55}$$

The two groups of symbols before the first ";" are possible substitutions for Σ. The groups up to the next ";" are the possible substitutions for α, and the next group are possible substitutions for β.

Given the grammar of (55), let us now write a code string for the set of two sentences, $CCCBCA$ and $CABDCD$. The derivation of the first of these is given by (54). The second is given by the tree:

$$\begin{array}{cc} & \Sigma \\ & \downarrow \\ & \beta D \\ & \downarrow \\ & C\alpha \\ & \downarrow \\ & \alpha\Sigma C \\ \swarrow & \downarrow \\ A & \beta D \\ & \downarrow \\ & B \end{array} \tag{56}$$

The first symbol in the code for $CCCBCA$, using derivation (54), is $\Sigma 1$. This indicates that the first substitution for Σ was used. From the notation $\Sigma 1$ alone, we know that the sentence must be out of the form $\alpha\beta$.

The next symbol tells which substitution to make in the leftmost substitutable symbol of the sequence, $\alpha\beta$. The notation $\alpha 2$ indicates that the second substitution for α (i.e., $\alpha \longrightarrow C\beta$) is to be made.

The code sequence $\Sigma 1\alpha 2$ indicates that the sentence is of the form $C\beta\beta$.

$\Sigma 1\alpha 2\beta 1$ is the intermediate code for $CC\alpha\beta$. Again, the newest symbol, $\beta 1$, gives the substitution for the leftmost substitutable symbol in $C\beta\beta$ (whose code is $\Sigma 1\alpha 2$).

Continuing, we have the following set of intermediate codes and partially coded strings:

$$\begin{array}{lcl} \Sigma 1\alpha 2\beta 1\alpha 2 & ; & CCC\beta\beta \\ \Sigma 1\alpha 2\beta 1\alpha 2\beta 2 & ; & CCCB\beta \\ \Sigma 1\alpha 2\beta 1\alpha 2\beta 2\beta 1 & ; & CCCBC\alpha \\ \Sigma 1\alpha 2\beta 1\alpha 2\beta 2\beta 1\alpha 3 & ; & CCCBCA \end{array}$$

After $CCCBCA$ has been entirely coded, it is clear that any further symbols in the code will be for the code of the next sentence. Proceeding as before, to code both $CCCBCA$ and $CABDCD$ using for the derivation of $CABDCD$ the tree of (56):

$$\begin{array}{lcl} \Sigma 1\alpha 2\beta 1\alpha 2\beta 2\beta 1\alpha 3\Sigma 1 & ; & CCCBCA, \beta D \\ \Sigma 1\alpha 2\beta 1\alpha 2\beta 2\beta 1\alpha 3\Sigma 1\beta 1 & ; & CCCBCA, C\alpha D \\ \Sigma 1\alpha 2\beta 1\alpha 2\beta 2\beta 1\alpha 3\Sigma 1\beta 1\alpha 1 & ; & CCCBCA, C\alpha\Sigma CD \\ \Sigma 1\alpha 2\beta 1\alpha 2\beta 2\beta 1\alpha 3\Sigma 1\beta 1\alpha 1\alpha 3 & ; & CCCBCA, CA\Sigma CD \\ \Sigma 1\alpha 2\beta 1\alpha 2\beta 2\beta 1\alpha 3\Sigma 1\beta 1\alpha 1\alpha 3\Sigma 2 & ; & CCCBCA, CA\beta DCD \\ \Sigma 1\alpha 2\beta 1\alpha 2\beta 2\beta 1\alpha 3\Sigma 1\beta 1\alpha 1\alpha 3\Sigma 2\beta 2 & ; & CCCBCA, CABDCD \end{array}$$

Our intermediate code for the pair of sentences $CCCBCA$ and $CABDCD$ is then:

$$\alpha\beta, \beta D; \alpha\Sigma C, C\beta, A; C\alpha, B; \Sigma 1\alpha 2\beta 1\alpha 2\beta 2\beta 1\alpha 3\Sigma 1\beta 1\alpha 1\alpha 3\Sigma 2\beta 2 \tag{57}$$

To compute the a-priori probability of (57) we will use techniques similar to those of Sections 4.1.1, 4.1.2, 4.2.1, and 4.2.2.

To code (57), first write the alphabet symbols, preceded by $\alpha\Sigma$ and followed by ";".

$$\alpha\Sigma ABCD; \tag{58}$$

The symbols of (58) are the only legal initial symbols of an intermediate code, The symbol ";" at the beginning would indicate that no definitions were to be used in the code — that the code was for a Bernoulli sequence as in the sections following (4.1).

The first symbol of a more final code for (57) is the integer "1," indicating the first symbol of (58). The probability of this symbol is $\frac{1}{7}$, since 6 other choices are also legal.

We now rewrite (58) as:

$$\beta\alpha\Sigma ABCD, ; \alpha \tag{59}$$

Now, β, "," and ";" are all legal possibilities. The code for β, the second symbol of (57) is the integer 1, since β is the first symbol of (59), and its probability is $\frac{1}{10}$ since there are 9 other legal choices.

We now rewrite (59) as:

$$\gamma\beta\alpha ABCD,;\alpha\beta \tag{60}$$

The third symbol of (57), i.e., "," is coded by the integer 9, and is given the probability $\frac{1}{12}$.

We now rewrite (60) as:

$$\gamma\beta\alpha\Sigma ABCD\alpha\beta, \tag{61}$$

The "," and ";" are omitted here, since neither "," nor ";" can follow "," immediately. Also, neither "," nor ";" can follow ";" immediately. These are the only constraints that we will use in coding the first 20 symbols of (57), i.e., the "grammar part" of (56).

For β, the fourth symbol of (57), either 2 or 10 may be used, so the probability of β is $\frac{2}{11}$.

Using the above methods, we can code up to the rightmost ";" of (57) to obtain the a-priori probability of the pure grammar string (55). By inspection of the string (55), it is clear that the only symbols that can follow are:

$$\Sigma 1, \Sigma 2, \alpha 1, \alpha 2, \alpha 3, \beta 1 \quad \text{and} \quad \beta 2 \tag{62}$$

String (55) could *not* be followed by more grammar symbols since any new grammar symbols could only serve to define the possible substitution of γ. This would be pointless since γ is not referred to in any of the previous substitutions so there would never be any need to know what its permissible substitutions are. Another way to look at this is to note that there is no chain of conceivable substitutions by which one could start with the symbol Σ and eventually get to a string containing the symbol γ.

To code the part of (57) that follows the grammar description (55) we start by listing the legal symbols, i.e.,

$$\Sigma 1\Sigma 2\alpha 1\alpha 2\alpha 3\beta 1\beta 2 \tag{63}$$

The subsequence $\Sigma 1$ is to be regarded as a single symbol, as are $\Sigma 2$, $\alpha 1$, etc.

The code for $\Sigma 1$ — the first symbol to follow the (56) subsequence of (57) — is 1, since $\Sigma 1$ is the first symbol of (63). Its probability is then $\frac{1}{2}$, since $\Sigma 1$ and $\Sigma 2$ are the only possible symbols at that point. So we write (63) followed by $\Sigma 1$.

$$\begin{array}{ll} \text{Index No.} & 1 \\ \Sigma 1\Sigma 2\alpha 1\alpha 2\alpha 3\beta 1\beta 2 & \Sigma 1 \\ \text{Code No.} & 1 \end{array} \tag{64}$$

The 1 above the terminal symbol of (64) indicates that this $\Sigma 1$ is the first symbol of the latter part of (57).

To code $\alpha 2$, the next symbol of (57), we use the symbol 2 since $\alpha 2$ is the second of the three legal possibilities — i.e., $\alpha 1$, $\alpha 2$, and $\alpha 3$. Its probability is $\frac{1}{3}$.

Index Nos.		1	2
	$\Sigma 1\Sigma 2\alpha 1\alpha 2\alpha 3\beta 1\beta 2$	$\Sigma 1$	$\alpha 2$
Code Nos.		1	7

(65)

In expression (66) we have written the nongrammar part of (57), preceded by (63). The numbers above the latter terms of (66) are simply to be used to refer to the various symbols. The number or numbers below each such term give the code representation or representations. Above the index number of each symbol is written its probability:

Probability	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{2}{4}$	$\frac{1}{3}$	$\frac{2}{4}$	$\frac{1}{5}$	$\frac{2}{3}$	$\frac{3}{5}$	$\frac{1}{6}$	$\frac{2}{7}$	$\frac{1}{4}$	$\frac{2}{6}$
Index Nos.	1	2	3	4	5	6	7	8	9	10	11	12	13
$\Sigma 1\Sigma 2\alpha 1\alpha 2\alpha 3\beta 1\beta 2$	$\Sigma 1$	$\alpha 2$	$\beta 1$	$\alpha 2$	$\beta 2$	$\beta 1$	$\alpha 3$	$\Sigma 1$	$\beta 1$	$\alpha 1$	$\alpha 3$	$\Sigma 2$	$\beta 2$
Code Nos.	1	2	1	2	2	1	3	1	1	1	3	2	2
				4		3		3	3		6		4
									5				

(66)

For an example, let us take the eleventh term, $\alpha 3$. The only possible legal symbols at that point, are α substitutions, so $\alpha 1$, $\alpha 2$ and $\alpha 3$ are the only possibilities. The previous set of α substitutions listed are:

Index Nos.		2	4	7	10
	$\alpha 1\alpha 2\alpha 3$	$\alpha 2$	$\alpha 2$	$\alpha 3$	$\alpha 1$

(67)

The third of sequence (67) and the sixth of that sequence are both $\alpha 3$, so 3 and 6 are both legal codes for $\alpha 3$ at this point.

There are 7 symbols in the sequence (67) so the probability of a3 is just 2 out of 7 or $\frac{2}{7}$.

The other probabilities in (66) are computed similarly. The final probability to be associated with expression (57) is the product of the probabilities obtained in (66) and the probabilities previously obtained in coding the pure grammar string (55).

If it is desired, the methods just discussed can be used to assign large integers to each possible code representation. This process would follow the techniques discussed in Section 4.1.1.

It must be noted that the probability obtained in this way will be that associated with a *particular* grammar and a *particular* pair of derivations for the two sentences of interest. To obtain the *total* probability of these sentences with respect to a particular grammar, it is necessary to sum the probabilities of all possible pairs of derivations for them.

To obtain the total probability of the two sentences with respect to all possible grammars, it is necessary to sum the probabilities over all possible grammars and sets of derivations.

It is probably possible, however, to make satisfactory predictions by using only the "best" grammar and its derivations. This corresponds to predicting using the single "best fitting" model or hypothesis, as is very often done. An example is that of fitting curves to empirical data, in which the single "best fitting" curve is used to make predictions.

The use of string (57) to code the two strings

$$CCCBCA,\, CABDCD \tag{68}$$

and thereby obtain the probability as described would seem to be a rather long way to go about it. Would it not be far easier to code the two strings of (68) as a Bernoulli sequence, and would not the resultant code be shorter and have a higher probability associated with it?

The answer to both of these questions is "yes."

The two strings of (68) would *not* best be coded using the complex PSG of (55). The particular example of strings and grammar used to code them were taken to illustrate the coding method and the method of probability assignment. The problem of *finding* a grammar that can best be used to describe a set of strings (i.e., that results in a description having highest possible a-priori probability) will be discussed in the next section.

4.3.2 A Criterion for "Goodness of Fit" of a PSG to a Particular Set of Strings

In general, the problem of finding the PSG that "best fits" a given set of "acceptable sentences" is not a well defined problem. There are at least two grammars that will always "fit" the sentences in question. The first grammar, which we will call the "promiscuous grammar", is one in which all sequences of symbols drawn from the alphabet are acceptable. For the alphabet of symbols 0, 1, one such grammar is the PSG

$$\begin{array}{ll} \Sigma \to \Sigma\alpha & \alpha \to 0 \\ \Sigma \to \alpha & \alpha \to 1 \end{array} \tag{69}$$

In more compact notation it is:

$$\Sigma\alpha, \alpha; 0, 1; \tag{70}$$

The second type of grammar which we will call the "ad hoc grammar" is the one in which *only* the given symbol sequences are acceptable sentences and *no others*. A PSG of this sort for the acceptable sentences

$$10110 \text{ and } 001111 \tag{71}$$

is

$$\begin{array}{c} \Sigma \to 10110 \\ \Sigma \to 001111 \end{array} \tag{72}$$

In more compact form, the description of the two strings is

$$10110, 001111;\ \Sigma 1 \Sigma 2 \tag{73}$$

These two kinds of grammars are both incapable of making any useful extrapolations. The formal reasons for their inacceptability are, however, different. If there is a large number of strings that we want to fit the grammar to, then using the promiscuous grammar (70) to help describe this set will result in a grammar string of high a-priori probability (i.e., the grammar is "simple"), followed by a descriptive string that is quite long and is of low a-priori probability. The product of these two probabilities is ordinarily *much* lower than the probability obtained by using the "optimum" grammar for description.

On the other hand, the ad hoc grammar of (72) is rather "complex" — it has a low a-priori probability — while the rest of the descriptive string of (73), i.e., $\Sigma 1 \Sigma 2$, is short and has high a-priori probability. The product of these two probabilities is again usually *much* lower than would be the corresponding quantity, using a more "optimum" grammar for description.

We will use the product of these two probabilities (i.e., the a-priori probability of the resultant description) as an approximate criterion of "goodness of fit" of the PSG in question to the given set of strings. A more exact criterion would sum over the probabilities of all possible sets of derivations with respect to the given PSG.

In the following examples, the corpus will only have one set of derivations with respect to the grammars being considered, thus simplifying the discussion somewhat. It should be noted that this situation will not occur when more complex PSG's are considered.

Consider the grammar

$$\begin{gathered} \Sigma \to 0\Sigma 1 \\ \Sigma \to 01 \end{gathered} \tag{74}$$

All acceptable sentences will be of the form $0^{(n)}1^{(n)}$ — which is a sequence of $n0$'s followed by a sequence of $n1$'s. Consider the set of strings:

$$01,\ 0011,\ 00001111,\ 0000011111 \tag{75}$$

We will use three different grammars to describe (75). First the promiscuous grammar (70), then an ad hoc grammar like (72), then its "proper" grammar (74). It will be seen that the description via (74) is of much higher a-priori probability than those corresponding to the other two descriptions. Furthermore, the "degree of betterness" of this description would increase if more strings were added to (75).

Consider first the description via the promiscuous grammar.

Our intermediate code starts with the grammar string of (70). This is followed by $\Sigma 1 \Sigma 2 \alpha 1 \alpha 2$ which describes 01. Next comes $\Sigma 1 \Sigma 1 \Sigma 1 \Sigma 2 \alpha 1 \alpha 1 \alpha 2 \alpha 2$ which describes 0011. Next comes $(\Sigma 1)^{(7)} \Sigma 2 (\alpha 1)^{(4)} (\alpha 2)^{(4)}$ which describes $0^{(4)}1^{(4)}$. Finally comes $(\Sigma 1)^{(9)} \Sigma 2 (\alpha 1)^{(5)} (\alpha 2)^{(5)}$, which describes $0^{(5)}1^{(5)}$.

The probability assigned to the grammar string of (70) is

$$\frac{1}{5} \cdot \frac{1}{7} \cdot \frac{1}{8} \cdot \frac{1}{3} \cdot \frac{1}{10} \cdot \frac{1}{7} \cdot \frac{1}{6} \cdot \frac{1}{8} \cdot \frac{1}{7} \tag{76}$$

In (77) we have written the probability below each of the corresponding grammar symbols.

$$\alpha\Sigma 01,; \quad \begin{array}{ccccccccc} \scriptstyle 1 & \scriptstyle 2 & \scriptstyle 3 & \scriptstyle 4 & \scriptstyle 5 & \scriptstyle 6 & \scriptstyle 7 & \scriptstyle 8 & \scriptstyle 9 \\ \Sigma & \alpha & , & \alpha & ; & 0 & , & 1 & ; \\ \frac{1}{5} & \frac{1}{7} & \frac{1}{8} & \frac{2}{6} & \frac{1}{10} & \frac{1}{7} & \frac{2}{12} & \frac{1}{8} & \frac{2}{14} \end{array} \tag{77}$$

The rest of the description consists of a sequence of $\Sigma 1$'s, $\Sigma 2$'s, $\alpha 1$'s, and $\alpha 2$'s. The a-priori probability of this part is the same as that which would be obtained by considering it to be two Bernoulli sequences — one containing only the symbols $\Sigma 1$ and $\Sigma 2$, the other containing only $\alpha 1$, and $\alpha 2$.

The first Bernoulli sequence contains 20 $\Sigma 1$'s and 4 $\Sigma 2$'s. Its probability is:

$$\frac{(2-1)!\ 20!\ 4!}{(2-1+20+4)!} \tag{78}$$

The second Bernoulli sequence has 12 $\alpha 1$'s and 12 $\alpha 2$'s. Its probability is:

$$\frac{(2-1)!\ 12!\ 12!}{(2-1+12+12)!} \tag{79}$$

Expressions (78) and (79) are similar to expressions (35) and (66) and are obtained using similar reasoning. The final probability obtained for the description of (75) using the promiscuous grammar is the product of the probabilities (76), (78), and (79) — which is approximately 2.3×10^{-21}.

Let us now describe (75) using an ad hoc grammar, i.e.,

$$01,\ 0011,\ 00001111,\ 0000011111;\ \Sigma 1\Sigma 2\Sigma 3\Sigma 4 \tag{80}$$

The first part of the description of (80) is obtained by first writing the legal symbol string $,;\Sigma\alpha 01$, followed by (75).

$$\begin{array}{lcccccccccccccccccccccccccccc} \text{Index nos.} & \scriptstyle 1 & \scriptstyle 2 & \scriptstyle 3 & \scriptstyle 4 & \scriptstyle 5 & \scriptstyle 6 & \scriptstyle 7 & \scriptstyle 8 & \scriptstyle 9 & \scriptstyle 10 & \scriptstyle 11 & \scriptstyle 12 & \scriptstyle 13 & \scriptstyle 14 & \scriptstyle 15 & \scriptstyle 16 & \scriptstyle 17 & \scriptstyle 18 & \scriptstyle 19 & \scriptstyle 20 & \scriptstyle 21 & \scriptstyle 22 & \scriptstyle 23 & \scriptstyle 24 & \scriptstyle 25 & \scriptstyle 26 & \scriptstyle 27 & \scriptstyle 28 \\ ,;\Sigma\alpha 01 & 0 & 1 & , & 0 & 0 & 1 & 1 & , & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & , & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & ; \end{array} \tag{81}$$

In giving probabilities to the first 28 symbols of expression (80), the process is identical to that used in coding a Bernoulli sequence with the 6 symbol alphabet

$$,;\Sigma\alpha 01$$

except that the 4th, 9th, and 18th factors have denominators that are two smaller than in the Bernoulli Sequences and the first factor has a denominator that is one smaller. This is because the symbol "," is not legal in all four of those positions and ";" is not legal in three of them.

The resultant probability is:

$$\frac{(6-1)!\,3!\,1!\,0!\,0!\,12!\,12!}{(6-1+3+1+0+0+12+12)!}\,\frac{1+6}{1+6-1}\cdot\frac{4+6}{4+6-2}\,\frac{9+6}{9+6-2}\,\frac{18+6}{18+6-2} \tag{82}$$

The last four factors of (82) are corrections due to the constraints existing in the coding of the first, fourth, ninth and eighteenth symbols.

Treating the sequence $\Sigma 1 \Sigma 2 \Sigma 3 \Sigma 4$ as a Bernoulli sequence, we obtain for it the probability

$$\frac{1}{4} \cdot \frac{1}{5} \cdot \frac{1}{6} \cdot \frac{1}{7} \tag{83}$$

The final probability of the description of (75) using the grammar

01, 0011, 00001111, 0000011111;

is the product of expressions (82) and 83), which is approximately 3.8×10^{-23}.

Let us now describe (75) using the grammar of (74) . The entire description is

$$0\Sigma 1, 01; \Sigma 2\Sigma 1\Sigma 2\Sigma 1\Sigma 1\Sigma 1\Sigma 2\Sigma 1\Sigma 1\Sigma 1\Sigma 1\Sigma 2 \tag{84}$$

The probability of

$$0\Sigma 1, 01; \tag{85}$$

is obtained by first writing it as

$$\alpha\Sigma, ; 01\, \overset{1}{0}\, \overset{2}{\Sigma}\, \overset{3}{1}\, \overset{4}{,}\, \overset{5}{0}\, \overset{6}{1}\, \overset{7}{;} \tag{86}$$

The product of the probabilities for the various symbols give the probability

$$\frac{1}{5}\, \frac{1}{7}\, \frac{1}{8}\, \frac{1}{9}\, \frac{2}{8}\, \frac{2}{11}\, \frac{1}{12} \tag{87}$$

for (85).

The sequence $\Sigma 2\Sigma 1\Sigma 2\Sigma 1\Sigma 1\Sigma 1\Sigma 2\Sigma 1\Sigma 1\Sigma 1\Sigma 1\Sigma 2$ contains 8 $\Sigma 1$'s and 4 $\Sigma 2$'s. Its probability is therefore

$$\frac{(2-1)!\, 8!\, 4!}{(2-1+8+4)!} \tag{88}$$

The final probability of the description of (75) using the correct grammar, (74), is then the product of expressions (87) and (88), which is approximately 9×10^{-11}.

Let us compare the probabilities obtained via the three grammar types:

1. The promiscuous grammar: 2.3×10^{-21}
2. The ad hoc grammar: 3.8×10^{-23}
3. The "correct" grammar: 9×10^{-11}

It is seen that the probability via the "correct" grammar is *much* greater than the probabilities via the other two. In particular, the probabilities via the first two grammars are very roughly the square of that of the third. This is to a large extent due to the fact that the first two grammars each require about twice as many decisions to be made in their description, as are made in the third grammar.

If we were to forsake PSG's entirely in describing (75) and were to describe it as a Bernoulli sequence with the constraint that the first, fourth, ninth, and eighteenth symbols cannot be ",", the probability assignment would be roughly 4.5×10^{-13}. About as many decisions have to be made as in either of the first two grammars, but each decision is given a higher probability since there are usually not as many alternative choices.

4.3.3 How to Find a PSG That Best "Fits" a Given Set of Strings

In the previous section we had shown how to obtain a probability from a given set of strings, a PSG that could have produced these strings, and a set of legal derivations of the strings from the grammar rules.

From a *formal* point of view, this solves the problem of obtaining a PSG of optimum fit (i.e., highest probability), since we can order all PSG's and their derivations (if any) of the set of strings. We can then find the probability of each PSG, by summing over the probabilities of each of its possible derivations of the set of strings. The PSG of maximum probability may be then selected and used for extrapolation. More exact extrapolations can be obtained by considering all PSG's whose fit is close to that of the "best" PSG. A weighted average of the extrapolations associated with each possible grammar must be used. The weight of a PSG is proportional to the "goodness of fit" of that grammar to the known corpus.

This is not, however, a practical solution. The problem of finding a set of PSG's that "fits well" cannot be solved in any reasonable length of time by ordering all PSG's and sets of derivations, and testing each PSG in turn.

A method that appears to hold some promise of finding a well fitting grammar utilizes a method of digital "Hill climbing" not unlike that used in Section 4.2.3.

We start out with a particular PSG description of the set of sentences — in the cases that have been investigated, the ad hoc grammar was used. A probability is assigned to this description using the methods of Section 4.3.2.

Next, a set of possible "mutations" of the description are considered that leave the set of strings described invariant. Such possible modifications involve the defining of new intermediate symbols in the grammar, the modification or elimination of various grammar rules, etc. Of the set of mutations considered, one is selected that increases the entire description probability most (i.e., it most simplifies the description), and the corresponding modification is made in the description.

From this new description, a new set of mutations are tried and the "best" one is retained. This process of successive mutation and selection is continued until a maximum is reached — at which point the resultant grammar is retained.

For an example of a "mutation," consider the set of strings (75), as described by the ad hoc grammar, i.e., expression (80). We can make a mutation by "factoring out" the symbol 0 from the lefthand side of each of the strings to obtain the description

$$0\alpha; 1, 011, 0001111, 000011111; \Sigma 1\alpha 1\Sigma 1\alpha 2\Sigma 1\alpha 3\Sigma 1\alpha 4 \tag{89}$$

This description must be evaluated to see if it has a higher probability than that of (80). (It should be noted that each occurrence of $\Sigma 1$ in (89) is of probability 1 and so these $\Sigma 1$'s may be considered redundant.)

Other mutations can be devised, utilizing various of the "factoring" and "inclusion" rules discussed by Solomonoff (1960).

The effectiveness of such a "Hill climbing" technique rests largely upon what kinds of mutations are considered in each step. Selection of suitable mutations will also prevent, to some extent, one's getting stuck at "local maxima" rather than reaching the highest point of the "hill."

Another technique of dealing with local maxima is to select the 10 (say) best mutations at each step and proceed with the next set of mutations, always retaining the 10 best descriptions thus far and using them as bases for mutations.

At the present time, a set of mutation types has been devised that makes it possible to discover certain grammar types, but the effectiveness of these mutation types for more general kinds of grammars is as yet uncertain.

4.3.4 A Criticism of the PSG Coding Method

In the coding method of Section 4.2, subsequences of unusually *low* frequency are recognized, and given definitions, thereby increasing the probability of the resultant code. This is accomplished by the equivalent of certain "parsing constraints" so if $\alpha \equiv AB$ has been defined, the sequence AB can only occur as a case of α.

In the PSG coding method that has been described, there are no such "parsing constraints," and as a result, there is no good way to take advantage of the fact that a certain subsequence occurs with an unusually *low* frequency.

It is hoped that further investigation will make it possible to remove this apparent deficiency from the present PSG coding method.

5 Use of the Theory for Decision Making

It is possible to use the probability values obtained through the present theory to make decisions in a variety of ways (Luce and Raiffa (1957), Solomonoff (1957)). Van Heerden (1963) has devised a theory of decision making using of many heuristic arguments that are applicable to the presently described induction theory. He makes decisions as though the prediction given by "the best prediction operator" would indeed occur. "The best prediction operator" is defined to be the one for which the number of bits in the operator's description plus the number of bits in its past errors of prediction is minimum.

In the terms of the present paper, this is equivalent to making predictions on the basis of the shortest description of a corpus and disregarding all other descriptions.

The essential arbitrariness in his theory arises through the arbitrary choice of a language to describe operators. In the present induction theory this corresponds to the arbitrary choice of a universal machine.

RECEIVED: May 24, 1962

REFERENCES

Chomsky, N. (1956), "Three models for the description of language." *IRE Trans. Inform. Theory, Vol* IT–2, 113–124.

Keynes. J. M. (1921), "A Treatise on Probability." Chap. XXX. Macmillan, New York.

Luce. R. D. and Raiffa, H. (1957), "Games and Decisions." Wiley, New York.

Solomonoff, R. J.(1959), "A Progress Report on Machines to Learn to Translate Languages and Retrieve Information." pp. 941–953. Advances in Documentation and Library Sciences, Vol III, Part 2. Interscience, New York. AFOSR TN-59-646 (Contract AF 49(638)-376); ZTB–134.

Solomonoff, R. J. (1960), The Mechanization of Linguistic Learning. *Proc. Second Intern. Congr. Cybernetics, Namur, Belgium, September, 1958,* pp. 180–193. AFOSR TN-59-246 (Contract AF 49(638)-376); ASTIA AD No. 212 226; ZTB–125.

Van Heerden, P. J. (1963), "A General Theory of Prediction." Polaroid Corp. Cambridge 39, Mass. (Privately circulated report).