Comparison and Extension of Theories of Zipf and Halstead

R. E. PRATHER
Computing and Information Sciences, Trinity University, 715 Stadium Drive, San Antonio, Texas 78284, USA

The length estimates of Zipf (behavioural psychology) and Halstead (software science) are shown to emerge from a single empirical law, providing a unified approach to the software metrication problem. Alternatively, both estimates may be derived from a probabilistic hypothesis concerning the programming process. On the other hand, it is shown that these two models (those of Zipf and Halstead) yield quite different token frequency distributions. It is suggested – and empirical evidence serves to support the claim – that the real situation lies somewhere between the two theories. A series of inequalities is proposed for guiding future research along these more realistic lines.

Received June 1986, revised February 1987

1. INTRODUCTION

Beginning in the 1930s, G. K. Zipf developed a theory[1] describing the structure of written or spoken language, one that has since been applied to the study of analogous questions for computer programming languages. In a seemingly unrelated development, the work of M. H. Halstead has led to a theory of 'software physics'[2] designed to address these same questions. We seek to compare and unify these two separate theories and to broaden and extend their application in an empirical software engineering setting.

Whenever the author speaks on this subject, he is reminded of the popular bumper-sticker 'expect a miracle'. For indeed, it seems that something like a miracle is needed if the predictive character of the Zipf–Halstead theory is to be manifested in actual programming practice. That is because we are presented with a means, presumably, for estimating the length of a program *before it has been written*! In fact, we will see that the estimates of Zipf and Halstead can be treated as lower and upper bounds, respectively, on the expected program length. Moreover, we will show that both of these bounds can be obtained from a single unifying hypothesis – that of a 'Zipf's law' distribution of the frequency of occurrence of the program tokens (or alternatively, but perhaps, less believably, from 'Halstead's equiprobability hypothesis'.)

We then proceed to show how the theory may be modified and extended to improve its performance in actual practice. Toward this end, we derive a series of inequalities that will ensure that a model for token frequency distribution will yield results that lie somewhere between those predicted by the Zipf and Halstead theories.

2. ZIPF'S LAW

In any given sampling of linguistic text, whether a newspaper article, an essay, or whatever, it will be observed that certain words (notably the definite and indefinite articles, 'the', 'a', 'an') occur with great frequency whereas others (e.g. 'aardvark' or 'supercalifragilisticexpialidocious') appear only once, if at all. Based on studies such as that shown in Fig. 1, Zipf has further postulated that the overall frequency behaviour

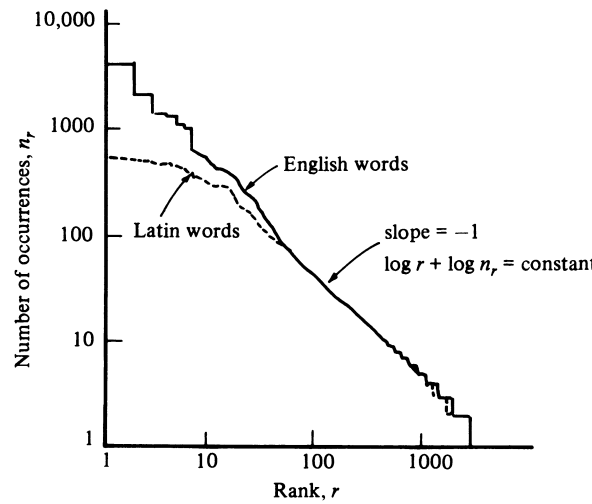

Figure 1.

exhibits a straight line with negative unity in slope when graphed on the log–log coordinates:

$$R = \log r$$
r = rank of token (= 1, 2, ..., t)

$$N_r = \log n_r$$
n_r = number of occurrences of token of rank r

where t (the *vocabulary* size) is the number of distinct tokens appearing in the text. [In the study cited here, the difference in the low-rank limiting behaviour for the Latin text as opposed to the English is due to the fact that in Latin, declension of nouns takes the place of the use of articles as separate tokens.] The behaviour can thus be expressed in the form

$$R + N_r = \text{constant}.$$

When rephrased in terms of the 'natural' coordinates r and n_r, we obtain the *Zipf's law*[1] hyperbolic relationship

$$rn_r = t,$$

since we may evaluate the resulting constant by assuming $n_r = 1$ when $r = t$ (as in the case of 'aardvark' or 'supercalifragilisticexpialidocious').

In a simple counting of the successive ordinates appearing in this relationship, we obtain an estimate for

n (the total *length* of the text) as each token is counted according to its multiplicity of occurrence:

$$n = \sum_{r=1}^{t} n_r = t \sum_{r=1}^{t} \frac{1}{r} > t(\gamma + \ln t).$$

Note. Whereas the infinite sum of $1/r$ diverges, the modified partial sums

$$\sum_{r=1}^{t} \frac{1}{r} - \ln t$$

are known to converge from above to the *Euler constant* $\gamma \approx 0.5772$, and the above estimate then follows immediately.

Example

A newspaper article using $t = 100$ different words should be at least

$$n > 100\ (\gamma + \ln 100) \approx 518$$

words in length – assuming a Zipf law distribution.

In fact, we can show that the Zipf law distribution imposes an upper bound on the length as well. Imagine a grade-school student with a limited vocabulary who has been asked to write an essay. In principle, arbitrarily long essays could be written, but in fact, nothing like this is observed. It is as if the vocabulary size acts as a limiting factor. One could argue that this is somehow related to the estimate:

$$n = \sum_{r=1}^{t} n_r = t \sum_{r=1}^{t} \frac{1}{r} < t(1 + \ln t) < t \log_2 t$$

(the latter inequality valid for all $t \geqslant 9$).

Note. The first inequality follows directly from the definition of the natural logarithm.

Combining our two estimates we obtain

Theorem 1 (Zipf–Halstead)

Assuming a Zipf law distribution for the frequency of occurrence of tokens, we obtain the *bounds on length*:

$$t(\gamma + \ln t) < n < t \log_2 t$$

(used as *estimates of the length* by Zipf and Halstead, respectively).

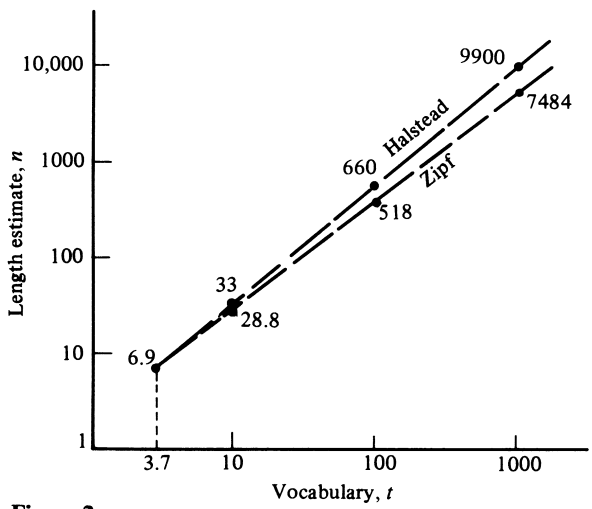

Figure 2.

In Fig. 2 we show that the lower and upper bounds are reasonably 'tight', at least for the range of values in t (the vocabulary size) typically encountered in practice. Here we refer to the programming applications wherein the tokens are understood in the compiler-theoretic sense, as representing the indivisible entities of a programming language: the operators, punctuation symbols, parentheses and delimiters and, most notably, the identifiers – names of variables, functions, arrays, etc. And it is in this latter sense that we are given a means for predicting the length of a program at an early design stage, i.e. before the program has been written. Note that in contrast to the studies of some other researchers, most notably Halstead himself,[2] we draw no particular distinction here between 'operators' and 'operands' in the programming language context. (Treating them all simply as 'tokens' has an obvious simplifying effect on the argument but perhaps misses the chance to observe that the operators are rather more limited in number, in general.)

In most program design methodologies[3] – Warnier–Orr diagrams, HIPO charts, pseudocode, etc., one is required to provide a rather complete listing and identification of all program variables, functions and data structures, as a means of documentation for the preliminary design phase. And in this context, a reasonably good estimate for t (the size of the program vocabulary) will be known well in advance of the detailed coding phase. This estimate of t can then be used to predict the program length n (using the bounds on length as given by Theorem 1) *assuming* that the composition of program text indeed follows a Zipf law distribution, regarding the frequency of occurrence of the various programming tokens. However, this is itself a matter of some debate, and we will have more to say of this assumption as we proceed.

3. HALSTEAD'S HYPOTHESIS

In somewhat more of a probabilistic vein[4] Halstead seeks to determine the 'expected value' of the program length n, using a rather naïve model of the programming process. Again a program is viewed as a sequence of tokens, but with the choice of any one of them considered as equally likely, from one point to the next. The program is completed when the last unused token is chosen for the first time. If we let

l_k = length of subprogram from $(k-1)$st to kth token occurrence

and

t = total number of distinct tokens (vocabulary size)

n = number of token occurrences (total length)

as before, then we may compute the probability

$$P(l_{k+1} = s) = (k/t)^{s-1}\left(1 - \frac{k}{t}\right)$$

as a geometric distribution of Bernoulli trials.[5] Summing over all possible values for s we obtain the expected value

$$E(l_{k+1}) = \sum_{s=1}^{\infty} s(k/t)^{s-1}\left(1 - \frac{k}{t}\right) = \frac{t}{t-k}$$

using a familiar identity

$$\sum_{1}^{\infty} ix^{i-1} = \frac{1}{(1-x)^2}$$

from the discrete calculus.[6] Finally, in noting that $n = \Sigma_{k-1}^{t} l_k$ we obtain

$$E(n) = \sum_{k-1}^{t} E(l_k) = t \sum_{k-1}^{t} \frac{1}{t-k+1} = t \sum_{r-1}^{t} \frac{1}{r}.$$

Since this is the same finite summation as that obtained in the earlier Zipf law analysis, we obtain the following pair of estimates.

Theorem 2

Assuming the Halstead hypothesis for program construction, we obtain the bounds *on the expected value* of the program length:

$$t(\gamma + \ln t) < E(n) < t \log_2 t$$

Example

If we throw an ordinary die until all six tokens finally appear, we may expect the number n of required throws to lie in the range

$$14.2 \approx 6(\gamma + \ln 6) < E(n) < 6 \log_2 6 \approx 15.5$$

say about 15 times as a round number.

Of course, it is one thing to say that a dice game follows the laws of chance. It is quite another to suggest that the programming task involves a stochastic process. This may not correspond to the actual practice. And yet it is certainly interesting to see that Halstead's rather simplistic hypothesis leads (essentially) to the same length estimates as those resulting from Zipf's law. The impression that somehow these two different theories lead to similar predictions of length has long been a part of the software engineering folklore,[7] though perhaps not so well documented previously. Even less apparent from the literature, however, is the difference in distribution of token occurrence in these two instances, that resulting from Halstead's probabilistic hypothesis as opposed to the Zipf hyperbolic distribution. This difference is quite significant and of immediate concern, by way of motivating the discussion to follow.

4. DIFFERENCE IN DISTRIBUTION OF TOKEN OCCURRENCES

Under Halstead's probabilistic hypothesis, it is fairly easy to see that the expected number of occurrences of the first occurring token is n/t. Using either of our estimates from the length, this number may be considered to be of the order of $\log t$ as a function of t alone. Owing to the variance of the distribution, we might expect the maximum number of occurrences of any token in a program string to be somewhat larger than this, but not nearly so large as $n_1 = t$, the value predicted by the Zipf theory. This point of view and the importance of this distinction have been outlined in a private communication to the author.[8]

To confirm this suspicion that the Halstead hypothesis leads to quite a different distribution from that of Zipf, even though the resulting length estimates are similar, one can perform the following simple experiment:

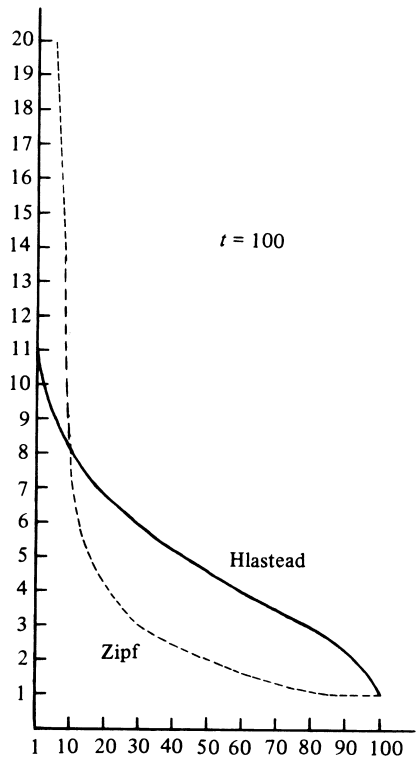

Figure 3.

generate random 'Halstead' token sequence trials
sort the results of each trial by token frequency
average the resulting rank frequencies over many trials

These averages should then give a fairly good indication of the Halstead distribution, representing the expectations of token occurrences for the various ranks. These could then be compared with the Zipf hyperbolic distribution, as suggested in Ref. 8.

In performing this experiment, the comparisons are indeed striking. We have shown the results in Fig. 3 for the case $t = 100$. Perhaps most noticeable is the difference in the initial value of n_r, as had already been anticipated, but also a corresponding difference in the initial value of the *magnitude of the slope*:

$$d_r = \left| \frac{dn_r}{dr} \right|.$$

[It is more convenient to work with these non-negative quantities in our subsequent discussions, noting that the slope itself is negative.] Thus we have the following:

$$\text{(A) } n_1 \ll t \quad \text{(B) } d_1 \ll t$$

in comparing the Halstead distribution with that predicted by the Zipf theory. [Note that $(dn_r/dr) = -t/r^2$ in differentiating Zipf's law, yielding $d_1 = t$.]

Empirical support for the Zipf–Halstead length formulas has been cited previously.[9,10] But we know of no study which would tend to support one of the two distribution theories as opposed to the other. In fact, it is quite likely that the real situation is *somewhere in between the two theories*. This is a point of view expressed in Ref. 8 and one that we plan to explore more fully as we proceed.

Example

Consider the PL/I program of Fig. 4, used as an illustration in Ref. 3. In figure 5 we tabulate the ranks r

```
FIBONACCI:  PROCEDURE;
            N=2;
            GET LIST (PREV_F, LAST_F, LIMIT);
            PUT LIST (PREV_F, LAST_F);
   REPEAT:  TEMP=LAST F;
            LAST_F=LAST_F+PREV_F;
            PREV_F=TEMP;
            PUT LIST (LAST_F)
            N=N+1
            IF N<LIMIT THEN GOTO REPEAT;
            END FIBONACCI;
```

Figure 4.

Rank r	Probability f_r	Symbol	Count n_r
1	0.2000	:	11
2	0.1091	LAST_F	6
3	0.0909	=	5
4	0.0727	N	4
5	0.0727	PREV_F	4
6	0.0545	.	3
7	0.0364	:	2
8	0.0364	PUT LIST	2
9	0.0364	+	2
10	0.0364	FIBONACCI	2
11	0.0364	LIMIT	2
12	0.0364	REPEAT	2
13	0.0364	TEMP	2
14	0.0182	PROCEDURE	1
15	0.0182	GET LIST	1
16	0.0182	IF THEN	1
17	0.0182	<	1
18	0.0182	go to	1
19	0.0182	end	1
20	0.0182	2	1
21	0.0182	1	1
Total			55

Figure 5

of the $t = 21$ tokens and the number n_r of their respective occurrences. Indeed, a plot of n_r vs R (the logarithms of n_r and r, respectively) would seem to suggest a Zipf-like relationship. [Note that in languages like PL/I or Pascal, the semicolon delimiter plays a role like that of an article ('a', 'an', 'the') in a natural language]. However, there are several observations that must be made here, particularly in the light of the analysis just completed.

Most importantly, the resulting length estimate, assuming a Zipf law distribution, yields the range:

$$76.06 = t(\gamma + \ln t) < n < t \log_2 t = 92.24,$$

in which n is supposed to lie. But $n = 55$ according to Fig. 5. Moreover, the Zipf estimate for $n_1 = t$ is too high (we have $n_1 = 11$, whereas $t = 21$). And the magnitude of the initial slope is likewise overestimated at $d_1 = t = 21$ by the Zipf theory, as we see by considering the initial data $n_1 = 11$, $n_2 = 6$, $n_3 = 5$, etc. Indeed, it would seem that the impression that 'real situation is somewhere in between the two theories – those of Zipf and Halstead' may be borne out here, and perhaps in many other examples as well.

5. CONCLUDING REMARKS

Certainly the Zipf law is of the ultimate simplicity. And its use in predicting program length (through the estimates of Theorem 1) has an undeniable appeal. But the question remains as to whether it is really the case that actual programs follow the strict Zipf law distribution, or whether instead some modification is in order, one where we would be able to move towards the direction of the 'middle grounded' between the theories of Zipf and Halstead.

In extending the observations of the preceding section, in particular those labelled (A) and (B), we propose that the following set of inequalities should be considered as a basis for exploring this middle ground:

(A) $1 \leqslant n_t < \ln t < n_1 \ll t$

(B) $0 \leqslant d_t < d_1 \ll t.$

Note, in reference to (A), that the value $n_r = \ln t$ is almost certain to be attained at some intermediate point by any predictive theory that we would be willing to consider. Since the constant function with this value again yields $n = t \ln t$ as a length estimate, and because n_r must of necessity be monotonic, it is reasonable to take $\ln t$ as an upper bound on n_t and a lower bound on n_1. Moreover, in the case of the latter we ensure that n_1 ranges upward from the Halstead prediction, as discussed previously.

It has been noted by several investigators[8,9] that 'tokens occurring a small number of times will be more common in real programs of significant size than will tokens occurring just once', as is predicted by both the Zipf and the Halstead theories. For this reason, we use the Zipf–Halstead $n_t = 1$ only as a lower bound. Similarly, in (B), we take the near-zero slope of n_r at the low occurrence end (as predicted by Zipf) as only a lower bound on d_t, recognising again the monotonicity of any predictive theory that we might consider.

A survey of twenty-six Pascal programs from Ref. 11 was conducted by a software engineering graduate class at the University of Denver in 1985. Partial results of this survey are listed in Fig. 6, and it is seen that the inequalities of (A) are satisfied in all but one instance. In all cases, the inequalities of (B) were satisfied as well. We note however that an intermediate value (corresponding to the $\ln t$ of (A)) has not been proposed for (B), and thus these latter results are perhaps less significant by comparison. In (B) an intermediate value of $(\ln^2 t/t)$ had been considered, but thus far, empirical results are less than completely supportive of such a position, particularly at the low-occurrence end of the distribution.

In conclusion, we would propose that the set of inequalities (A) and (B) might be used to guide future research in the development of more realistic software metrication theories. For example, one might consider generalized Zipf distributions of the form

$$n_r = \frac{Ct}{(r+A)^B}$$

where A, B, C are parameters used to provide an optimum fit to the observed data. One would then compute

$$n \approx Ct \sum_{r=1}^{t} \frac{1}{(r+A)^B}$$

as an approximation to the length, once a method had been devised for 'calibrating the model' so that realistic

n_t	ln t	n_1	t
1	3.43	9	31
1	3.30	6	27
1	3.26	5	26
1	3.00	5	20
1	3.47	10	32
1	3.53	8	34
1	3.53	11	34
1	3.61	11	37
1	3.64	11	38
1	3.04	6	21
1	3.40	10	30
1	3.37	13	29
1	3.18	7	24
1	3.26	8	28
1	3.43	18	31
1	3.26	9	26
1	3.09	5	22
1	3.18	7	24
1	3.58	15	36
1	3.40	7	30
1	4.39	88*	81
1	3.78	15	44
1	4.07	29	59
1	3.89	26	49
1	4.62	51	102

Figure 6

values for A, B and C could be determined. Note that if $A = 0$, $B = 1$ and $C = 1$ we obtain the pure Zipf law distribution. On the other hand, in the event that A, B and C satisfy the inequalities (*a*) $A > 0$, (*b*) $0 < B < 1$ and (*c*) $C < (1+A)^B$ we obtain

$$n_1 = \frac{Ct}{(1+A)^B} < t$$

$$d_1 = \frac{B}{1+A}\frac{Ct}{(1+A)^B} < \frac{B}{(1+A)}t < \frac{1}{1+A}t < t$$

after noting that

$$\frac{dn_r}{dr} = \frac{-BCt}{(r+A)^{B+1}}.$$

Thus (*a*), (*b*) and (*c*) would ensure that a part of our system of inequalities (A) and (B) would be satisfied.

Yet another approach with some promise is suggested by the observation that (A) and (B) may be interpreted as (a range of) boundary values for some differential equation. In noting that the pure Zipf law $rn_r = t$ is a solution to the differential equation

$$r\frac{dn_r}{dr} + n_r = 0$$

one might be led to consider the more general Cauchy–Euler differential equations[12] of which this is the simplest form. The inequalities (A) and (B) could then be used to suggest particular solutions $n_r = f(r, A, B, C, \ldots)$ that could be expected to provide a good empirical fit to the data.

Of course, it must be emphasised that these are only illustrations from among the many possible ideas that quickly come to mind. We do not intend to advocate any one of these competing theories over another, but only to provide a direction that could serve to focus such future explorations, generally. Finally, we include a necessary word of advice – that no new distribution theory be advanced as a serious candidate for study unless or until a substantial empirical backing compels its consideration.

Acknowledgement

The author would like to acknowledge the significant contribution of Professor Dan Johnston of the University of Queensland, Australia, particularly in reference to the analysis of Section 4.

REFERENCES

1. G. K. Zipf, The Psycho-biology of Language: *An Introduction to Dynamic Philology*. Houghton Mifflin, Boston (1935); MIT Press, Cambridge, Mass. (1965).
2. M. H. Halstead, *Elements of Software Science*. Elsevier/North-Holland, New York (1977).
3. M. L. Shooman, *Software Engineering*. McGraw-Hill, New York (1983).
4. M. H. Halstead and R. Bayer, Algorithmic dynamics. *Proc. ACM Conf.* (1973).
5. W. Feller, *An Introduction to Probability Theory and Its Applications*, 2nd ed., vol. 1. Wiley, New York (1957).
6. R. E. Prather, *Elements of Discrete Mathematics*. Houghton Mifflin, Boston (1986).
7. S. N. Mohanty, Models and measurements for quality assessment of software. *ACM Computing Surveys* **11** (1979).
8. D. Johnston, private communication (1986).
9. T. Sunohara, A. Takano, K. Vehara and T. Ohkawa, Program complexity measure for software development management. *5th Intl. Conf. on Software Engineering*, IEEE, New York (1981).
10. B. Curtis, S. B. Sheppard, P. Millman, M. A. Borst and T. Love, Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics. *IEEE Trans. on Software Engineering* **SE-5** (1979).
11. R. E. Prather, *Problem Solving Principles: Programming with Pascal*. Prentice-Hall, Englewood Cliffs, N.J (1981).
12. A. L. Rabenstein, *Elementary Differential Equations with Linear Algebra*, 3rd ed. Academic Press, New York (1982).