Res. Lett. Inf. Math. Sci., 2003, Vol.**5**, pp 17-22
Available online at http://iims.massey.ac.nz/research/letters/

Entropies Defined by Parsing Algorithms

B.I. Mills

Institute of Information & Mathematical Sciences
Massey University at Albany, Auckland, New Zealand.

b.i.mills@massey.ac.nz

Common deterministic measures of the information content of symbolic strings revolve around the resources used in describing or parsing the string. The well known and successful Lempel-Ziv parsing process is described briefly, and compared to the lessor known Titchener parsing process that might have certain theoretical advantages in the study of the nature of deterministic information in strings. Common to the two methods we find that the maximal complexity is asymptotic to $hn/\log n$, where h is a probabilistic entropy and n is the length of the string. By considering a generic parsing process that can be used to define string complexity, it is shown that this complexity bound appears as a consequence of the counting of unique words, rather than being a result specific to any particular parsing process.

---

1 Information, Entropy and Complexity

Various measures of the rate at which a sequence of symbols provides information have revolved around either the size of the smallest formal algorithm to generate the string, the logarithm of the improbability, the logarithm of the lexicon size, or the number of words generated during a string building process.

The intuitive concepts of Information, Entropy and Complexity were given a variety of precise formalisms during the twentieth century. At first the concept was one of the logarithm of a vocabulary, but there was some disagreement over the choice of vocabulary

Nyquist 1924: Intelligence rate is proportional to the logarithm of the number of symbol available to transmit at each point in time (1).

Hartley 1928: Information is the Logarithm of the lexicon size (2).

Shannon 1948: Entropy is the expected information per symbol, which in turn is the logarithm of the improbability of that symbol (3).

Solmonoff 1964: The Solmonoff probability with respect to a given Turing machine T is the probability that $T(X) = S$ given random X. This is not in general computable (7)

Kolmogorov 1965: The Kolmogorov complexity of a string S with respect to a given Turing machine T is the minimal length of an input string X, such that $T(X) = S$. This is not in general computable. (5). This definition has been explored in detail by Chaitin (6).

Wallace-Boulton 1968: The complexity of a string, measured with respect to a collection of possible

hypotheses, is the combined length of the optimal encoding of the hypothesis, and the optimal encoding of the string, given that hypothesis. (9) Similar work was independently completed by Risanen (10).

Lempel-Ziv 1976: The production complexity of a string is the number of words generated by repeatedly parsing the string for the first unknown word and adding that word. This is fairly easy to compute for individual strings. (8).

Titchener 1998: The Titchener complexity of a string is the sum of $\log(k+1)$ where k is the length of runs of symbols absorbed in the parsing process philosophically related to Lempel-Ziv parsing. This is fairly easy to compute numerically for individual strings, and algebraically for some classes of recursively defined strings (11).

2 Lempel-Ziv parsing

Given a string S, we build the string in the form $S_1S_2\cdots S_n = S$, such that each S_i is a word that occurs for the first time at that location. More formally, the vocabulary $v(S)$ of a string is the set of all substrings of S, that is, $W \in v(S)$ iff $S = AWB$. We say that S reproduces SQa, $S \rightarrow SQa$ iff $Qa \in v(SQ)$ and that S produces $SQab$, $S \Rightarrow SQab$ iff $S \rightarrow SQa$ An exhaustive production is when $S \Rightarrow R$ but $S \not\rightarrow R$. An exhaustive production sequence for S is a sequence $S_1 \Rightarrow S_1S_2 \Rightarrow \cdots \Rightarrow S$, such that each production, expect possibly the last, is exhaustive. Each string S has a unique exhaustive production sequence. The length of this exhaustive production sequence is the Lempel-Ziv complexity of S.

The Eigen vocabulary $e(S)$ helps to explain the manner in which complexity is generated, it is the set of substrings that appear for the first time at the end of the string. That is, $Q \in e(S)$ iff $S = AQ$ and $S \neq AQBb$. So $S \in e(S)$ and $S = AaW$ and $W \in e(S)$ implies $aW \in e(S)$. So S can be split into $S = AxB$, such that $S = AxWX$ implies $X \notin e(S)$ and $S = WxXB$ implies $XB \in e(S)$. The length of the string Ax is the size of the eigen vocabulary, and represents the innovation in S.

For an example, we start with $S = 101101011$ and parse to the right, marking when the extension occurs for the first time.

$$
\begin{array}{l}
101101011 \\
1|01101011 \\
1|0|1101011 \\
1|0|1-101011 \\
1|0|11|01011 \\
1|0|11|0-1011 \\
1|0|11|01-011\ * \\
1|0|11|010|11
\end{array}
$$

Note: In the step marked *, 01 is in the string at location 2, even though it is not marked out. Lempel-Ziv parsing does not restrict the vocabulary considered to those words actually marked, but includes all words in the string parsed so far. At each stage we have a collection of *words already seen* (initially empty) which must be avoided in the building process. As the string is built this lexicon increases. Alternatively we can say that we start with an infinite lexicon of acceptable words, from which we delete words as they occur.

The complexity of S is the number of words parsed exhaustively, in this case 5. In 1976, Lempel-Ziv (8) showed that this complexity measure is bounded by

$$\frac{hn}{\log n}$$

where n is the length of S and h is the Shannon entropy of the source

3 Titchener Parsing

Titchener parsing gives a complexity that is reminiscent of Lempel-Ziv complexity, but introduces a couple of other aspects. Call a string of the form ax^n an initiated run. Titchener parsing builds the string up by extending it with a run of words from a lexicon L_i, that is built during parsing. The initial lexicon is the alphabet, and is increased at each production. The productions are always of the form $S \to SX^k$. The new lexicon, L_i, includes words of the form, WX^m for each word $W \in L_{i-1}$. However, we look for innovation in the runs, and must not use the same word again, so if we come upon a run we must absorb all of it at once. The added complexity is $\log_2(k+1)$, which amounts to a count of new words if no non trivial runs are encountered.

For example we start with the string $x10110101$, where the initial x has be placed there for emphasis, since the first character is essentially ignored.

$x101101011$
$x1|01101011$
$x1|01|101011$
$x1|01|101|011$
$x1|01|101|01|1$

It needs to be noted that there are several variants of Titchener parsing in the literature. I have selected here the one that seems to be most simply illustrative of the required characteristics.

4 Comparison of Titchener and Lempel-Ziv

- Both build the string by extension.
- Both look for innovation.
- If there are no non trivial runs then both just count the words.
- Lempel-Ziv reduces, but Titchener increases the lexicon.
- The complexity of each is bounded by $\frac{hn}{\log n}$

What is the origin of this formula?

5 Tokenisation Complexity

Given a finite string S there are a finite number of ways to parse $S = S_1 S_2 \cdots S_n$, into substrings of S, where none of the S_i are empty. The shortest is for $n = 1$, with $S_1 = S$, the longest, with $n = \texttt{length}(S)$ is where each S_i is just an individual symbol from S.

We define a complexity measure: the greatest number of distinct words into which the string can be parsed. This measure asks for the maximal size of $\{S_i : i \in 1..n\}$ over all possible parsings of S.

For example:

$$111111111 \;=\; 1\ 11\ 111\ 111$$

Because this is a run of 1s, the only substrings are also runs of 1s. To get as many as possible we need to use the smallest we can. Starting with 1s. `1`, `11`, `111`, we find we can fit no more, so the complexity is 3.

$$101101011 \;=\; 1\ 0\ 11\ 01\ 011$$

Given the above decomposition, the complexity is clearly at least 5, that it is actually 5 is apparent on considering how to make 6 distinct symbols as short as possible, 0 1 00 01 10 11, which has a total length of 10, while the above string has only length 9.

5.1 Maximal tokenisation complexity

If we split a string into distinct words, how many can we get? If the words are long, not many will fit into the string, but if the words are short, we can't get more than A^m distinct ones, where A is the alphabet size, and m the length of the words. The maximum number of words comes when the complete collection just fits into the string, that is $mA^m = n$. And the actual count is $A^m = n/m$.

The condition $mA^m = n$ defines m as a function of n. It is not possible to get a simple closed form for this function, but we can ask what is the asymptotic behaviour of m as a function of n?

Firstly we note that, mA^m is monotonic strictly increasing at least for $m, A > 1$ (which is a natural constraint in this context). Also, $(\log_A n)A^{\log_A n} = (\log_A n)n > n$, so if $mA^m = n$, then $m < \log_A n$ and $\log_A m < \log_A \log_A n$.

Thus,

$$0 \leq \lim_{n\to\infty} \frac{\log_A m}{\log_A n} \leq \lim_{n\to\infty} \frac{\log_A \log_A n}{\log_A n} = 0$$

$$\lim_{n\to\infty} \frac{\log_A m}{\log_A n} = 0$$

From $mA^m = n$ we get $m = \log_A n - \log_A m$.

so

$$\frac{m}{\log_A n} = 1 - \frac{\log_A m}{\log_A n}$$

so

$$\lim_{n\to\infty} \frac{m}{\log_A n} = \lim_{n\to\infty} 1 - \lim_{n\to\infty} \frac{\log_A m}{\log_A n}$$

so

$$\lim_{n\to\infty} \frac{m}{\log_A n} = \lim_{n\to\infty} 1$$

so

$$A^m = \frac{n}{m} \sim \frac{n}{\log_A n}$$

So the asymptotic bound on the number of distinct words is

$$(\log_2 A)\frac{n}{\log_2 n}$$

But, A^m was simply the number of distinctions that m symbols could make.

That is, $\log_2(A^m) = m(\log_2 A)$ is the information in m symbols,

so $\log_2 A$ is the entropy.

It must be noted that Titchener parsing counts $\log(k+1)$ for a run of length k, but the number of distinct words into which a run can be split of order $\sqrt(k)$, which is greater than $\log(k+1)$ for sufficiently large k, and so the use of the logarithm will not introduce any higher asymptotic behaviour (although it can produce higher transient behaviour) So, the maximal complexity for Titchener and Lempel-Ziv parsing, as well as any similar parsing algorithm, is asymptotically bounded by

$$\frac{hn}{\log n}$$

Where h is the Shannon entropy of an ergodic source,

and n is the length of the string.

6 Specific cases of Titchener parsing

The recursive nature of Titchener parsing can make complexity easy to compute. The author is not aware of any similar effect for Lempel-Ziv parsing, and due to the overlapping nature of the strings used in the extensions it seems difficult to achieve. The potentially algebraic nature of Titchener parsing is demonstrated below on a couple of cases.

6.1 Hanoi

Let $\texttt{hanoi}(0) = \epsilon$, $\texttt{hanoi}(n) = \texttt{hanoi}(n-1)n\texttt{hanoi}(n-1)$ where ϵ represents the empty string, where $S\epsilon = \epsilon S = S$. This is the well known sequence of moves required to solve Towers of Hanoi puzzle. It is derived from the recognition that to shift n disks we may shift $n-1$ disks, then shift disk n, and then shift the $n-1$ disks again.

So

$$\begin{aligned}\texttt{hanoi}(1) &= \epsilon 1 \epsilon = 1\\ \texttt{hanoi}(2) &= 121\\ \texttt{hanoi}(3) &= 1213121\\ \texttt{hanoi}(4) &= 1213121 4 1213121\end{aligned}$$

Direct computation shows

$$\texttt{reduce}(\texttt{hanoi}(4)) = [12]13[12]14[12]13[12]1$$

$$\texttt{reduce}^2(\texttt{hanoi}(4)) = [121]3[121]4[121]3[121]$$

Substituting $121 \to 1$, $3 \to 2$, $4 \to 3$, into $\texttt{reduce}^2(\texttt{hanoi}(4))$ we get $1213121 = \texttt{hanoi}(3)$. Thus $\texttt{reduce}^2(\texttt{hanoi}(4)) \equiv \texttt{hanoi}(3)$ More generally, it is apparent $\texttt{reduce}^2(\texttt{hanoi}(n)) \equiv \texttt{hanoi}(n-1)$. Intuitively the effect is that if we glue the top two disks of a Hanoi puzzle together, then we have the equivalent of the Hanoi puzzle with one less disk.

The complexity measure of the Hanoi strings thus satisfy the recurrence relation, $\texttt{measure}(\texttt{hanoi}(n)) = 2 + \texttt{measure}(\texttt{hanoi}(n-1))$ with initial condition, $\texttt{measure}(\texttt{hanoi}(1)) = 0$, and so $\texttt{measure}(\texttt{hanoi}(n)) = 2(n-1)$. In a similar manner the length of the Hanoi strings satisfy the recurrence relation $\texttt{length}(\texttt{hanoi}(n)) = 2\texttt{length}(\texttt{hanoi}(n-1)) + 1$, with initial condition, $\texttt{length}(\texttt{hanoi}(1)) = 1$, and so $\texttt{length}(\texttt{hanoi}(n)) = 2^n - 1$. Thus, we have $C_n = 2(n-1)$, and $L_n = 2^n - 1$ Thus, $C_n = 2(\log_2(L_n + 1) - 1)$. Asymptotically this is: $C_n \sim 2 \log_2 L_n$

6.2 The Liouville example

Based on the Liouville number (which uses a factorial number of zeros) we can define, $\texttt{liou}(0) = x1$ and $\texttt{liou}(n+1) = \texttt{liou}(n)0^{n+1}1$. The first few such strings are as follows:

$$\begin{aligned}\texttt{liou}(0) &= x1\\ \texttt{liou}(1) &= x101\\ \texttt{liou}(2) &= x101001\\ \texttt{liou}(3) &= x1010010001\\ \texttt{liou}(4) &= x101001000100001\end{aligned}$$

In general, $\texttt{liou}(n)$ is $x1$ followed in sequence by all the runs of 0s, up to n in length, each run terminated by 1. Direct computation, as in the previous example, shows that:

$$\texttt{reduce}(\texttt{liou}(4)) \;=\; [x1][01]0[01]00[01]000[01]$$

$$\equiv\; x1010010001 \;=\; \texttt{liou}(3)$$

Along these lines, it is apparent in general that $\texttt{reduce}(\texttt{liou}(n)) = \texttt{liou}(n-1)$. So the measure satisfies the recurrence, $\texttt{measure}(\texttt{liou}(n)) = \texttt{measure}(\texttt{liou}(n-1)) + 1$, with boundary condition $\texttt{measure}(\texttt{liou}(1)) = 1$ and so $\texttt{measure}(\texttt{liou}(n)) = n$. And, the length can be computed directly as 1 occurrence of x, $n+1$ occurrences of 1, and the zeros counted as $1 + 2 + 3 + \cdots + n$, is $\texttt{length}(\texttt{liou}(n)) = 1 + n + 1 + \sum_{i=1}^{n} i$, so $\texttt{length}(\texttt{liou}(n)) = 2 + n + \frac{n(n+1)}{2} = \frac{1}{2}(n^2 + 3n + 4)$ Thus we have, $C_n \sim n$, $L_n \sim \frac{1}{2}n^2$ and so $C_n \sim \sqrt{(2L_n)}$.

7 Parsing Entropy Precis

The parsing complexity of a string is a direct measure of the resource usage by an algorithm parsing that string into a sequence of (mostly) unique tokens. Typically the maximal complexity is asymptotically proportional to $n/(\log n)$. Thus we define the parsing information of the string as the complexity divided by $\log n$, and the entropy as the information divided by the length. Under these conditions, for maximal entropy, under a wide range of parsing algorithms, the parsing entropy is equal to the Shannon entropy. This observation can be reverse to suggest that if a parsing algorithm does not behave in this manner, then the parsing algorithm must be missing patterns in the string, or in some other way not reflecting the true complexity of the string. Due to the asymptotic equivalence of the results of various parsing algorithms, we may consider this entropy to be a property of the string rather than the specific parsing algorithm used. It is suggested that Titchener parsing is particularly compatible with recursive string definition, and can be used to determine, by examination of the generating algorithm, of the entropy of classes of strings.

References

[1] H.Nyquist [1924] Certain factors affecting telegraph speed Bell Labs Technical Journal 1924 p324

[2] R.V.L Hartley, [1928] Transmission of information Bell Systems Technical Journal Vol 7 pp 535-563, (July 1928)

[3] Claude E Shannon [1948] A mathematical Theory of Communication Bell Systems Technical Journal vol27 pp379-423, 623-656.

[4] P Martin-Lof [1966] The definition of random sequences Inform Contr vol 9 pp602-619,

[5] Kolmogorov A.N. [1965] Three approaches to the quantitative definition of information. Problems in Information Transmission, vol 1, pp4-7.

[6] Chaiten G.J. [1966] On the lengths of programs for computing binary sequences. Journal of Computing Machinery vol 13 pp547-569

[7] Solmonoff R.J. [1964] A formal theory of inductive inference. I,II Information and Control, vol 7, pp1-22, 224-254

[8] Lempel A and Ziv J [1976] On the complexity of finite sequences IEEE Transactions on Information Theory Vol IT22, No 1, Jan 1976 pp75-81

[9] Wallace C.S and Boulton D.M [1968] An information measure for classification Journal of Computing vol13 pp63-69

[10] Rissanen J.J [1978] Modelling by shortest data description Automatics vol14 pp465-471

[11] M.R.Titchener [1998] A deterministic Theory of Complexity, Information and Entropy IEEE Information Theory Workshop, Feb 1998 (San Diego).