1989-04-01

THE ZERO FREQUENCY PROBLEM: ESTIMATING THE PROBABILITIES OF NOVEL EVENTS IN ADAPTIVE TEXT COMPRESSION

Witten, Ian H.; Bell, Timothy C.

Introduction

In data compression, the encoder and decoder share a *model* of the source of data. The former encodes incoming characters with respect to that model, while the latter decodes the resulting bit-stream using the same model. *Adaptive* methods of compression are those that adjust the model dynamically, moulding it to the characteristics of the message. When adaptive encoding is done in a single pass through the message, the model is formed on the basis of the *preceding* portion of the message only, and changes constantly as the transmission proceeds. The encoder reads the next token, encodes it with respect to the current model, and updates the model accordingly. The decoder receives the corresponding bit string, decodes it to recover the token, and updates its model accordingly. Assuming error-free transmission the models will always agree, even though they are never transmitted explicitly.

This paper assumes that the model is a statistical one, estimating probabilities from the observed message and using these to encode tokens for transmission. The coding problem is solved by the method of "arithmetic coding," discovered in 1976 by Pasco (1976) and Rissanen (1976) and developed into a practical technique by Rubin (1979), Guazzo (1980), and Rissanen & Langdon (1979). Arithmetic coding provides a way of coding tokens with respect to a probability distribution in an optimal way; Witten *et al* (1987) give an accessible implementation of the technique. The most prominent alternative to statistical coding is adaptive dictionary coding, well exemplified by the popular Ziv-Lempel method (Ziv & Lempel, 1977, 1978) of which a number of variants have appeared (eg Welch, 1984); a survey and comparison appears in Bell & Witten (1987). While adaptive dictionary methods appear to lend themselves to more efficient implementations (eg Thomas *et al*, 1985), statistical techniques seem to have the edge in compression performance (Bell *et al*, in press)—although the new, carefully optimized, Ziv-Lempel variants of Fiala & Greene (1989) present a significant challenge. At any rate, the problem addressed in this paper concerns statistical coding only.

Adaptive statistical coding involves estimating probabilities of tokens in appropriate contexts and using these probabilities for compression. Tokens may be characters, words, n-grams, or other units. Their probabilities are estimated by their relative frequencies, measured from some sample—which is typically all the text seen so far. The model may estimate the probabilities in isolation (no context), in the context of the previous token (first-order model), the previous two tokens (second-order model), or larger contexts—perhaps even of varying lengths. The problem we address in this paper transcends these details of token and context. It concerns the fact that there will (almost) always be situations in which the encoder encounters a token in a context where it has never been seen before. In order to encode and transmit such a token, it must be given a non-zero prediction probability—despite its frequency being zero. And yet, by definition, there is no evidence on which to base this probability. This is the "zero-frequency problem" (Roberts, 1982).

In previous work on adaptive text compression, unprincipled, *ad hoc* approaches have been taken to the zero-frequency problem. In fact, as mathematicians and philosophers have frequently noted (eg Good, 1965; Pierce 1956), in the absence of *a priori* knowledge there seems to be no theoretical basis for choosing one solution over another. Following an introduction to the zero-frequency problem in text compression by way of two examples in the next section, we summarize the methods that have been used in practice. We then present a Poisson process model for the appearance of new tokens, which was originally introduced by Good & Toulmin (1956) to estimate the number of unseen biological species, and later applied by Efron & Thisted (1976) to model the appearance of new words in natural language text. This model is evaluated and compared with the other methods in terms of its ability to predict novel characters, words, and n-grams in text. It is also tested in an actual statistical compression program.

Two examples of the zero-frequency problem

Adaptive word coding. Imagine a coding scheme in which the encoder reads the next word of text, looks it up in a list, and transmits the index in the list in place of the word itself (Bentley *et al,* 1986). Sending numbers between 0 and $r-1$ (where r is the size of the list) instead of words offers a possible saving in bandwidth. If the next word is not present in the list, a special code—say the number r—must be transmitted to signal the fact, followed by the word itself, perhaps as a sequence of ASCII characters. The new word could then be added to both the encoder and decoder's lists in case it appears again. If both lists were empty initially, by the time the message had been transmitted they would record all of the words in it.

The code that is used when the next word is not in the list is called an *escape* code, since it signals an escape to a different scheme—character instead of word coding. In this example, a fraction $1/(r+1)$ of the code space is reserved for the escape code (assuming that numbers are transmitted as fixed-length binary strings). In general, however, words need not be treated equally. For instance, Bentley *et al* (1986) discuss how the word list can be maintained in most-recently-used order, with variable-length coding to give words near the front of the list shorter codes. This decreases the fraction of the code space devoted to the escape code, increasing its coded length. Alternatively, frequencies can be associated with each word in the list, and as words occur in the message they can be coded with respect to the current frequency distribution using the arithmetic coding method (Witten *et al,* 1987). Then the code space allocated to a word reflects more accurately the likelihood of that word. However, we must decide what probability to associate with the escape code, the "escape probability," by estimating the likelihood of a novel word occurring. This is the zero-frequency problem.

The zero-frequency issue only concerns *adaptive* modeling. For example, if in the word-coding example the encoder made an initial pass over the message to accumulate a dictionary and transmitted it first, then the question of escapes would not arise. In many situations, one-pass or "real-time" coding is necessary and the zero-frequency issue must be faced.

Predicting characters in context: the PPM method. One of the most effective statistical compression methods, the "prediction by partial match" or PPM technique, makes extensive use of escape codes (Cleary & Witten, 1984b). The basic idea is to predict each character on the basis of the last few characters of context. For some value of k (generally between 3 and 10), the frequency of each $(k+1)$-tuple is recorded. This is tantamount to recording all occurrences of each context of length k, and counting the frequency of characters in that context. These frequencies are maintained adaptively as the encoder proceeds. For prediction, all recorded $(k+1)$-tuples are retrieved whose first k characters match the current k characters of context, and used to form a probability distribution for the upcoming character. This is referred to as an *order-k fixed-context model.*

The snag, of course, is the zero-frequency problem: what to do if the distribution assigns zero probability to the upcoming character. To overcome this, PPM also stores all shorter fixed-context models with context length o, for each value of o between 0 and k. If the upcoming character is not predicted by the order-k model, an escape code is transmitted and the order $k-1$ model is used instead. If the character is not predicted by this model either, another escape is sent and the order $k-2$ model is used. It is possible that the encoder will escape all the way down to the order-0 model, which records the frequencies of the individual characters. Indeed, if the character has not been seen before at all, the encoder will escape again, at which point the ASCII code of the character is transmitted.

The PPM method can encode formatted text in as little as 2.2 bit/character, and the results in Table 4 below indicate that it averages under 2.5 bit/character on a variety of different unformatted text files, including programs. Fuller details of the technique, including an optimized version developed by Moffat (1988), can be found in Bell *et al* (in press).

Methods for computing escape probabilities

This section reviews approaches that have been taken to the zero-frequency problem. Laplace's law of succession (described below) addresses the issue, but only in the context of a finite alphabet of known size. In many situations, for example word coding, the alphabet is indeterminate. In others, for example coding characters in the context of a few preceding ones, the alphabet for any particular context is bounded (by the total number of characters) but the bound is very loose. For example, very few letters can follow "q" in English. Using Laplace's method with a loose bound incurs needless wastage of code space at every prediction. Consequently after introducing Laplace's law (for a two-symbol alphabet) and its generalization to a q-symbol alphabet, we turn to the *ad hoc* methods that have actually been used in practical adaptive compression schemes.

Laplace's law of succession. The first quantitative model for a zero-frequency phenomenon was developed by Laplace in 1812. From an urn known to contain a mixture of N black and white balls, suppose n are drawn and c of them found to be black. What is the best estimate for the total number of black balls, and for the probability that the next ball drawn will be black? The maximum likelihood estimate for this probability is c/n, and for large n and c there is hardly a problem—the proportion of black balls will closely approximate this value. For small c and n, however, and most particularly if c happens to be 0 or n, the estimate c/n is suspect. It would seem risky to conclude from the fact that the first couple of balls drawn are black that the urn contains only black balls.

Changing from the urn terminology, where balls are black or white, to one involving trials which succeed or fail, Laplace's law of succession estimates the probability that the next trial will be a success by

$$\hat{p} = \frac{c+1}{n+2} .$$

This can be derived from Bayes theorem in the form

$$Pr[H_j|E] = \frac{Pr[H_j]\,Pr[E|H_j]}{\sum_{i=1}^{N} Pr[H_i]\,Pr[E|H_i]}$$

where E is the evidence that c successes have occurred in the first n trials; and H_j $(j=0,1,\cdots,N)$ is the hypothesis that there will be a total of j successes in all N trials (which include the n observations). We assume a uniform prior distribution for the probability of success, in other words the $N+1$ hypotheses are equally likely *a priori:*

$$Pr[H_j] = \frac{1}{N+1} \quad (j=0,1,\cdots,N).$$

From this assumption the law of succession can be obtained (the derivation can be found in standard texts, for example Johnson & Katz, 1977). A common next step is to seek greater flexibility by assuming that the prior is the the two-parameter beta distribution (Good, 1965); but this does not concern us here.

The law of succession can be applied to the coding of a binary source as follows. Suppose $n=c_1+c_2$ tokens have been encountered so far, of which c_1 are zeros and c_2 are ones. Then the prediction probabilities are estimated as

$$Pr[\text{next token is zero}] = \frac{c_1+1}{n+2} \qquad Pr[\text{next token is one}] = \frac{c_2+1}{n+2}\ .$$

In the event that only zeros have been seen so far, so that $c_1=n$, the escape probability reserved for a novel event is $1/(n+2)$.

The generalized law of succession. The law of succession can be generalized to dispense with the restriction to binary sources. Suppose that instead of two event types (zero and one, success and failure, or black and white balls) there are q of them. Out of the complete set of N observations, suppose there are C_1 of type 1, C_2 of type 2, and so on, where these random variables are related by

$$C_1 + C_2 + \cdots + C_q = N.$$

It is clear that the prior probability distribution of the random variables cannot be uniform, since the probabilities of the types interact in that they must all sum to 1. For example, if three of the prior probabilities exceeded ¼, none of the others could exceed ¼. Consequently a different assumption is needed.

Instead, assume that all compositions—that is, all sets of values of C_1, C_2, $\cdots$, C_q—are equally likely. Now if in our sample of n tokens we find c_1 of type 1, c_2 of type 2, $\cdots$, c_q of type q, where $n=\sum c_i$, it can be shown (eg Jeffreys, 1939, p.117ff) that

$$Pr[\text{next token will be of the } i\text{th type}] = \frac{c_i+1}{n+q}\ .$$

This can be applied to coding providing the size q of the alphabet is finite and known in advance. If at any given point a total of r $(r \leq q)$ different token types have been seen, so that $q-r$ of the counts c_i are zero, the probability that the next token will be novel is $(q-r)/(n+q)$, and this escape probability is divided evenly amongst all unseen tokens. In effect the method adds 1 to all counts, treating each symbol i as if it had been counted c_i+1 times to give a total of $n+q$. If i has not been seen so far it is allocated a count of 1. Cleary & Witten (1984a) investigated this model theoretically and placed some bounds on its performance in terms of the overhead it imposes on coding.

Method A. Laplace's law of succession applies to two-symbol alphabets, and its generalization applies to q-symbol alphabets for some finite, known, q. When the alphabet has unknown size, other methods must be employed. The best strategy is to allocate part of the code space for the "escape" event and, when a novel symbol appears, transmit it using some other coding method. Method A uses

$$Pr\,[\text{next event will be novel}] \;=\; \frac{1}{n+1}\;,$$

where n is the number of tokens seen so far. The prediction probability for a token i that has been seen c_i times so far is

$$Pr\,[\text{next token will be of the } i\text{th type}] \;=\; \frac{c_i}{n+1}\;.$$

(As n increases this approaches the maximum likelihood estimate c_i/n). In effect, one count is allocated to the possibility that a previously unseen symbol will occur next. This method was investigated theoretically in the finite-alphabet case by Cleary & Witten (1984a), who found that although a stricter bound could be placed on the generalized-law-of-succession method, method A tended to perform better than it in practice (although artificial examples could be found where it gave worse results).

Method B. This method, introduced by Cleary & Witten (1984b), classes a symbol in a particular context as novel unless it has already occurred *twice*. This is motivated by the consideration that a once-off event may be an error or other anomaly, whereas an event that has occurred twice or more is likely to be repeated further. The probability of an event that has occurred more than once in the context is then estimated to be

$$Pr\,[\text{next token will be of the } i\text{th type}] \;=\; \frac{c_i-1}{n} \quad \text{when } c_i>1.$$

The escape probability is

$$Pr\,[\text{next event will be novel}] \;=\; \frac{r}{n}\;,$$

r being the number of types which have occurred more than once so far.

Method C. This method was introduced by Moffat (1988) as a compromise between A and B. He found it desirable to increase the probability of novel events as more types are observed, but wasteful to wait for something to occur twice before using it. Consequently he used event probabilities of

$$Pr\,[\text{next token will be of the } i\text{th type}] \;=\; \frac{c_i}{n+r}\ ,$$

leaving an escape probability of

$$Pr\,[\text{next event will be novel}] \;=\; \frac{r}{n+r}\ .$$

This can be rationalized by imagining "novelty" as an event in its own right, which has of necessity occurred r times if that is the number of different tokens seen so far. Then maximum likelihood estimates are used for the token probabilities and novelty probability.

Poisson process model for the appearance of new tokens

A different approach to modeling the escape probability is to consider the appearance of each token as a separate Poisson process. This idea was pioneered by Fisher's (1943) work on estimating the number of unseen species in ecological studies. Given the number of species, and the number of individual butterflies in each species, captured in one day on a desert island, how many different species might one expect there to be all told? Or, given that Shakespeare's complete works of 885,000 words include 31,500 different ones, of which 14,400 appear only once, 4,300 twice, etc, how many words did he know? The latter question was studied by Efron & Thisted (1976), and the analysis below follows their exposition.

The Poisson process model. Suppose there is an unknown number q of different types, and in a sample of n tokens we find c_i tokens of type i $(1 \le i \le q)$, where $n = \sum c_i$. Not all types are manifested in the sample, of course; that is the zero-frequency problem. The basic assumption is that tokens of type i appear according to a Poisson process with an expectation λ_i of occurring in a sample of size N; in other words, c_i is a sample from a Poisson distribution with mean λ_i $(i=1, \cdots, q)$. We do *not* need to assume that the q individual Poisson processes are independent of each other.

The problem is to extrapolate from the counts in the n-token sample to those that might be expected in a larger corpus, say one having a total of $N = (1+\theta)n$ tokens. Let C_i be the number of times that the token of type i appears in the larger corpus, where $N = \sum C_i$. The Poisson process assumption implies that

- C_i has a Poisson distribution with mean $(1+\theta)\lambda_i$;
- the sample of size n is typical of the larger sample of size N.

The second condition can be characterized more precisely by the fact that, given the value of C_i, c_i will be binomially distributed with C_i trials and parameter $1/(1+\theta)$.

Analysis of the model. The analysis rests on supposing that $G(\lambda)$ is the empirical cumulative distribution function of the numbers $\lambda_1, \cdots, \lambda_q$. We make no assumptions about the form of G; just that it exists. From our observations we can find the number of types observed exactly r times in the sample of size n, say t_r, for r=1, 2, $\cdots$. These are random variables with expected values

$$\tau_r = E(t_r) = q \int_0^\infty \frac{e^{-\lambda}\lambda^r}{r!} \, dG(\lambda),$$

since the integrand is just the probability that a particular type (with parameter λ) appears exactly r times in the sample. Using this type-token data, our goal is to extrapolate to the larger, N-token, sample and estimate the number of new types that will appear in it. We can calculate the expected number of new types by

$$q \int_0^\infty e^{-\lambda} \, (1 - e^{-\lambda\theta}) \, dG(\lambda),$$

which integrates over λ the probability that a particular type (with parameter λ) does not appear in the original sample (probability $e^{-\lambda}$) but does appear in the extended sample (probability $1 - e^{-\lambda\theta}$). By substituting the expansion

$$1 - e^{-\lambda\theta} = \lambda\theta - \frac{\lambda^2\theta^2}{2!} + \frac{\lambda^3\theta^3}{3!} - \cdots$$

into this expression and using the formula for τ_r above, we obtain as the expected number of new types

$$\tau_1\theta - \tau_2\theta^2 + \tau_3\theta^3 - \cdots .$$

This remarkable result first appeared in Good & Toulmin (1956). It suggests estimating the number of new types by substituting the actual type-token data t_1, t_2, $\cdots$ from the n-token sample for their expected values τ_1, τ_2, $\cdots$:

$$t_1\theta - t_2\theta^2 + t_3\theta^3 - \cdots . \qquad (1)$$

While this technique was applied to the analysis of Shakespeare's vocabulary out of curiosity by Efron and Thisted over a decade ago, the work has recently found practical use. Kolata (1986) relates the fascinating story of its application to a previously unknown poem, suspected to have been penned by Shakespeare, which was discovered in November 1985 in the Bodleian library in Oxford, England.

Application to the zero-frequency problem: methods P and X. In the special case where the larger sample contains just one more token than the smaller one, N=n+1, the expected number of new tokens becomes the probability that the next token is new:

$$Pr\,[\text{next event will be novel}] \;=\; t_1.\frac{1}{n} - t_2.\frac{1}{n^2} + t_3.\frac{1}{n^3} - \cdots .$$

We call this method P. Retaining the first term only of the series gives an excellent approximation, dubbed method X:

$$Pr\,[\text{next event will be novel}] \;=\; \frac{t_1}{n} .$$

Recall that t_1 is the number of types that have occurred once only—in other words, the number of *hapax legomena*—in the n-token sample. Both of these methods will break down if this becomes zero, method P because the novel-event probability will almost certainly turn out to be negative, and method X because the probability is zero, making it impossible to encode a novel event if one does occur. Moreover, the methods break down if $t_1=n$ because the novel-event probability becomes 1, and this will certainly be the case after just one token has been encoded. Special precautions must be taken to avoid these situations, although they are not as serious, or as frequent, as the original zero-frequency problem.

Empirical comparison of zero-frequency methods

These methods of estimating the probability of novel events were compared empirically. For reference, Table 1 summarizes the various models for the escape probability. The principal source of text used for comparison was Thomas Hardy's 136,000-word novel *Far from the madding crowd* (book 1). Several different kinds of tokens were used, and Table 2 shows the number of types and tokens for characters, 2-grams, 3-grams, 4-grams, 5-grams, and words.

Predicting novel words. A ''novel-word event'' occurs when a previously unseen word is encountered for the first time. For instance, the word ''love-sick'' does not appear until about half-way through Hardy's book, word number 68,055 to be precise. The probability of this occurrence can be calculated according to each of the zero-frequency methods detailed above. According to method A, it is 1/(68055+1) = 0.000015. The following word, number 68,056, happens to be ''man'' which is not novel. The probability of this, again according to method A, is 68056/(68056+1) = 0.999985. Note that we are not estimating the probability of the word being ''man,'' only of it being novel. Of course, the other models will give different probabilities for these events.

The entropy of the novel-word event is plotted against position in the book in Figure 1. Entropies are calculated as $-\log_2 p$ bits, where p is the probability of the event; thus the two words above contribute $-\log_2 0.000015=16.05$ bits and $-\log_2 0.999985=0.000021$ bits to the entropy respectively. The entropy is averaged in 10,000-word chunks (words 1 to 10,000; words 10,001 to 20,000, etc) and plotted against word number. Notice that the averaging is not cumulative, so that the last point of the graph represents performance over the last full block of 10,000 words, not over the whole book.

As Figure 1(a) shows, method A is substantially inferior to the other methods at predicting the novel-word event. For the first half of the book it gives an entropy of over 1 bit/word—a coin toss would be a better predictor of whether a word is novel or not! In general, the novel-event probability assigned by method A is far too low.

The results for the other methods are shown in Figure 1(b) with an amplified vertical scale. None of them work very well at the beginning of the book, though they improve on the 1 bit/word offered by random coin tossing. By the end they are performing at around 0.25 bit/word. Since all graphs follow the same shape very closely, the undulations seem to represent identifiable features of the text itself rather than random fluctuations. The bottom line corresponds to methods P and X, which are indistinguishable at this scale (in fact they are identical to several significant figures when the number of words exceeds 100). It is satisfying to find the more principled methods out-performing the *ad hoc* ones.

Figure 2 gives the result of the same experiment on a different book, *Principles of computer speech* (book 2) by one of the present authors (IHW). As far as relative performance of the methods is concerned, it shows the same picture. Method A is very substantially inferior to the others. Methods P and X (which are again virtually identical) out-perform A, B, and C. However, while the overall decreasing trend in the first half of the graph is the same as that for Hardy's book, the shape of the curves in the second half is markedly different.

Evaluation of the Poisson model for words. A more direct evaluation of the Poisson process model is to count the number of new words that appear in different stretches of the text, and compare the figure with the number predicted by the model. This may help explain the overall shapes of the curves in Figures 1 and 2, as well as providing an assessment of how well the model is doing.

On the basis of the type-token data exhibited in the first 50,000 words, equation (1) was used to estimate the number of new words that would appear in a 10,000 word stretch of text (θ=0.2). The predicted figures were 798.0 and 455.4 for books 1 and 2 respectively. Table 3 shows, for each 10,000-word stretch after that point, the actual number of new words that did not appear in the initial 50,000 words.

The average number of new words observed in book 1 is quite close to the predicted average, which falls well within the 95% confidence interval generated from the data (693 to 820). A rather large number of new words (865) appears in the 100,000–110,000 range, which accounts for the hump in Figure 1 at the 110,000 point. Book 2 does not fare so well, since all the observed figures lie well above the predicted average. Examination of the text reveals that the extraordinarily large number of new words at the end of the book is due to the appearance of several numeric tables; numbers are treated as "words" by our tokenizer. Moreover, the subject matter of the book develops somewhat, which probably accounts for the greater than predicted incidence of new words from word 50,000 onwards. These observations explain the anomalies in Figure 2—the peak at the end and the upward trend before it—and highlight the breakdown of the fundamental assumption of the Poisson model, that the sample on which predictions are based is typical of the whole text.

Predicting novel characters and *n*-grams. When units are characters rather than words, a rather different picture emerges, for now it is B and C that fare badly, while A, P, and X exhibit very similar performance—although P and X are slightly better.

A "novel-character event" is defined analogously to the novel-word event, and as before the probability of this occurrence can be calculated according to each zero-frequency method. As noted previously, the escape probability assigned by method P (and X) becomes 1 if the number of types seen so far coincides with the number of tokens. This inevitably occurs after the very first token has been seen; not surprisingly it never happens again in the texts we evaluated. The escape probability using these methods becomes negative (or zero) when the least frequent token has been encountered more than once. This situation is much more likely when tokens are single characters than when they are words (or *n*-grams). Curiously, however, it only occurs once in the 763,362 characters of Hardy's text. The last few characters to appear are "3", "4", "2", "&", "9", and "8", in order of appearance. As a matter of interest, the characters "&" and ">", which both appear once only, are keying errors—the former in the context "m&thod" and the latter instead of a period at the end of a sentence. Thanks to the digits and these errors, the frequency of the least frequent token never exceeds 1, and the problem of anomalous prediction probability is minimal.

Figure 3 shows the average entropy throughout Hardy's text of the novel-token event according to each of the methods A, B, C, P, and X when tokens are characters, 2-grams (letter pairs), 3-grams (letter triples), 4-grams, 5-grams, and words. When characters are used as tokens, A outperforms B and C, while P and X are a shade better still. However, the relative performance of A decreases dramatically going from single characters to 2-grams, and remains low thereafter; while the performance of B and C—which is always very similar—improves gradually, as tokens grow from single characters to 5-grams. Although there exist situations in which A outperforms B and C, and others in which the reverse is true, P and X—which are indistinguishable from one another—are always better than all three methods. This gives confidence that the Poisson model is a valid one.

Application to the PPM text compression method. The PPM technique, described earlier, was originally introduced and evaluated using escape methods A and B (Cleary & Witten, 1984b). Subsequently Moffat (1988), in the course of developing an optimized implementation, investigated its performance using method C and found it superior to A and B (Moffat, 1988; findings verified independently by Bell *et al*, in press). Consequently we only compare methods C and X.

Unfortunately, with the low-frequency contexts used by PPM, method X frequently breaks down because t_1 is often equal to zero or n, making the novel-event probability 0 or 1. One *ad hoc* way of preventing this occurrence is to use an expression of the form $(t_1+\alpha)/(n+\beta)$. This was tested on a number of samples of text and other data, for various small values of α and β, and $\alpha=1$, $\beta=2$ was found to be uniformly best in terms of compression performance. Consequently method X below refers to this modification.

Figure 4 shows the relative performances of methods C and X in the PPM compression scheme, using 10 different ASCII files containing text of various types (English books, papers; source programs, news files, bibliography files) and four different binary files (two object programs, a grey-scale picture, and some geophysical data). This is the same corpus of test files for data compression that is used by Bell *et al* (in press). The Figure shows that for books 1 and 2, method X outperforms C by a small margin, while for most of the other text files they are almost identical. On all binary files, however, C outperforms X by a small margin.

Method X incorporates an *ad hoc* correction to avoid breakdown when t_1=0 or 1, as noted above. An obvious improvement is to use the more principled method X whenever it applies, but to revert to C in case of breakdown. This gives method XC:

$$Pr\,[\text{next event will be novel}] \; = \; \begin{cases} \dfrac{t_1}{n} & \text{when } 0<t_1<n \\[2ex] \dfrac{r}{n+r} & \text{otherwise.} \end{cases}$$

The performance of this modified method is also shown in Figure 4, and can be seen to give a slight improvement over method X alone on the text files, but is worse than both on the binary files.

The results of Figure 4 are aggregated over text and binary files in Table 4. The first columns show the number of files and their combined size. The number of bytes generated when the files are compressed by PPM is broken down into those used to code non-zero-frequency events and those needed for the zero-frequency ones. The former figure is the amount of information consumed in coding the actual characters of text in the appropriate context. The latter is given for the three different escape methods, C, X, and XC. As can be seen, XC is better than C for text files by about 4%, but worse for binary files by about 14%. Moreover, the Table shows that the number of bytes dedicated to escape codes is about 25% of the total for text files and 30% for binary ones. Thus the choice of escape method is not critical since it contributes relatively little to the size of the output. This also means that a better escape method does not necessarily improve the overall compression much; for example, method XC offers an improvement of about 1% over C in overall coding efficiency for text files.

Summary and conclusions

This paper has surveyed and compared several *ad hoc* approaches to the zero-frequency problem that have been used in text compression systems (A, B and C), and proposed and evaluated the Poisson process model (P), its close approximant (X), and the combination method (XC). We can summarize the outcome of the experiments as follows.

- P is uniformly best for all cases in which it applies—words, characters, and n-grams.

- X is virtually indistinguishable from P.

- A is a good approximation to X for character models, and in this case B and C behave badly. This is because t_1, the number of *hapax legomena,* is very small (less than 5 for books 1 and 2), so that method A's $1/(n+1)$ differs from X's t_1/n by only a small factor.

- B and C are good approximations to P for word and n-gram models with n>1 or 2, in which case A behaves badly. This is because t_1 becomes relatively large (more than 20% of r for words and 3-, 4-, and 5-grams of books 1 and 2). Note that the number of tokens r is $\sum t_i$, so that method B's r/n is $(t_1+t_2+\cdots)/n$, which comes within a small factor of method X's t_1/n.

- C is uniformly slightly better than B since it is smaller, and both over-estimate the escape probability.

However, P and X are untenable when t_1=0 (ie there are no *hapax legomena*), and unfortunately this is frequently the case in practical coding using the PPM method. The modified method XC (method X unless it is untenable, in which case method C) can be used in this case and outperforms C on text files. It does not fare so well on binary files, when the Poisson model clearly breaks down.

We conclude that the Poisson model gives uniformly better results than other methods in situations where it applies; further, a simple approximation to it is very easy to compute and just as accurate in practice. Although it breaks down when used directly with the PPM compression scheme, it can be patched by combining it with the best previously-known method to give a small improvement in overall coding efficiency on text files (about 1%). More significantly, the work gives a rationale for explaining why the relative performance of the previously-used *ad hoc* methods differ when different tokens are used by evaluating the extent to which they approximate the Poisson model, and confirms the suitability of methods B and C since they perform almost as well as the Poisson model.

Acknowledgements

We would like to thank Alistair Moffat for the use of his PPM program, and John Cleary and Radford Neal who have both been influential in developing and evaluating methods for the zero-frequency problem. This research is supported by the Natural Sciences and Engineering Council of Canada.

References

Bell, T.C. and Witten, I.H. (1987) "Greedy macro text compression" Research Report 87/285/33, Department of Computer Science, University of Calgary.

Bell, T.C., Witten, I.H., and Cleary, J.G. (in press) "Modeling for text compression" *Computing Surveys,* Also available as Research Report 88/327/39, Department of Computer Science, University of Calgary.

Bentley, J.L., Sleator, D.D., Tarjan, R.E., and Wei, V.K. (1986) "A locally adaptive data compression scheme" *Communications of the Association for Computing Machinery, 29* (4) 320-330, April.

Cleary, J.G. and Witten, I.H. (1984a) "A comparison of enumerative and adaptive codes" *IEEE Trans Information Theory, IT-30* (2) 306-315, March.

Cleary, J.G. and Witten, I.H. (1984b) "Data compression using adaptive coding and partial string matching" *IEEE Trans Communications, COM-32* (4) 396-402, April.

Efron, B. and Thisted, R. (1976) "Estimating the number of unseen species: how many words did Shakespeare know?" *Biometrika, 63* (3) 435-447.

Fiala, E.R. and Greene, D.H. (1989) ''Data compression with finite windows'' *Communications of the Association for Computing Machinery, 32* (4) 490-505, April.

Fisher, R.A., Corbet, A.S., and Williams, C.B. (1943) ''The relation between the number of species and the number of individuals in a random sample of an animal population'' *J Animal Ecology, 12,* 42-58.

Good, I.J. and Toulmin, G.H. (1956) ''The number of new species, and the increase in population coverage, when a sample is increased'' *Biometrika, 43* (Parts 1 & 2) 45-63, June.

Good, I.J. (1965) ''The estimation of probabilities'' Research Monograph 30, MIT Press.

Guazzo, M (1980) ''A general minimum-redundancy source-coding algorithm'' *IEEE Trans Information Theory, IT-26* (1) 15-25, January.

Jeffreys, H. (1939) *Theory of probability.* Clarendon, Oxford.

Johnson, N.L. and Katz, K. (1977) *Urn models and their application.* Wiley, New York.

Kolata, G. (1986) ''Shakespeare's new poem: an ode to statistics'' *Science, 231,* 335-336, 24 January.

Moffat, A. (1988) ''A note on the PPM data compression algorithm'' Research Report 88/7, Department of Computer Science, University of Melbourne, Parkville, Victoria 3052, Australia.

Pasco, R. (1976) ''Source coding algorithms for fast data compression'' PhD Thesis, Department of Electrical Engineering, Stanford University.

Pierce, C.S. (1956) ''The probability of induction'' in *The world of mathematics Vol 2,* edited by J.R.Newman, pp 1341-1354. Simon and Schuster, New York.

Rissanen, J.J. (1976) ''Generalized Kraft inequality and arithmetic coding'' *IBM J Research and Development, 20,* 198-203, May.

Rissanen, J. and Langdon, G.G. (1979) ''Arithmetic coding'' *IBM J Research and Development, 23* (2) 149-162, March.

Roberts, M.G. (1982) ''Local order estimating Markovian analysis for noiseless source coding and authorship identification'' PhD Thesis, Stanford University.

Rubin, F. (1979) ''Arithmetic stream coding using fixed precision registers'' *IEEE Trans Information Theory, IT-25* (6) 672-675, November.

Thomas, S.W., McKie, J., Davies, S., Turkowski, K., Woods, J.A., and Orost, J.W. (1985) ''*Compress* (version 4.0) program and documentation'' Available from joe@petsd.UUCP.

Welch, T.A. (1984) ''A technique for high-performance data compression'' *IEEE Computer, 17* (6) 8-19, June.

Witten, I.H., Neal, R., and Cleary, J.G. (1987) ''Arithmetic coding for data compression'' *Communications of the Association for Computing Machinery, 30* (6) 520-540, June.

Ziv, J. and Lempel, A. (1977) ''A universal algorithm for sequential data compression'' *IEEE Trans Information Theory, IT-23* (3) 337-343, May.

Ziv, J. and Lempel, A. (1978) ''Compression of individual sequences via variable-rate coding'' *IEEE Trans Information Theory, IT-24* (5) 530-536, September.

List of tables and figures

method	escape probability
A	$\frac{1}{n+1}$
B	$\frac{r}{n}$
C	$\frac{r}{n+r}$
P	$\frac{t_1}{n} - \frac{t_2}{n^2} + \frac{t_3}{n^3} - \cdots$
X	$\frac{t_1}{n}$

where n is the number of tokens seen so far
r is the number of types seen so far
t_i is the number of types that have been seen exactly i times so far

Table 1 Summary of different models for the escape probability

unit	number of types	number of tokens
characters	75	763,352
2-grams	1,547	763,351
3-grams	10,704	763,350
4-grams	39,514	763,349
5-grams	99,728	763,348
words	12,285	136,072

Table 2 Numbers of types and tokens in *Far from the madding crowd*

words	book 1	book 2
50,000– 60,000	818	597
60,000– 70,000	727	619
70,000– 80,000	751	761
80,000– 90,000	772	1190
90,000–100,000	718	
100,000–110,000	865	
110,000–120,000	790	
120,000–130,000	610	
average	756.4	791.8
predicted	798.0	455.4

Table 3 Number of new words appearing in different stretches of the books

type	files	input size	bytes to code				(XC/C)
			non-zero-frequency events	escape method			
				C	X	XC	
text	10	2,257,690	536,257	171,682	167,590	165,063	(96.1%)
binary	4	883,934	174,231	75,751	82,377	86,647	(114.4%)

Table 4 Performance in PPM coding: methods C, X, XC

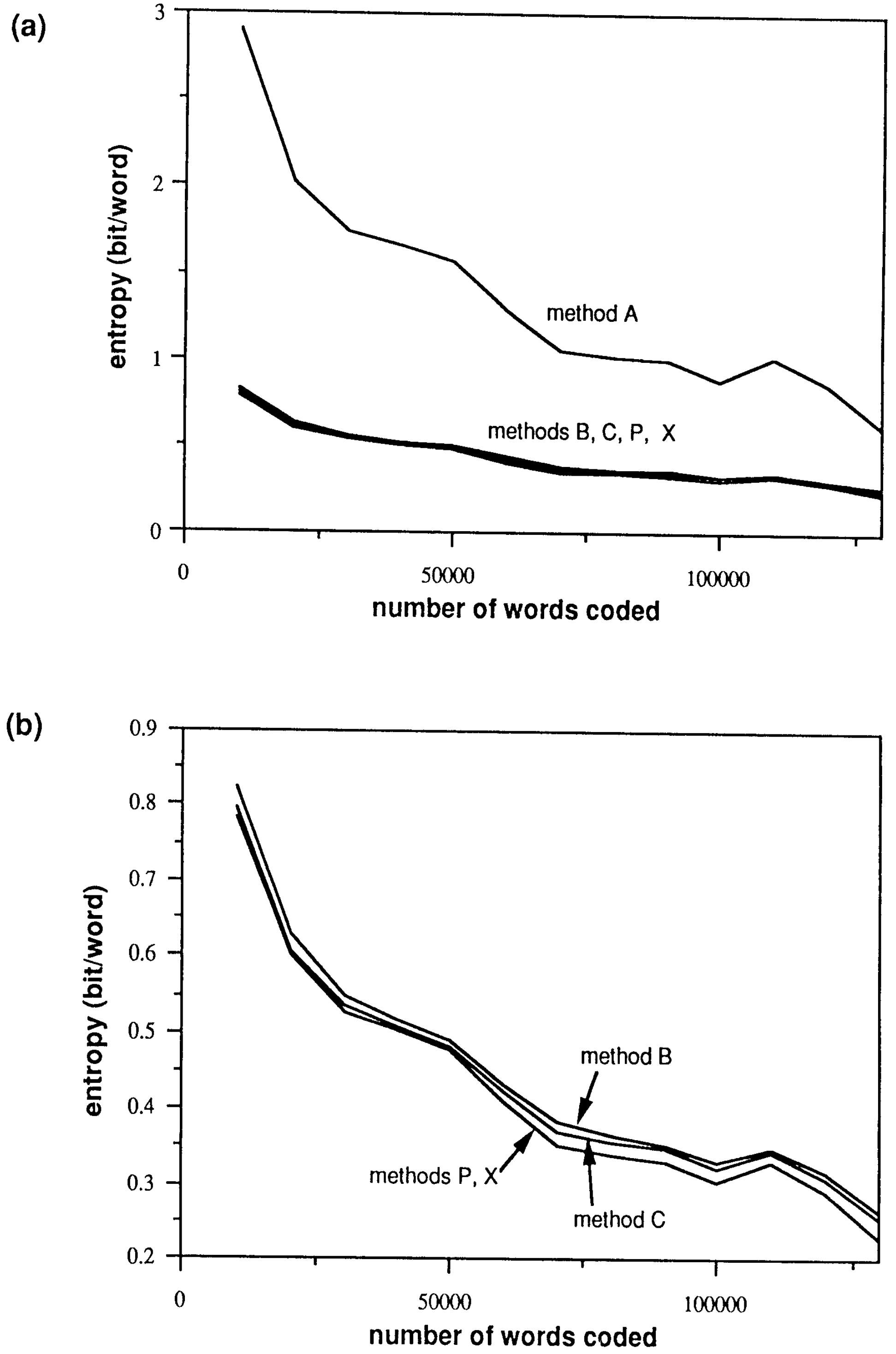
(a)
3
2
1
0
entropy (bit/word)
method A
methods B, C, P, X
0
50000
100000
number of words coded
(b)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
entropy (bit/word)
method B
methods P, X
method C
0
50000
100000
number of words coded

Figure 1

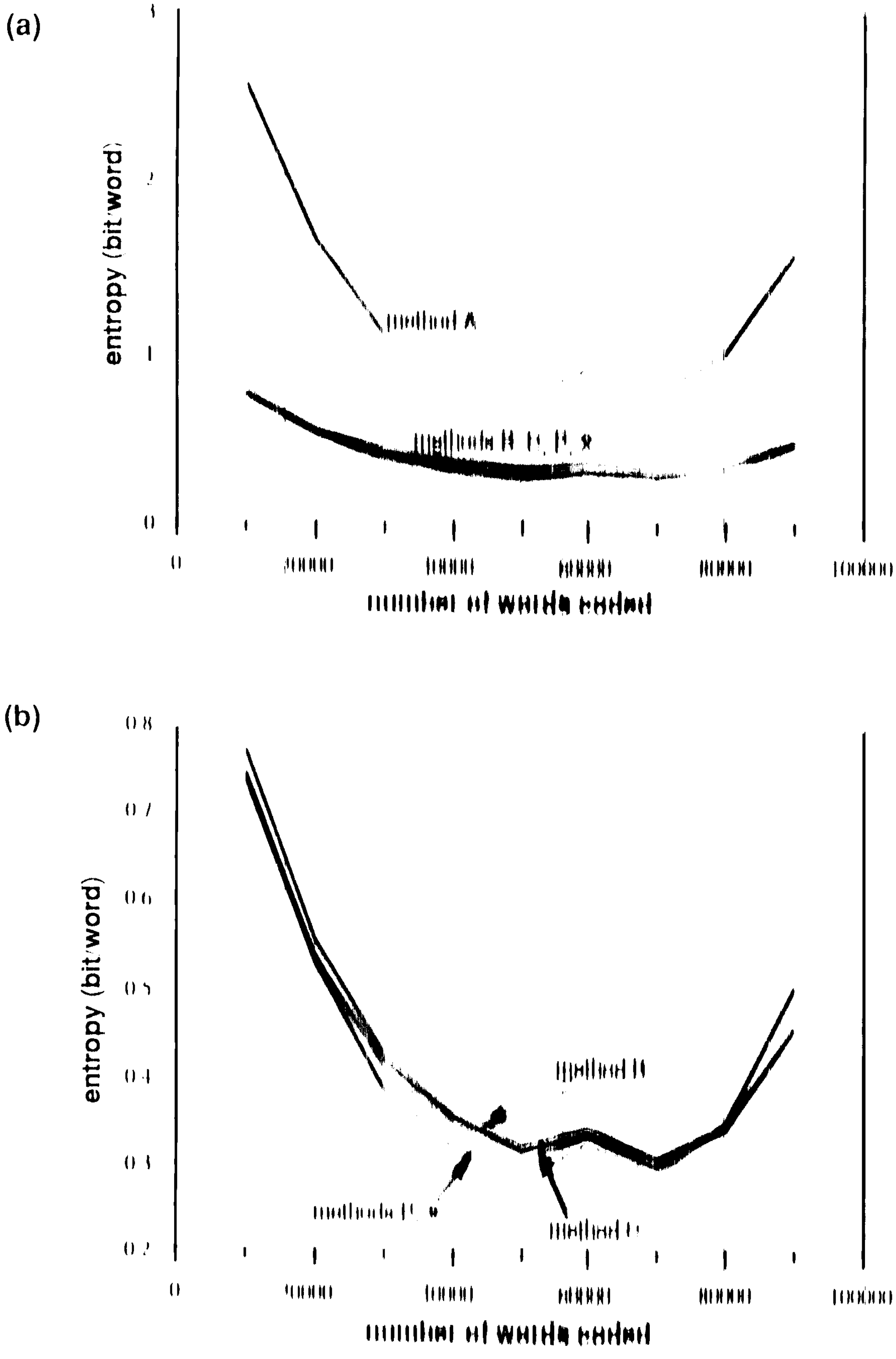
(a)
entropy (bit/word)
3
2
1
0
method A
methods B, C, D, &
0
20000
40000
60000
80000
100000
number of words coded
(b)
entropy (bit/word)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
method B
methods D, E
method C
0
20000
40000
60000
80000
100000
number of words coded

Figure 2

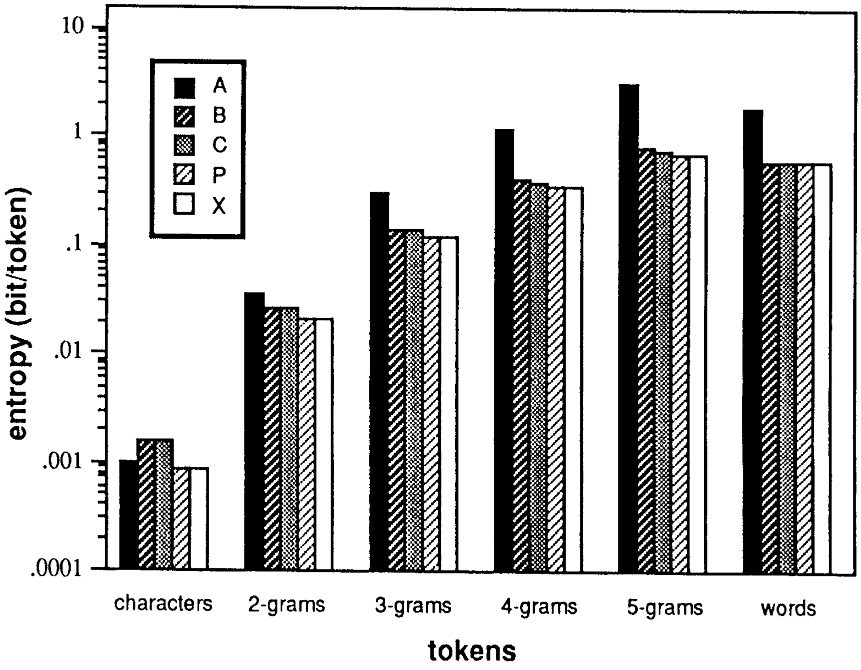
10
1
.1
.01
.001
.0001
entropy (bit/token)
A
B
C
P
X
characters
2-grams
3-grams
4-grams
5-grams
words
tokens

Figure 3

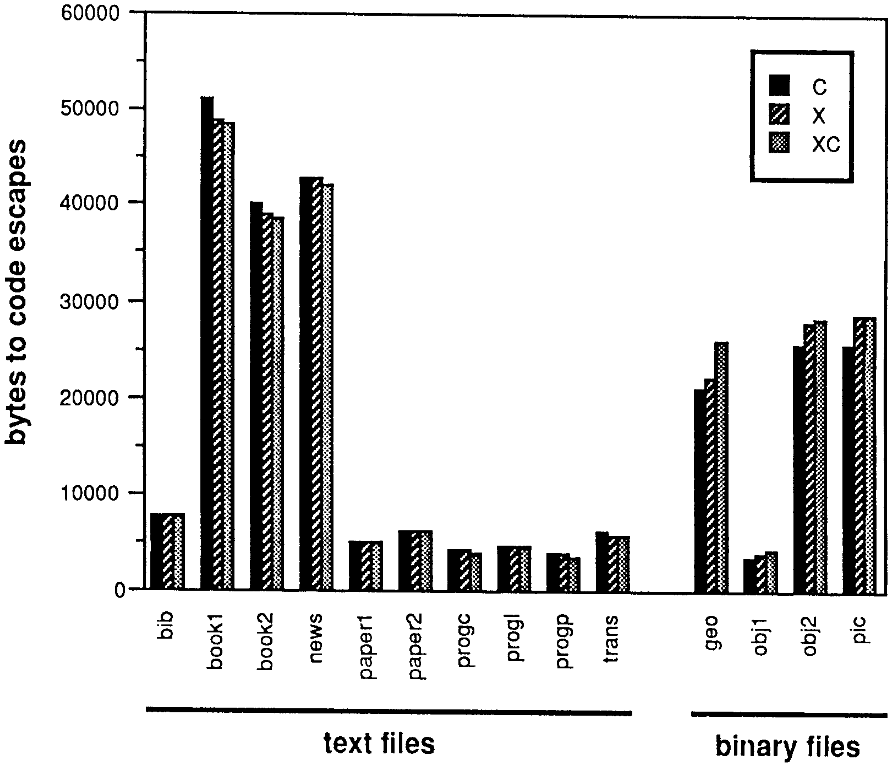
60000
50000
40000
30000
20000
10000
0
bytes to code escapes
C
X
XC
bib
book1
book2
news
paper1
paper2
progc
progl
progp
trans
geo
obj1
obj2
pic
text files
binary files

Figure 4