An Estimate of an Upper Bound for the Entropy of English

Peter F. Brown*
Vincent J. Della Pietra*
Robert L. Mercer*
IBM T.J. Watson Research Center

Stephen A. Della Pietra*
Jennifer C. Lai*

We present an estimate of an upper bound of 1.75 bits for the entropy of characters in printed English, obtained by constructing a word trigram model and then computing the cross-entropy between this model and a balanced sample of English text. We suggest the well-known and widely available Brown Corpus of printed English as a standard against which to measure progress in language modeling and offer our bound as the first of what we hope will be a series of steadily decreasing bounds.

1. Introduction

We present an estimate of an upper bound for the entropy of characters in printed English. The estimate is the cross-entropy of the 5.96 million character Brown Corpus (Kucera and Francis 1967) as measured by a word trigram language model that we constructed from 583 million words of training text. We obtain an upper bound of 1.75 bits per character.

Since Shannon's 1951 paper, there have been a number of estimates of the entropy of English. Cover and King (1978) list an extensive bibliography. Our approach differs from previous work in that

1. We use a much larger sample of English text; previous estimates were based on samples of at most a few hundred letters.
2. We use a language model to approximate the probabilities of character strings; previous estimates employed human subjects from whom probabilities were elicited through various clever experiments.
3. We predict all printable ASCII characters.

2. Method

Our estimate for the entropy bound is based upon the well-known fact that the cross-entropy of a stochastic process as measured by a model is an upper bound on the entropy of the process. In this section, we briefly review the relevant notions.

2.1 Entropy, Cross-Entropy, and Text Compression

Suppose $X = \{\ldots X_{-2}, X_{-1}, X_0, X_1, X_2 \ldots\}$ is a stationary stochastic process over a finite alphabet. Let P denote the probability distribution of X and let E_P denote expectations

* P.O. Box 704, Yorktown Heights, NY 10598

© 1992 Association for Computational Linguistics

with respect to P. The entropy of X is defined by

$$H(X) \equiv H(P) \equiv -E_P \log P(X_0 \mid X_{-1}, X_{-2}, \ldots). \quad (1)$$

If the base of the logarithm is 2, then the entropy is measured in bits. It can be shown that $H(P)$ can also be expressed as

$$H(P) = \lim_{n\to\infty} -E_P \log P(X_0 \mid X_{-1}, X_{-2}, \ldots, X_{-n}) = \lim_{n\to\infty} -\frac{1}{n} E_P \log P(X_1 X_2 \ldots X_n). \quad (2)$$

If the process is ergodic, then the Shannon–McMillan–Breiman theorem (Algoet and Cover 1988) states that almost surely

$$H(P) = \lim_{n\to\infty} -\frac{1}{n} \log P(X_1 X_2 \ldots X_n). \quad (3)$$

Thus, for an ergodic process, an estimate of $H(P)$ can be obtained from a knowledge of P on a sufficiently long sample drawn randomly according to P.

When P is not known, an upper bound to $H(P)$ can still be obtained from an approximation to P. Suppose that the stationary stochastic process M is a model for P. The cross-entropy of P as measured by M is defined by

$$H(P, M) \equiv -E_P \log M(X_0 \mid X_{-1}, X_{-2}, \ldots). \quad (4)$$

Under suitable regularity conditions, it can be shown that

$$H(P, M) = \lim_{n\to\infty} -E_P \log M(X_0 \mid X_{-1}, X_{-2}, \ldots, X_{-n}) = \lim_{n\to\infty} -\frac{1}{n} E_P \log M(X_1 X_2 \ldots X_n). \quad (5)$$

If P is ergodic, then it can be shown that almost surely for P

$$H(P, M) = \lim_{n\to\infty} -\frac{1}{n} \log M(X_1 X_2 \ldots X_n). \quad (6)$$

The cross-entropy $H(P, M)$ is relevant to us since it is an upper bound on the entropy $H(P)$. That is, for any model M,

$$H(P) \leq H(P, M). \quad (7)$$

The difference between $H(P, M)$ and $H(P)$ is a measure of the inaccuracy of the model M. More accurate models yield better upper bounds on the entropy. Combining Equations (6) and (7) we see that almost surely for P,

$$H(P) \leq \lim_{n\to\infty} -\frac{1}{n} \log M(X_1 X_2 \ldots X_n). \quad (8)$$

Entropy and cross-entropy can be understood from the perspective of text compression. It is well known that for any uniquely decodable coding scheme (Cover and Thomas 1991),

$$E_P\, l(X_1 X_2 \ldots X_n) \geq -E_P \log P(X_1 X_2 \ldots X_n), \quad (9)$$

where $l(X_1 X_2 \ldots X_n)$ is the number of bits in the encoding of the string $X_1 X_2 \ldots X_n$. Combining Equations (2) and (9), we see that $H(P)$ is a lower bound on the average number of bits per symbol required to encode a long string of text drawn from P:

$$H(P) \leq \lim_{n\to\infty} \frac{1}{n} E_P\, l(X_1 X_2 \ldots X_n). \quad (10)$$

On the other hand, an arithmetic coding scheme (Bell, Cleary, and Witten 1990) using model M will encode the sequence $x_1x_2\cdots x_n$ in

$$l_M(x_1x_2\ldots x_n) = \lceil -\log M(x_1x_2\ldots x_n) + 1 \rceil \tag{11}$$

bits, where $\lceil r \rceil$ denotes the smallest integer not less than r. Combining Equations (7) and (11) we see that $H(P,M)$ is the number of bits per symbol achieved by using model M to encode a long string of text drawn from P:

$$H(P,M) = \lim_{n\to\infty} \frac{1}{n} l_M(X_1X_2\ldots X_n). \tag{12}$$

2.2 The Entropy Bound

We view printed English as a stochastic process over the alphabet of 95 printable ASCII characters. This alphabet includes, for example, all uppercase and lowercase letters, all digits, the blank, all punctuation characters, etc. Using Equation (8) we can estimate an upper bound on the entropy of characters in English as follows:

1. Construct a *language model* M over finite strings of characters.
2. Collect a reasonably long *test sample* of English text.
3. Then

$$H(English) \leq -\frac{1}{n} \log M(test\ sample), \tag{13}$$

where n is the number of characters in the sample.

We emphasize that for this paradigm to be reasonable, the language model M must be constructed *without knowledge of the test sample*. Without this proscription, one might, for example, construct a model that assigns probability one to the test sample and zero to any other character string of the same length. Even quite subtle use of knowledge of the test sample can have a profound effect on the cross-entropy. For example, the cross-entropy would be noticeably lower had we restricted ourselves to characters that appear in the test sample rather than to all printable ASCII characters, and would be lower still had we used the actual vocabulary of the test sample. But these values could not be trumpeted as upper bounds to the entropy of English since Equation (13) would no longer be valid.

3. The Language Model

In this section, we describe our language model. The model is very simple: it captures the structure of English only through token trigram frequencies. Roughly speaking, the model estimates the probability of a character sequence by dissecting the sequence into tokens and spaces and computing the probability of the corresponding token sequence. The situation is slightly more complicated than this since, for a fixed token vocabulary, some character sequences will not have any such dissection while others will have several. For example, the sequence *abc xyz* might not have any dissection while the sequence *bedrock* might be dissected as one token or as two tokens without an intervening space.

We address the difficulty of sequences that cannot be dissected by introducing an *unknown token* that can account for any spelling. We address the problem of multiple

dissections by considering the token sequences to be hidden. The model generates a sequence of characters in four steps:

1. It generates a hidden string of *tokens* using a token trigram model.
2. It generates a *spelling* for each token.
3. It generates a *case* for each spelling.
4. It generates a *spacing* string to separate cased spellings from one another.

The final character string consists of the cased spellings separated by the spacing strings.

The probability of the character string is a sum over all of its dissections of the joint probability of the string and the dissection:

$$M(character_string) = \sum_{dissections} M(character_string, dissection). \tag{14}$$

The joint probability of the string and a dissection is a product of four factors:

$$\begin{aligned} &M(character_string, dissection) = \\ &\quad M_{token}(tokens)\, M_{spell}(spellings \mid tokens)\, M_{case}(cased_spellings \mid spellings, tokens) \\ &M_{space}(character_string \mid cased_spellings, spellings, tokens). \end{aligned} \tag{15}$$

3.1 The Token Trigram Model

The token trigram model is a second-order Markov model that generates a token string $t_1 t_2 \ldots t_n$ by generating each token t_i, in turn, given the two previous tokens t_{i-1} and t_{i-2}. Thus the probability of a string is

$$M_{token}(t_1 t_2 \ldots t_n) = M_{token}(t_1 t_2) \prod_{i=3}^{n} M_{token}(t_i \mid t_{i-2} t_{i-1}) \tag{16}$$

The conditional probabilities $M_{token}(t_3 \mid t_1 t_2)$ are modeled as a weighted average of four estimators f_i

$$M_{token}(t_3 \mid t_1 t_2) = \lambda_3(t_1 t_2) f_3(t_3 \mid t_1 t_2) + \lambda_2(t_1 t_2) f_2(t_3 \mid t_2) + \lambda_1(t_1 t_2) f_1(t_3) + \lambda_0(t_1 t_2) f_0, \tag{17}$$

where the weights λ_i satisfy $\sum \lambda_i = 1$ and $\lambda_i \geq 0$.

The estimators f_i and the weights λ_i are determined from the training data using a procedure that is explained in detail by Jelinek and Mercer (1980). Basically, the training data are divided into a large, primary segment and a smaller, held-out segment. The estimators f_i are chosen to be the conditional frequencies in the primary segment, while the smoothing weights λ_i are chosen to fit the combined model to the held-out segment. In order to decrease the freedom in smoothing, the λ_i are constrained to depend on $(t_1 t_2)$ only through the counts $c(t_1 t_2)$ and $c(t_2)$ in the primary training segment. When $c(t_1 t_2)$ is large, we expect $\lambda_3(t_1 t_2)$ to be close to 1, since in this case the trigram frequency in the primary segment should be a reliable estimate of the

frequency in the held-out segment. Similarly, when $c(t_1t_2)$ is small, but $c(t_2)$ is large, we expect $\lambda_3(t_1t_2)$ to be close to 0 and $\lambda_2(t_1t_2)$ to be close to 1.

The token vocabulary consists of

1. 293,181 spellings, including a separate entry for each punctuation character;
2. a special *unknown_token* that accounts for all other spellings;
3. a special *sentence_boundary_token* that separates sentences.

3.2 The Spelling Model

The spelling model generates a spelling $s_1s_2 \ldots s_k$ given a token. For any token other than the *unknown_token* and *sentence_boundary_token*, the model generates the spelling of the token. For the *sentence_boundary_token*, the model generates the null string. Finally, for the *unknown_token*, the model generates a character string by first choosing a length k according to a Poisson distribution, and then choosing k characters independently and uniformly from the printable ASCII characters. Thus

$$M_{spell}(s_1s_2 \ldots s_k \mid unknown_token) = \frac{\lambda^k}{k!}e^{-\lambda}p^k, \tag{18}$$

where λ is the average number of characters per token in the training text, 4.1, and $1/p$ is the number of printable ASCII characters, 95.

3.3 The Case Model

The case model generates a cased spelling given a token, the spelling of the token, and the previous token. For the *unknown_token* and *sentence_boundary_token*, this cased spelling is the same as the spelling. For all other tokens, the cased spelling is obtained by modifying the uncased spelling to conform with one of the eight possible patterns

$$L^+ \quad U^+ \quad UL^+ \quad ULUL^+ \quad ULLUL^+ \quad UUL^+ \quad UUUL^+ \quad LUL^+$$

Here U denotes an uppercase letter, L a lowercase letter, U^+ a sequence of one or more uppercase letters, and L^+ a sequence of one or more lowercase letters. The case pattern only affects the 52 uppercase and lowercase letters.

The case pattern C for a token t is generated by a model of the form:

$$M_{case}(C \mid t, b) = \lambda_2(t)f(C \mid t, b) + \lambda_1(t)f(C \mid b) + \lambda_0(t). \tag{19}$$

Here b is a bit that is 1 if the previous token is the *sentence_boundary_token* and is 0 otherwise. We use b to model capitalization at the beginning of sentences.

3.4 The Spacing Model

The spacing model generates the spacing string between tokens, which is either null, a dash, an apostrophe, or one or more blanks. It is generated by an interpolated model similar to that in Equation (19). The actual spacing that appears between two tokens should depend on the identity of each token, but in our model we only consider the dependence on the second token. This simplifies the model, but still allows it to do

a good job of predicting the null spacing that precedes many punctuation marks. For strings of blanks, the number of blanks is determined by a Poisson distribution.

3.5 The Entropy Bound

According to the paradigm of Section 2.2 (see Equation (13)), we can estimate an upper bound on the entropy of characters in English by calculating the language model probability $M(character_string)$ of a long string of English text. For a very long string it is impractical to calculate this probability exactly, since it involves a sum over the different hidden dissections of the string. However, for any particular dissection $M(character_string) \geq M(character_string, dissection)$. Moreover, for our model, a straightforward partition of a character string into tokens usually yields a dissection for which this inequality is approximately an equality. Thus we settle for the slightly less sharp bound

$$H(English) \leq -\frac{1}{n} \log M(character_string, dissection) \tag{20}$$

where *dissection* is provided by a simple finite state tokenizer. By Equation (15), the joint probability $M(characterstring, dissection)$ is the product of four factors. Consequently, the upper bound estimate (20) is the sum of four entropies,

$$\begin{aligned} H(English) \quad \leq \quad & H_{token}(character_string) + H_{spell}(character_string) \\ & + H_{case}(character_string) + H_{spacing}(character_string). \end{aligned} \tag{21}$$

4. The Data

4.1 The Test Sample

We used as a test sample the Brown Corpus of English text (Kucera and Francis 1967). This well-known corpus was designed to represent a wide range of styles and varieties of prose. It consists of samples from 500 documents, each of which first appeared in print in 1961. Each sample is about 2,000 tokens long, yielding a total of 1,014,312 tokens (according to the tokenization scheme used in reference [Kucera and Francis 1967]).

We used the Form C version of the Brown Corpus. Although in this version only proper names are capitalized, we modified the text by capitalizing the first letter of every sentence. We also discarded paragraph and segment delimiters.

4.2 The Training Data

We estimated the parameters of our language model from a training text of 583 million tokens drawn from 18 different sources. We emphasize that this training text *does not include the test sample*. The sources of training text are listed in Table 1 and include text from:

1. several newspaper and news magazine sources: the Associated Press; the United Press International (UPI); the *Washington Post*; and a collection of magazines published by Time Incorporated;
2. two encyclopedias: *Grolier's Encyclopedia* and the *McGraw-Hill Encyclopedia of Science and Technology*;

Table 1
Training corpora.

Source	Millions of words
United Press International	203.768
IBM Depositions	93.210
Canadian Parliament	85.016
Amoco PROFS (OC)	54.853
Washington Post	40.870
APHB	30.194
Associated Press	24.069
IBM Poughkeepsie (OC)	22.140
Time Inc.	10.525
Grolier's Encyclopedia	8.020
McGraw-Hill Encyclopedia	2.173
IBM Sterling Forest (OC)	1.745
IBM Research (OC)	1.612
Bartlett's Familiar Quotations	0.489
Congressional Record	0.344
Sherlock Holmes	0.340
Chicago Manual of Style	0.214
World Almanac and Book of Facts	0.173
Total	582.755

3. two literary sources: a collection of novels and magazine articles from the American Printing House for the Blind (APHB) and a collection of Sherlock Holmes novels and short stories;
4. several legal and legislative sources: the 1973–1986 proceedings of the Canadian parliament; a sample issue of the Congressional Record; and the depositions of a court case involving IBM;
5. office correspondence (OC) from IBM and from Amoco;
6. other miscellaneous sources: *Bartlett's Familiar Quotations*, the *Chicago Manual of Style*, and *The World Almanac and Book of Facts*.

4.3 The Token Vocabulary

We constructed the token vocabulary by taking the union of a number of lists including:

1. two dictionaries;
2. two lists of first and last names: a list derived from the IBM on-line phone directory, and a list of names we purchased from a marketing company;
3. a list of place names derived from the 1980 U.S. census;
4. vocabulary lists used in IBM speech recognition and machine translation experiments.

Table 2
Tokens in the test sample but not in the 293,181-token vocabulary.

Token	Occurrences
*J	1776
*F	1004
Khrushchev	68
Kohnstamm	35
skywave	31
Prokofieff	28
Helva	22
patient's	21
dikkat	21
Podger	21
Katanga	21
ekstrohm	20
Skyros	20
PIP	17
Lalaurie	17
roleplaying	16
Pont's	15
Fromm's	15
Hardy's	15
Helion	14

The resulting vocabulary contains 89.02% of the 44,177 distinct tokens in the Brown Corpus, and covers 99.09% of 1,014,312-token text. The twenty most frequently occurring tokens in the Brown Corpus not contained in our vocabulary appear in Table 2. The first two, *J and *F, are codes used in the Brown Corpus to denote formulas and special symbols.

5. Results and Conclusion

The cross-entropy of the Brown Corpus and our model is 1.75 bits per character. Table 3 shows the contributions to this entropy from the token, spelling, case, and spacing components (see Equation (21)). The main contribution is, of course, from the token model. The contribution from the spelling model comes entirely from predicting the spelling of the *unknown_token*. The model here is especially simple-minded, predicting each of the 95 printable ASCII characters with equal probability. While we can easily do better, even if we were able to predict the characters in unknown tokens as well as we predict those in known tokens, the contribution of the spelling model to the entropy would decrease by only 0.04 bits. Likewise, we can entertain improvements to the case and spacing models but any effect on the overall entropy would be small.

Our bound is higher than previous entropy estimates, but it is statistically more reliable since it is based on a much larger test sample. Previous estimates were necessarily based on very small samples since they relied on human subjects to predict characters. Quite apart from any issue of statistical significance, however, it is probable that people predict English text better than the simple model that we have employed here.

The cross-entropy of a language model and a test sample provides a natural quantitative measure of the predictive power of the model. A commonly used measure of the difficulty of a speech recognition task is the word perplexity of the task (Bahl et

Table 3
Component contributions to the cross-entropy.

Component	Cross-Entropy (bits)
Token	1.61
Spelling	0.08
Case	0.04
Spacing	0.02
Total	1.75

al. 1977). The cross-entropy we report here is just the base two logarithm of the character perplexity of a sample of text with respect to a language model. For a number of natural language processing tasks, such as speech recognition, machine translation, handwriting recognition, stenotype transcription, and spelling correction, language models for which the cross-entropy is lower lead directly to better performance.

We can also think of our cross-entropy as a measure of the compressibility of the data in the Brown Corpus. The ASCII code for the characters in the Brown Corpus has 8 bits per character. Because only 95 of the characters are printable, it is a straightforward matter to reduce this to 7 bits per character. With a simple Huffman code, which allots bits so that common characters get short bit strings at the expense of rare characters, we can reach 4.46 bits per character. More exotic compression schemes can reach fewer bits per character. For example, the standard UNIX command *compress*, which employs a Lempel–Ziv scheme, compresses the Brown Corpus to 4.43 bits per character. Miller and Wegman (1984) have developed an adaptive Lempel–Ziv scheme that achieves a compression to 4.20 bits per character on the Brown Corpus. Our language model allows us to reach a compression to 1.75 bits per character.

We do not doubt that one can reduce the cross-entropy below 1.75 bits per character. A simple way to do this is to find more reliable estimates of the parameters of the model by using a larger collection of English text for training. We might also consider structural changes to the model itself. Our model is static. One can imagine adaptive models that profit from the text in the early part of the corpus to better predict the later part. This idea is applicable to the token model and also to the spelling model.

From a loftier perspective, we cannot help but notice that linguistically the trigram concept, which is the workhorse of our language model, seems almost moronic. It captures local tactic constraints by sheer force of numbers, but the more well-protected bastions of semantic, pragmatic, and discourse constraint and even morphological and global syntactic constraint remain unscathed, in fact unnoticed. Surely the extensive work on these topics in recent years can be harnessed to predict English better than we have yet predicted it.

We see this paper as a gauntlet thrown down before the computational linguistics community. The Brown Corpus is a widely available, standard corpus and the subject of much linguistic research. By predicting the corpus character by character, we obviate the need for a common agreement on a vocabulary. Given a model, the computations required to determine the cross-entropy are within reach for even a modest research budget. We hope by proposing this standard task to unleash a fury of competitive energy that will gradually corral the wild and unruly thing that we know the English language to be.

References

Algoet, P. and Cover, T. (1988). "A sandwich proof of the Shannon–McMillan–Breiman theorem." *Annals of Probability* 16(2):899–909.

Bahl, L., Baker, J., Jelinek, F., and Mercer, R. (1977). "Perplexity—a measure of the difficulty of speech recognition tasks." In *Program, 94th Meeting of the Acoustical Society of America* 62:S63, Suppl. no. 1.

Bell, T. C., Cleary, J. G., and Witten, I. H. (1990). *Text Compression*. Englewood Cliffs, N.J.: Prentice Hall.

Cover, T., and King, R. (1978). "A convergent gambling estimate of the entropy of English." *IEEE Transactions on Information Theory* 24(4):413–421.

Cover, T. M., and Thomas, J. A. (1991). *Elements of Information Theory*. New York: John Wiley.

Jelinek, F., and Mercer, R. L. (1980). "Interpolated estimation of Markov source parameters from sparse data." In *Proceedings, Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands.

Kucera, H., and Francis, W. (1967). *Computational Analysis of Present-Day American English*. Providence, R.I.: Brown University Press.

Miller, V. S., and Wegman, M. N. (1984). "Variations on a theme by Ziv and Lempel." Technical Report RC 10630, IBM Research Division.

Shannon, C. (1951). "Prediction and entropy of printed English." *Bell Systems Technical Journal* 30:50–64.