Statistical structure in natural language

Tom Jackson

May 7, 2003

Abstract

We make use of recent results in information theory to study long-range correlations in selections of English language texts. Specifically, we choose to measure the mutual information between pairs of words as a function of the separation of those words. We find that this appears to decay as a power law whose exponent changes at length scales which, we hypothesize, are related to the different processes governing the composition of a text.

1 Introduction

This is the official description of the independent work I've done for the Program in Applied and Computational Mathematics certificate. This work began in some form as a Junior Paper in the physics department done with William Bialek in the spring of 2002. The majority of the computational work was done over that summer.

The motivation for this work came from the preprints [11, 12], who found statistical regularities in the way words are distributed in English language texts. The idea of studying language statistically is rather odd — everyone who uses language has an intuitive grasp of how it works — but can be motivated from a biophysical point of view. We can look at human language as any other evolved pattern of behavior and ask the usual questions one asks in biophysics: how is it organized? How might it have evolved? Is it maximally efficient? [13, 14]. Statistical methods allow us to investigate these questions with some degree of impartiality.

As a more concrete example, we recall that the most well-known statistical pattern in natural language is Zipf's law [27]. Based on empirical observations, Zipf noted that if one constructs a frequency histogram from words in any natural language it has a power-law slope (figure 1); i.e, the number of times the nth most common word in a text appears is proportional to $1/n^{\rho}$. The power-law exponent $\rho \approx 1$ for a wide variety of texts; this would imply that this distribution is not normalizable. The explanation for this is that the hypothetical infinite text which would have this word distribution can only have a finite number of words, otherwise we can't really claim that it's an example of a naturally occurring language. In this work we try to take the next logical step — instead of considering the distribution of individual words, we'd like to consider a quantity

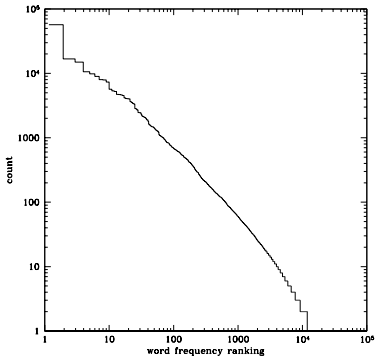

Figure 1: The sorted frequency histogram for the text of *War and Peace* (described below), showing Zipf law behavior.

analogous to the two-point correlation function which measures correlations between pairs of words. We find that the best measure of correlation for sets of words is definable in terms of information theory, which we review now.

2 Information theory

It was something of a triumph for applied mathematics when Claude Shannon assigned a precise, quantitative meaning to the concept of "information". In his sense, "information" is synonymous with "surprise" or "unexpectedness": events which have a low probability of happening are held to be more informative. From three axioms, Shannon was able to uniquely define the entropy ([19], appendix 2)

$$S(\{p_i\}) = -\sum_i p_i \log p_i$$

as the unique (up to a multiplicative constant) measure of information for the discrete probability distribution $\{p_i\}$. The sense in which this coincides with our intuitive concept of "information" is defined by the noiseless coding theorem, which states that the values of any random variable can be used to encode those of any other variable with a smaller or equal entropy in such a way that all properties of the distribution of the second variable can be reconstructed from those of the first. In this sense, we can think of $S(X)$ as the capacity of X to store information.

We notice that for independent variables, the entropy adds: $S(X \cdot Y) = S(X) + S(Y)$. We can define a measure of how correlated two variables are through their departure from additivity; in other words we can say

$$I(X;Y) = S(X) + S(Y) - S(X \cdot Y)$$

This quantity is called the mutual information between X and Y. In terms of probabilities,

$$I(X,Y) = \sum_{x,y} p_{xy} \log \frac{p_{xy}}{p_x p_y}$$

From this we can see that the mutual information is symmetric: X provides as much information

about Y as Y does about X. In particular, we have $S(X) \geq I(X;Y)$ and $S(Y) \geq I(X;Y)$ for any X, Y: $I(X;Y)$ represents the portion of information in X that can be said to be "about" Y.

2.1 Mutual information vs. correlation function

So far, we've been presenting the mutual information as a measure of how " correlated" two variables are, without making reference to the standard measure of correlation, the correlation function. It's clear that both are measures of how "ordered" a system is, but there are very good reasons for preferring the mutual information (as pointed out by Li [9]) when dealing with discrete variables. In a nutshell, correlation functions involve some notion of metric on the space of $\{x_i\}$; we need to specify how similar to one another each of the possible events are. Rather than attempt to define such a metric on the space of words *a priori*, we can instead use the mutual information, which is defined entirely in terms of probabilities and makes no reference to the identities of the words themselves. In this sense, mutual information is the appropriate measure of correlation to use for any variables that aren't defined with respect to a metric.

2.2 Mutual information in language

Shannon was an electrical engineer, and some of his early studies were concerned with the information content of natural language. In one experiment [3], he showed people an incomplete word and asked them to guess the next letter. By recording the number of guesses it took to arrive at the right answer, Shannon estimated the amount of entropy in a single letter. The result was surprising — since there are 26 letters in the alphabet one would expect the entropy to be less than 4.7 bits/letter, but Shannon's value was less than two bits (the estimation of entropy from sample data becomes complicated as the number of configurations grows large, for reasons explained below): this means that much of the information in written text is redundant.

The entropy of language is reduced because of the many structural factors that go into the composition of any passage. Letters are grouped into phonemes and phonemes into words, and here we have the constraint that words be (approximately) pronounceable, as well as fact that we have a system of modifying words in regular ways (called morphology in linguistics), for example, by adding prefixes or suffixes. Words are organized into phrases and phrases into sentences through the rules of syntax. Other regularities at the scale of a sentence arise through the use of colloquial expressions and figures of speech. Finally, we can think of sentences being organized into paragraphs and paragraphs into words under the constraints of composition: one topic must be developed before we move on to the next, and so on.

Ultimately, on the largest scales we expect correlations in a passage because it is *about* something, and we would not expect sudden random shifts in subject. This points to what is probably the Achilles' heel of looking at language in this way: words refer to something in the outside world; they make reference to information that any reader has acquired in a lifetime of experience but which is not explicit in the text[1]. Not to put too fine a point on it, this is the problem of *meaning*,

[1]For example, we know to expect the appearance of the word "sunny" to be more correlated with that of "weather" than "cheese", but we have no reason for this if we only regard these words as undistinguished values of some random

and it provides the foundation of Noam Chomsky's argument that there is no place for statistics in language [1, 3]. In particular, certain words may possess great intrinsic significance even though they appear infrequently. We return to these questions later.

Our goal of obtaining two-point statistics on large samples of text is complicated by the number of possible word pairs: if we try to naïvely estimate all the joint probabilities from a sample by computing relative frequencies, we need an impractical amount of data before our estimates are constrained to within reasonable errors. This is essentially the problem of inferring a probability distribution[2] from a finite set of data drawn from it. In the case where the data are continuous one solves this problem by making various smoothness assumptions [4], but in the discrete case it's unclear how "smoothness" should be defined. We now turn to the resolution of this question and to the mathematical methods used in this project.

3 Robust Entropy Estimation

3.1 Statement of the problem

We have set for ourselves the task of estimating the mutual information between word pairs in a typical document. To do this well, we need to deal with an almost universal fact of life in these application of information theory: undersampling. As an example, one document we looked at[3] was $\sim 10^6$ words long and contained $\sim 10^4$ different words. This is enough to sample the single-word entropy well, but there are $\sim 10^8$ possible word pairs and so that distribution will be severely undersampled. Making use of longer documents helps to some extent — at one extreme, every word in a sentence can be unique — but the number of different words scales with the size of the document, so this approach would require an impractically large corpus.

The approach we take is instead to use more sophisticated methods for estimating the entropy. Denote the number of times word i appears by n_i, set $N = \sum_i n_i$, and let k be the number of distinct words. The Maximum Likelihood estimator for the probability of word i to occur is just n_i/N, so one might suppose that the logical estimator for the entropy of this distribution would be

$$\hat{S}[n] = \sum_i -\frac{n_i}{N} \log_2 \frac{n_i}{N}$$

In the limit $N \to \infty$ this is the definition of entropy, but in the limit $N < k$ this leaves much to be desired: in a nutshell, this estimator assigns all unobserved events zero probability[4] when in reality we just didn't happen to observe them.

variable.

[2] Let us not forget that entropy is a property only of a distribution and not of a particular set of data.

[3] The Project Gutenberg text of *War and Peace*

[4] We have $\lim p \to 0 p \log p = 0$

3.2 Bayesian estimation

We proceed further by introducing ideas from Bayesian statistics. Recall that

$$\Pr(a|b)\Pr(b) = \Pr(b|a)\Pr(a) = \Pr(a \cdot b)$$

Bayes' rule uses this to say

$$\Pr(\text{data}|\text{model}) = \frac{\Pr(\text{data}|\text{model})\Pr(\text{model})}{\Pr(\text{data})}$$

While Pr(data) is constant and may be ignored, Pr(model) (known as the *prior*) cannot. Bayes' rule simply tells us how to update these prior estimates in the face of the data we have — what it does not do, and where much of the confusion surrounding Bayesian statistics lies, is tell us what the prior should be. A badly chosen prior distribution will produce a biased estimator: if the prior rates model distributions having a certain parameter value as being highly probable, large amounts of data will be needed to outweigh this and shift the estimation of the parameter to some other value. On the other hand, the prior is convenient in excluding possible situations that are physically impossible.

We consider the Dirichlet family of priors, which are parameterized by β:

$$\Pr(\{q_i\}) = \frac{1}{Z(\beta)}\delta\left(1 - \sum_{i=1}^{k} q_i\right)\prod_{i=1}^{k} q_i^{\beta-1}$$

where $Z(\beta)$ is a straightforward normalization factor. The $\delta-$function is simply a statement that distributions $\{q_i\}$ which are not properly normalized are given zero probability. The product is the real "active ingredient" here: by increasing β, we make distributions close to uniform more probable.

The Dirichlet prior can be implemented by the simple recipe of "add β to every bin":

$$\langle q_i \rangle = \frac{n_i + \beta}{N + k\beta}$$

We abbreviate $k\beta$ by κ in what follows. This solves the problem of unobserved events being assigned zero probability. In a derivation too long to reproduce here, Wolpert and Wolf [25, 26] found bayesian estimators for any moments of the distribution of the entropy assuming a Dirichlet prior:

$$\langle S[\{n_i\},\beta]\rangle = -\sum_i \frac{n_i+\beta}{N+\kappa}\left(\psi_0(n_i+\beta+1) - \psi_0(N+\kappa+1)\right) \tag{1}$$

$$\begin{aligned}\langle S^2[\{n_i\},\beta]\rangle = &\sum_{i\neq j}\frac{(n_i+\beta)(n_j+\beta)}{(N+\kappa)(N+\kappa+1)}[(\psi_0(n_i+\beta+1)-\psi_0(N+\kappa+2))\\ &\times(\psi_0(n_j+\beta+1)-\psi_0(N+\kappa+2)) - \psi_1(N+\kappa+2)]\\ &+\sum_i \frac{(n_i+\beta)(n_i+\beta+1)}{(N+\kappa)(N+\kappa+1)}[(\psi_0(n_i+\beta+2)-\psi_0(N+\kappa+2))^2\\ &+(\psi_1(n_i+\beta+2)-\psi_1(N+\kappa+2))]\end{aligned} \tag{2}$$

where $\psi_n(z) \equiv d^{n+1}/dz^{n+1} \log \Gamma(z)$ is the polygamma function.

Returning to the issue of bias, it's instructive to evaluate these in the case of zero counts, i.e. before any data has been observed. These give

$$\langle S[\{n_i = 0\}, \beta] \rangle = \psi_0(\kappa + 1) - \psi_0(\beta + 1)$$
$$\langle S^2[\{n_i = 0\}, \beta] \rangle = \frac{\beta + 1}{\kappa + 1} \psi_1(\beta + 1) - \psi_1(\kappa + 1)$$

as the first and second moments of the distribution of entropies the prior gives us — i.e, this is our estimate of the entropy before any data has been observed. Our goal is something uniform over the entire range of possible entropies $[0, \log k]$, but as figure 3.2 shows the prior is very far from uniform and becomes more biased as K increases.

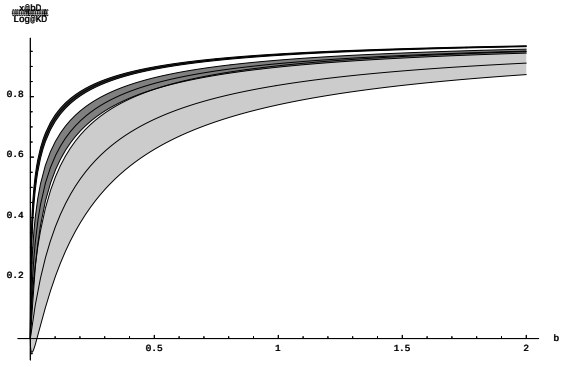

Figure 2: Plot of the prior estimated entropy as a function of the Dirichlet parameter β, for different numbers of categories K. The shaded bands denote our estimated 1σ error bars.

The key observation of [6] is to turn this weakness into a strength. As K grows larger, the variance in prior entropy gets smaller and the distribution of entropies gets sharply peaked. By integrating over β, we can in effect obtain a uniform prior. We have

$$\hat{S^m} = \frac{\int \frac{d\beta}{d\xi} \rho(\{n_i\}, \beta) \langle S^m[\{n_i\}, \beta] \rangle}{\int \frac{d\beta}{d\xi} \rho(\{n_i\}, \beta)} \tag{3}$$

where the weighting factor ρ is given by

$$\rho(\{n_i\}, \beta) = \frac{\Gamma(\kappa)}{\Gamma(N + \kappa)} \prod_{i=1}^{k} \frac{\Gamma(n_i + \beta)}{\Gamma(\beta)}$$

If the data are such that ρ is peaked at some value of β, this value dominates the integral in the numerator of 3 and we wind up with the estimated entropy corresponding to that value of β. This process can fail if the data fail to show any peak or if we obtain $\beta \to 0$, which causes the estimator to reduce to the MLE result 3.1. β acts as a smoothness parameter, and the $\beta \to 0$ limit is the analogue of overfitting the distribution: given enough parameters/enough roughness, we can always produce a model that matches any data exactly.

4 Implementation Details

Here we summarize the technical aspects of how our computations were carried out.

The procedure above was implemented in roughly 5000 lines of C. One program converts text files into lists of tokens. The other program reads those lists of tokens, constructs the appropriate word pairs, calls the entropy estimation and information bottleneck routines, and outputs the results. Implementing these programs was straightforward, with a few subtleties introduced by size and memory constraints.

4.1 Tokenizing

The problem here is to read in a text file and assign each unique word its own integer (a "token"). We reserve a few tokens to denote the beginning of new sentences, paragraphs, etc. Reading in the words is simple: given a position in the file we add characters to a buffer, converting all capitals to lowercase and removing apostrophes. The buffer and the current file position are returned when a non-alphabetic character is read. Several minor details involve the formatting of the file in question; for example the Reuters corpus contains SGML tags whose contents we ignore.

More difficult is the problem of indexing: once we've read in a word, how do we know if we've seen it before? To efficiently index the document we make use of a ternary search tree [18]. For our case, this simply a tree where each node contains a letter and has three links coming from it, corresponding to the next letter in the word coming before, after or equal to that letter in alphabetical order. All the words stored in the tree can be read out by following a succession of "equal" links. At the node corresponding to the end of each word we store the token assigned to it and a pointer to the file position where that word was first encountered. The file pointer is necessary because we only store as many letters of a word as we need to distinguish it from the other words: when we come to the end of a word in the tree in a search operation, we look up the complete word in the file and compare it to the word we want to add to the tree. If the word is new, we add it to the tree and create a new token for it; if not we return the token that was previously assigned to it (and stored in the tree).

To improve search time, instead of one search tree we maintain an array of 26^2 trees (most of which have zero length): the first two letters of a word are stored directly as the index of this array. The ternary search tree offers savings in both program running time and memory space: most nodes are used as part of more than one word, and the tree structure ensures that looking up a word takes a

number of operations that scales as the logarithm of the word's length.

At the end of the first pass through the text file, we've assigned all the tokens and stored all the words in the tree. We've output the tokens and a list of what token was assigned with each word. As a matter of convenience we want to assign tokens in order of word frequency: in other words, the most common word in the file gets the first token, the second most common word gets the second token, etc. This makes subsequent debugging and data analysis much easier, and is comparatively simple to accomplish.

To assign the tokens in order of word frequency, we make another pass through the list of tokens and build up a histogram of token frequencies. We sort this to obtain a translation table of the frequency ranks for each token. We then simply make a third pass through the token and key files and replace the old tokens by their new values. We finally sort the key file by token number.

4.2 Analysis of word pairs

Now that we have a tokenized text file, we want to compute the mutual information between word pairs at various separations: in other words we just need to assemble a histogram of how frequently the word pairs occur and pass it to the entropy estimation routines. The simplest way to accomplish this is simply a two dimensional array of counts indexed by the tokens of the two words in a pair. This was the approach used in early versions of the program, but it has the disadvantage of requiring too much memory for large files[5]

Instead of the array approach, we make use of the ternary search tree data structure again. This time, instead of storing words in the tree we store strings corresponding to the concatenation of the two tokens. Based on the results of the search in the tree, we manipulate a histogram data structure. Histograms are represented as a linked list of pairs of numbers: the number of objects in a histogram bin, and the (nonzero) number of bins in the histogram that have that many objects in them. With Zipf's law as a guide, we expect the great majority of possible word pairs to occur only one or not at all, so this data structure becomes very efficient. When calculating the entropies, instead of looping over all possible words or word pairs we only need to look at all the different occupancy numbers.

4.3 Implementing the entropy estimator

The majority of the program's run time is spent in the mutual information calculation. We estimate entropy by means of equation 3, a weighted average over the Wolpert-Wolf estimator (equation 1). As noted in [6], $\langle S \rangle$ can be evaluated without integration by simply using the value of β that maximizes the weight function ρ. However, in order to get the standard deviation of this estimate we need to evaluate$\langle S^2 \rangle$, which requires integration since we need to take error in estimating β into account.

[5]Example: for the Reuters corpus, we have $\sim 10^9$ possible word pairs; two bytes of storage for each would require several gigabytes of memory just for this table.

Using the Wolpert-Wolf estimator requires numerical evaluation of polygamma functions, and for this purpose we make use of the open-source Gnu Scientific Library [8]. The integral is evaluated numerically using the Romberg method, which requires fewer evaluations of the integrand than Simpson's rule-based methods [16]. We have three integrals to perform (the numerator and denominator of equation 3 for the entropy and its error) but because they include common weighting factors we save computational overhead by writing the routine to evaluate all three integrals simultaneously, rather than use one call for each.

Setting the limits of integration poses another difficulty. Formally, β should be integrated over $[0, \infty)$ — this could be accomplished numerically with a change of variables, but because of the nature of the weighting function ρ we anticipate that contributions to the integral from the narrow range of β where ρ is maximized. A narrow range of integration means that fewer evaluations of the integrand are needed, so we set the limits of integration by first locating the value of β that maximizes ρ using Brent's method [16] and then the values of β corresponding to the points where ρ is a specified number of e-foldings down from its peak using Ridder's method [16]. These are our limits of integration. We perform several checks to ensure that the peak in ρ is not too broad and that the range of integration is not set too narrowly, but these error conditions have not occurred in work with the actual data.

5 Results

5.1 Texts Used

The primary considerations guiding our selection were that the texts had to be long enough to reduce the error in our entropy estimator to a usable value, and that the text would be by one author or on a similar subject in order to produce long-range semantic correlations.

For our analysis, we used the Project Gutenberg e-text of *War and Peace*, by Leo Tolstoy [23]. *Remberance of Things Past* by Marcel Proust is longer, but as of this writing only the first two volumes are available electronically. The text contains 567,657 words with a vocabulary of 18,004 unique words.

We wanted to examine samples of text larger than this, so we also examined the Reuters corpus [17], a collection of $\approx 21,000$ short financial news articles that were carried by the Reuters news service in 1987. The Reuters corporation released these articles into the public domain in order to facilitate work in automatic document classification. Taken together the corpus is 2,478,116 words long, with a vocabulary of 40,696 unique words — many of which are proper names of companies, etc.

5.2 Plots

The essential idea of our results is conveyed by figure 5.2, a plot of mutual information between two words versus their separation in the text. We find that the mutual information appears to

decay as a power law, or rather as two power laws — for separations less than about five words, the mutual information decays quite steeply, but at long distances we lose less information. From the histogram plotted in the third panel, we can see that this crossover takes place on length scales on the order of a sentence.

We can perform a simple consistency check on the data in figure 5.2 by taking the original text of *War and Peace* and randomly reordering the words within it. By scrambling the word order in this way, we remove all correlations between word pairs that we consider meaningful. Additionally, we can choose to randomly reorder entire sentences (leaving the order of the words within intact) or only reorder words within a sentence. All four options are presented in figure 5.2.

One of the first conclusions evident from figure 5.2 is that the noise floor for this data set is ≈ 1.18 bits, much higher than one might assume from figure 5.2. This is troubling: why aren't our error bars this big? The most likely explanation seems to be the fact that both words in the pair are being chosen from the same (Zipf) distribution, which is highly non-uniform. We haven't explicitly included this fact in our construction of the estimator, so it assigns too much weight to the frequent pairs of relatively common words that cause the text to appear more correlated than it really is. Therefore, we think that we can regard the constant offset indicated by the red points as a systematic error while the error bars returned by the estimator are still relevant as measures of statistical error.

We can also perform the same calculation for the large Reuters corpus, which we show in figure 5.2. The notable feature of this plot is the fact that we apparently have a second break in the power law at separations of about 500 words, or roughly (we haven't gathered the statistics as of this writing) the length of a single news article.

6 Discussion

From the data we've presented, one can hypothesize that the power-law decay of mutual information is a generic feature of natural language and that we obtain breaks in the power law at length scales with semantic significance — i.e, the breaks in the power laws are manifestations of the hierarchical way language is organized. The rules of syntax are comparatively short ranged, so within the sentence we see a rapid decline of mutual information. At longer length scales, semantically relevant words (names of characters, etc.) can occur repeatedly over many sentences and indeed the entire corpus, so the mutual information decays more slowly. In the case of the Reuters corpus, we can expect an additional break at the scale of an article — words between articles are only related by the common theme of "business news in the late 1980s", while within a single article we expect words pertaining to the article's subject to occur frequently and increase the correlation on these scales.

There are many questions that need to be answered before we can claim that this hypothesis is any more than speculation. First and most concretely, it's not totally obvious that we really are observing power-law decays instead of a superposition of many exponential decays with varying length scales. This problem originates in the text itself, rather than our analysis: for texts of the length we've examined the statistical errors are small enough to permit discrimination between the

two types of behavior; on the other hand the distribution of sentence/paragraph/article lengths is quite broad and one could argue that we have many unimportant length scales in this situation instead of scale-free behavior.

The best way to resolve this problem is probably to consider a restricted version of natural language: one could construct a corpus of grammatically correct sentences of the same length and randomly order them in order to examine the power-law behavior at short distances. As figure 5.2 shows, though, an appreciable amount of information at short scales comes from inter-sentence correlations. In general it's not clear how to restrict this problem while still retaining all the features of natural language responsible for the observed behavior.

6.1 The information bottleneck method

Our hypothesis is essentially a statement about what parts of speech are responsible for carrying the relevant information at any given length scale — within a sentence the relevant information could be carried by something like the grammatical part of speech, while at larger scales the information is carried by words pertaining to the subject of the text. The mutual information between word pairs is much less than the upper bound set by the entropy of the distribution of words, so we should be able to encode all the information necessary to recover the behavior of the mutual information onto some new variable which has a lower entropy. We could construct this new variable by clustering: if many words are equivalent from the point of view of predicting the other word in a pair, we group them into a cluster and assign them the same symbol.

This is the idea behind information bottleneck compression [22, 20, 21]: one has a signal X which one would like to compress while preserving the information it provides about some other signal Y; in other words we want to compress X into some Z while minimizing the functional $I(Z;X)-\lambda I(Z;Y)$ for some value of the Lagrange multiplier λ. It turns out that this constraint yields a self consistent set of equations for the coding probabilities $\Pr(z|x)$ [22]; unfortunately, solving these equations to obtain an explicit compression algorithm is not so straightforward.

One could imagine a "simulated annealing" approach, where we begin at $\lambda = 0$ and all the words in one large cluster and then watch how the cluster splits as λ is raised. Detecting the splittings of clusters is difficult to do computationally [22]; alternatively, we can take the opposite route and start with each word as its own cluster and merge clusters together according to which step minimizes the mutual information functional [20]. Finally, we can obtain gains in computational performance and clustering accuracy over both methods if we first form a given number of clusters at random and exchange words between clusters in order to minimize the functional [21].

The information bottleneck idea provides a consistent extension of our methods in order to address this question of relevance. The main difficulty here is that the number of possible word pairs is quite large, so any of these algorithms becomes computationally expensive. In addition, the second approach is ruled out because of the large amount of degeneracy in our data: in accordance with Zipf's law, many words in a text occur only once, so many different merging steps yield the same minimization of the functional. In addition, we expect that meaningful encodings will involve of order 10 clusters, so we must perform on the order of 10^4 cluster merge steps before we obtain useful results.

The sequential algorithm [21] addresses these problems and appears quite feasible: one would compute the compression fidelity as a function of the number of clusters. If we see sharp drops in the fidelity after certain cluster arrangements (see [20] figure 2 for what we might expect) then ths implies that these clusterings are especially significant. Unfortunately, we were unable to carry out these calculations before the PACM deadline arrived.

7 Conclusions

The question we've considered in this work can be thought of as the logical extension of Zipf's experiment, in the sense that we switch from considering the distribution of words to correlations between words. It can also be seen as an extension of Shannon's results on the entropy of English text; instead of attempting to guess the second of a pair of letters we're in effect trying to guess the second of a pair of words. Our investigation was made possible by new Bayesian entropy estimation techniques [6]; at this point our results should be seen more as a demonstration of these methods than as making any definitive conclusions on the nature of language.

References

[1] Abney, S. Statistical methods and linguistics. In *The balancing act*, eds. J. L. Klavans and P. Resnik, Cambridge: MIT press, 1996.

[2] Bialek, W. Some notes on information theory, 2001. Unpublished work.

[3] Bialek, W. Notes on language and related issues, 2001. Unpublished work.

[4] Bialek, W., Callan, C. G., and Strong, S. P. Phys. Rev. Lett 77 (1996) 4693.

[5] Bialek, W., Nemenman, I., and Tishby, N. LANL preprint physics/0007070, 2001.

[6] Nemenman, I., Shafee, F. and Bialek, W. LANL preprint physics/0108025, 2002.

[7] Fano, R. M. *Transmission of information.* MIT press, 1961.

[8] GNU scientific libraries, v 1.1 accessed from `http://www.gnu.org/software/gsl/`.

[9] Li, W. J. Stat. Phys. 60 (1990) 823.

[10] McElice, R. J. *The theory of information and coding.* Addison-Wesley Publishing Company, 1977.

[11] Montemurro, M. and D. Zanette. LANL preprint cond-mat/0109218, 2001.

[12] Montemurro, M. and P. Pury. LANL preprint cond-mat/0201139, 2002.

[13] Nowak, M., N. Komarova and P. Niyogi. Science 291:114, 2001.

[14] Nowak, M., N. Komarova and P. Niyogi. Nature 417:611, 2002.

[15] Perline, R. Phys. Rev. E 54:220, 1996.

[16] Press, W, et al. Numerical Recipes in C. 2nd ed., Cambridge University Press, 1992

[17] Reuters-21578 corpus. Accessed from `http://about.reuters.com/researchandstandards/corpus/`.

[18] Sedgewick, R. Algorithms in C, 3rd Ed. Addison-Wesley, 1998.

[19] Shannon, C. E. and Weaver, W. *The mathematical theory of information*. Urbana: University of Illinois Press. 1949.

[20] Slonim, N. and N. Tishby. "Agglomerative Information Bottleneck", in Advances in Neural Information Processing Systems (NIPS) 12, pp. 617–623, 1999.

[21] Slonim, N., N. Friedman and N. Tishby. "Unsupervised Document Classification using Sequential Information Maximization", in Proc. of the 25th Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR), pp. 129–136, 2002.

[22] Tishby, N., F. Pereira and W. Bialek. LANL preprint physics/0004057, 2000.

[23] Tolstoy, *War and Peace*, electronic text accessed from `http://promo.net/pg/`.

[24] Wolpert, D. and Wolf, D. LANL preprint comp-gas/9403001, 1993.

[25] Wolpert, D. and Wolf, D. LANL preprint comp-gas/9403002, 1993.

[26] Wolpert, D. and Wolf, D. Phys. Rev. E 52 (1995) 6841.

[27] Zipf, G. *Human Behavior and the principle of least effort*. Addison-Wesley, 1949.

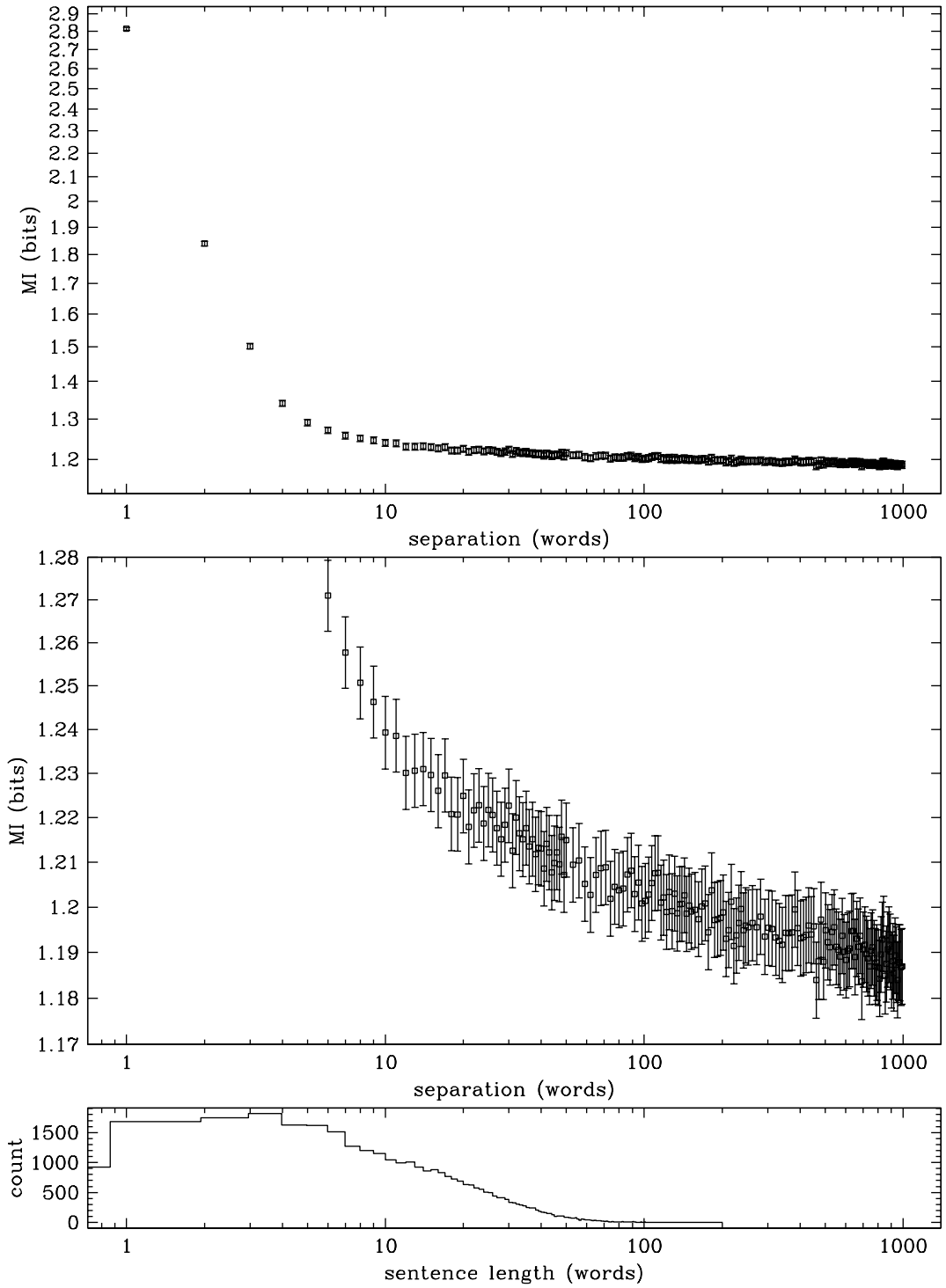

Figure 3: Plot of mutual information vs. word pair separation for *War and Peace*. The second panel plots the same information as the first on a magnified vertical scale in order to present the long distance information more clearly. The plotted error bars are obtained using equation 3 to estimate the second moment of the probability distribution of the mutual information.

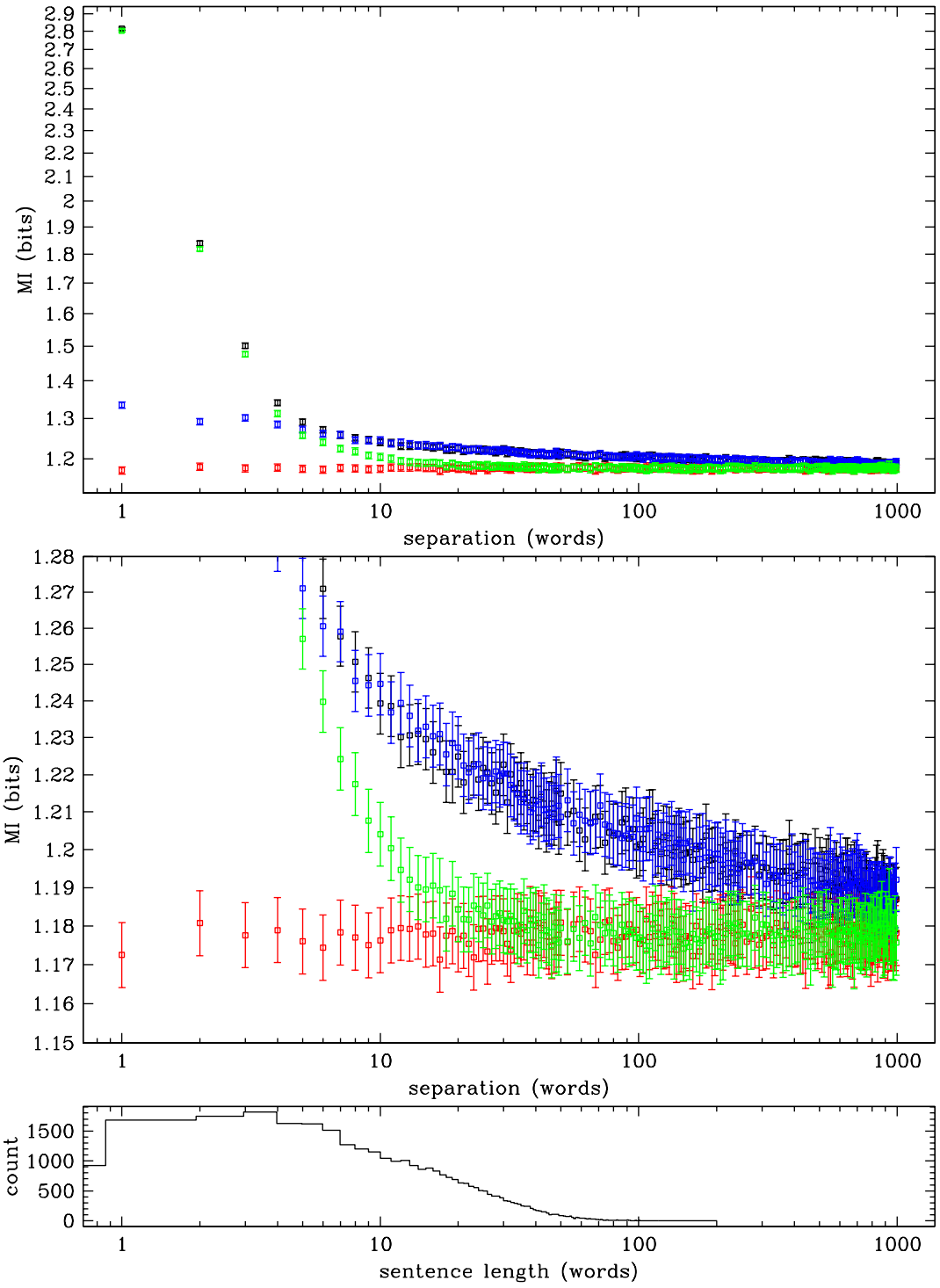

Figure 4: Plot of mutual information vs. word pair separation for *War and Peace*. Black points correspond to the original text while red points correspond to the completely scrambled text. The blue points show the result of scrambling words within a sentence, while the green points show the result of scrambling the order of sentences.

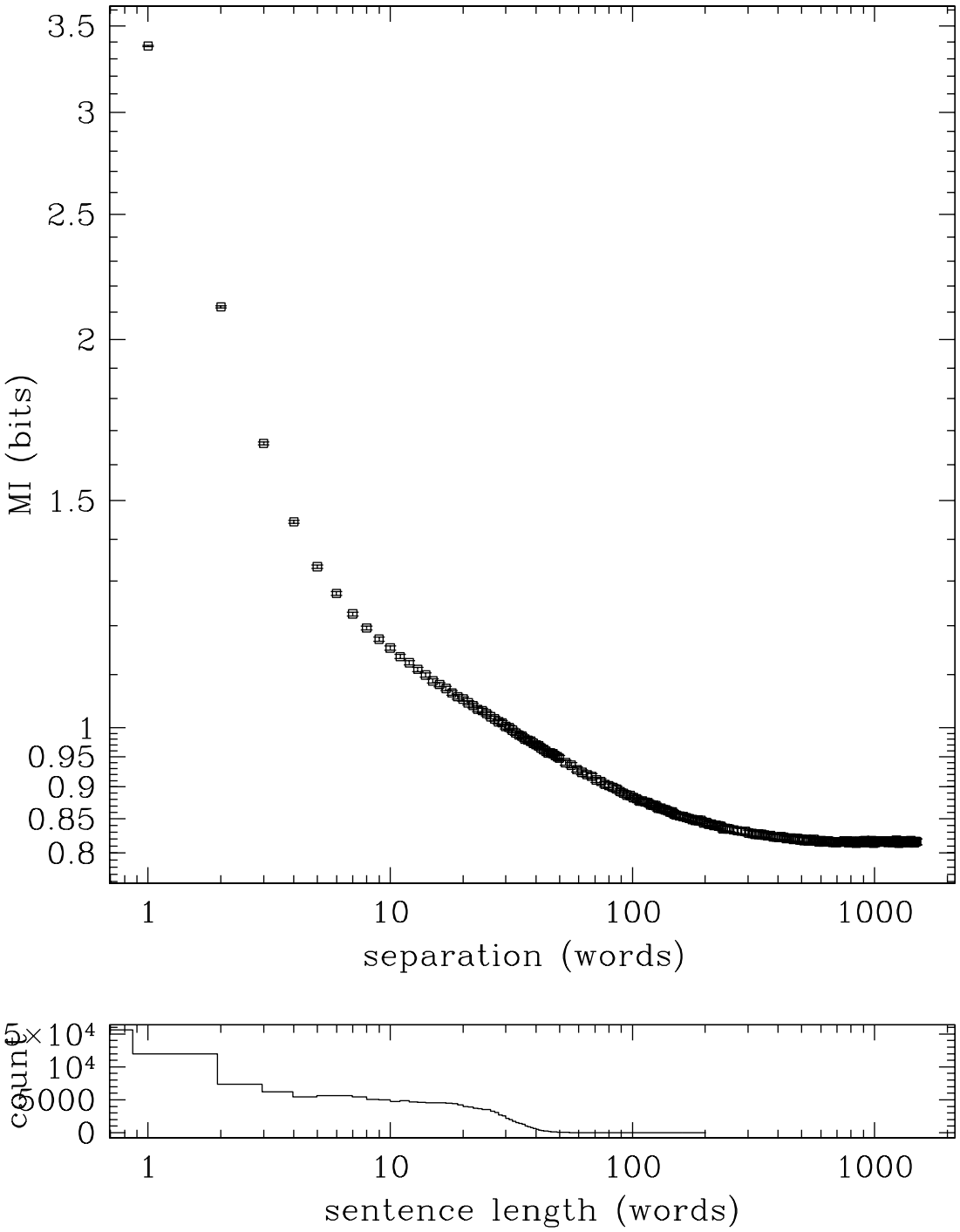

Figure 5: Plot of mutual information vs. word pair separation for the Reuters corpus.