Working Paper Series
ISSN 1170-487X

Adaptive Models of English Text

by W J Teahan and John G Cleary

Working Paper 97/30
December 1997

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Adaptive models of English text

W. J. TEAHAN AND JOHN G. CLEARY

b.teahan@cs.waikato.ac.nz, j.cleary@cs.waikato.ac.nz

Department of Computer Science, University of Waikato, Hamilton, New Zealand

Editor:

Abstract. High quality models of English text with performance approaching that of humans is important for many applications including spelling correction, speech recognition, OCR, and encryption. A number of different statistical models of English are compared with each other and with previous estimates from human subjects. It is concluded that the best current models are word based with part of speech tags. Given sufficient training text, they are able to attain performance comparable to humans.

Keywords: statistical language modeling, the entropy of English, text compression, adaptive n-gram models, PPM text compression scheme

1. Introduction

Statistical models of English text have a wide range of applications. Having a computer model that achieves close to a human's is critical in areas such as speech recognition, spell-checking, language identification and OCR (Teahan, 1997). It is also well-known in cryptography that removing redundancy is advisable prior to encryption to prevent statistical attacks (Wilson, 1994)—it is important that there are no models (human or otherwise) which are significantly better than the model used to remove the redundancy. The performance of statistical models can be evaluated by comparing their entropy against the entropy of their source.

Over 45 years ago, Claude E. Shannon estimated the entropy of English to be approximately 1 bit per character (Shannon, 1951). He did this by having human subjects guess samples of text, letter by letter. From the number of guesses made by each subject, he estimated upper and lower bounds of 1.3 and 0.6 bits per character ("bpc") for the entropy of English. Shannon's methodology was not improved upon until 1978 when Cover and King used a gambling approach to estimate the upper bound to be 1.25 bpc from the same text (Cover and King, 1978). Another estimate has come from the cryptographic community where an n-gram analysis suggests 1.5 bpc as the asymptotic limit for 26-letter English (Tilbourg, 1988).

On the surface there is a considerable margin between these estimates and the best computer based models. The best current models for English text have been the result of research in text compression. Not only is this an important application of such models but the amount of compression provides a measure of how well such models perform. For the best performed text compression scheme, PPM[1] (Cleary and Teahan, 1997, Moffat, 1990, Cleary and Witten, 1984), a result of around 2.4 bpc is often quoted based on Thomas Hardy's *Far from the Madding*

Crowd (Teahan and Cleary, 1996). These PPM results were obtained using initially empty context models. Similar results are achievable using a block-sort algorithm (Burrows and Wheeler, 1994) which has also been shown to be context based (Cleary and Teahan, 1997). In another experiment, Brown et al. used 583 million words of training text and a trigram based word model to obtain 1.75 bpc for the million word Brown Corpus.[2]

This paper will show that the difference between the best machine models and human models is smaller than might be indicated by these results. This follows from a number of observations: firstly, the original human experiments used only 27 character English (letters plus space) against full 128 character ASCII text for most computer experiments; secondly, using large amounts of priming text substantially improves performance; and thirdly, the PPM algorithm can be modified to perform better for English text. The result of this is machine performance down to 1.46 bpc.

This paper is organized as follows. A review of statistical language modeling is given first. Two measures for the performance of a language model are defined—"entropy" and "cross-entropy." The meaning of "the entropy of English" is discussed, and several estimates are reviewed. Character based language models based on the PPM text compression scheme are discussed next. Two techniques are found to significantly improve performance of these models: firstly, using training text to prime PPM's models improves compression by over 30%; and secondly, enlarging the alphabet leads to a further improvement of up to 7%. These improved models are used to estimate the entropy of English, on the same source text that Shannon used in his experiments, with compression down to 1.46 bpc. Two further class of models are then investigated—models based on words and models with auxiliary information in the form of parts of speech (tags). It is found that the tag based methods are comparable to word based methods, and generally outperform the PPM character based methods. A revised estimate of the entropy of English is then given, with the best result of 1.41 bpc being achieved by a tag based model.

2. Statistical language models

In a statistical (or probabilistic) model of language, the assumption is made that the symbols (e.g. words or characters) can be characterized by a set of conditional probabilities. A *language model* is a computer mechanism for determining these conditional probabilities. It assigns a probability to all possible sequences of symbols. The most successful types of language models are n-gram models (which base the probability of a word on the preceding n words) and n-pos models (which base the probability on the preceding words and parts of speech). n-graph models (models based on characters) have also been tried, although they do not feature as prominently in the literature as the other two classes of model. The probabilities for the models are estimated by collecting frequency statistics from a large corpus of text, called the *training text*. The size of the training text is often very large containing many millions (and in some cases billions) of words.

These models are sometimes referred to as "Markov models" because they are based on the assumption that language is a Markov source. An n-gram, n-pos or n-graph model is called an order $n-1$ Markov model (for example, a trigram model is an order 2 Markov model). The Markov models described here are based on a class of models identified by Jelinek (1990). A survey of the state of the art in human language technology describing various applications of these models is given in (Cole et al., 1995). Some other types of language models have been based on context-free grammars (Chen, 1996), decision trees (Bahl et al., 1989), local rules (Brill, 1993), collocational matrices (Garside et.al, 1987) and neural networks (Schmid, 1994).

*2.1. **n**-gram language models*

In an n-gram model, the probabilities are conditioned on the previous n words in the text. Formally, the probability of a sequence S, of m words, $w_1, w_2, \ldots, w_m$, is given by:

$$\begin{aligned} p(S) &= p(w_1)p(w_2|w_1)p(w_3|w_1,w_2)\ldots p(w_m|w_1,w_2,\ldots,w_{m-1}) \\ &= \prod_{i=1}^{m} p(w_i|w_1,w_2,\ldots,w_{i-1}). \end{aligned}$$

Here, w_i is called the *prediction* and $w_1, w_2, \ldots, w_{i-1}$ the *history*. n-gram language models make the assumption that the history is equivalent to the previous $n-1$ words (called the *conditioning context*). For example, bigram models make the following approximation:[3]

$$p(S) \approx \prod_{i=1}^{m} p(w_i|w_{i-1}).$$

Bigram and trigram models have been applied successfully to a wide range of domains especially machine translation and speech recognition (Bahl et al., 1983, Jelinek, 1990, Charniak, 1993, Chen, 1996).

*2.2. **n**-pos language models*

In an n-pos language model, the probabilities are conditioned on the parts of speech as well as the previous words. Since there are fewer types for parts of speech than words, this reduces the number of probabilities that must be estimated and therefore the amount of training data needed. Formally, the probability of a sequence S of m words $w_1, w_2, \ldots, w_m$ with tags $t_1, t_2, \ldots, t_m$ is given by:

$$p(S) = \prod_{i=1}^{m} p(t_i|t_1,t_2,\ldots,t_{i-1};w_1,w_2,\ldots,w_{i-1})\, p(w_i|t_1,t_2,\ldots,t_i;w_1,w_2,\ldots,w_{i-1}).$$

As with the n-gram models, this formula can be simplified by making a few assumptions which do not seem to drastically affect performance. For example, if we assume that the word is dependent on only its tag, and if we also assume that the current tag is independent of the previous words, and only dependent on the previous two tags, then we have:

$$p(S) \approx \prod_{i=1}^{m} p(t_i|t_{i-2}, t_{i-1})\ p(w_i|t_i).$$

That is, the word depends on two probabilities—the *contextual* probability which predicts the tag based on its two predecessors, $p(t_i|t_{i-1}, t_{i-2})$, and the *lexical* probability which predicts the word based on its tag, $p(w_i|t_i)$. This formula is the basis of the language models used in many applications, such as part of speech tagging (Charniak, 1993, Weischedel et al., 1993) and speech recognition (Jelinek, 1990, Kuhn and DeMori, 1990). Brown *et al.* (1992b) describe class-based language models that automatically assign classes to words based on a clustering process (with the word classes being substituted for the tags in the above formulas).

*2.3. **n**-graph language models*

In an n-graph model, the probability for the upcoming character is conditioned on the previous characters in the text. Thus, the probability of a sequence S of m characters $c_1, c_2, \ldots, c_m$ is given by:

$$p(S) = \prod_{i=1}^{m} p(c_i|c_1, c_2, \ldots, c_{i-1}).$$

Practical systems restrict the history to just a few of the previous characters; for example, character 6-graph models use the previous five characters:

$$p(S) \approx \prod_{i=1}^{m} p(c_i|c_{i-5}, c_{i-4}, c_{i-3}, c_{i-2}, c_{i-1}).$$

Character based language models have been most vigorously applied to the domain of text compression (Teahan and Cleary, 1996, Ristad and Thomas, 1995, Bell et al., 1990) but seem less popular in other application domains, some examples being cryptography (Irvine, 1997), language identification (Ganeson and Sherman, 1993) and various applications for automatically correcting words in texts such as OCR and spell-checking (Kukich, 1992). Their main advantage is that they are much more economical of memory space, and that they can be applied more easily to a wider range of problems in natural language processing such as cryptology, OCR and spell-checking (Teahan, 1997).

3. Entropy and cross-entropy

In information theory (Shannon, 1948), the fundamental coding theorem states that the lower bound to the average number of bits per symbol needed to encode a message (i.e., a sequence of text) is given by its *entropy*:

$$H(P) = -\sum_{i=1}^{k} p(x_i) \log p(x_i) \tag{1}$$

where the probabilities are independent and sum to 1 and where there are k possible symbols with probability distribution $P = p(x_1), p(x_2), \ldots, p(x_k)$. If the base of the logarithm is 2, then the unit of measurement is given in bits. The entropy is a measure of how much uncertainty is involved in the selection of a symbol—the greater the entropy, the greater the uncertainty. It can also be considered as a measure of the "information content" of the message—more probable messages convey less information than less probable ones. Various treatments of entropy are given in (Charniak, 1993, Brown et al., 1992a, Jelinek, 1990, Shannon, 1948) and these are drawn upon in the following discussion.

Applying this theory to language modeling, with word based models, we have a sequence of words $w_1, w_2, \ldots, w_m$, and Equation 1 can be reformulated as:

$$H(W) = -\sum p(w_1, w_2, \ldots, w_m) \log p(w_1, w_2, \ldots, w_m) \tag{2}$$

where the sum[4] is over all possible sequences $w_1, w_2, \ldots, w_m$ and the probability distribution is W.

The value for $H(W)$ gives us a measure for the difficulty of the language modeling task—higher values of $H(W)$ show that it is more difficult to predict the next word. Since the entropy increases with the length of the message, a better measure of the difficulty of the language modeling task is the *per-word entropy*:

$$\frac{1}{m} H(W) = -\frac{1}{m} \sum p(w_1, w_2, \ldots, w_m) \log p(w_1, w_2, \ldots, w_m). \tag{3}$$

Equation 1 can be extended to the more general case for a language with probability distribution L:

$$H(L) = \lim_{m \to \infty} -\frac{1}{m} \sum p(x_1, x_2, \ldots, x_m) \log p(x_1, x_2, \ldots, x_m). \tag{4}$$

This is called the *entropy of a language* and can be considered to be the limit of the entropy as the length of the message gets very large. For independent sources,

$$p(x_1, x_2, \ldots, x_m) = p(x_1)p(x_2) \ldots p(x_m) \tag{5}$$

and Equation 4 reduces to Equation 1. If the process generating the language is *ergodic* (i.e., sufficiently long sequences of symbols are typical of it, and can be used to deduce its statistical structure), then this formula reduces to the Shannon-McMillan-Breiman theorem (Brown et al., 1992a):

$$H(L) = \lim_{m \to \infty} -\frac{1}{m} \log p(x_1, x_2, \ldots, x_m). \tag{6}$$

Usually the true probability distribution L is not known. However, an upper bound to $H(L)$ can still be obtained by using a model M as an approximation to L:

$$H(L, M) = -\sum p_M(x_1, x_2, \ldots, x_m) \log p_M(x_1, x_2, \ldots, x_m) \tag{7}$$

where the correct model is $p(x_1, x_2, \ldots, x_m)$ and the probabilities are estimated by the model $p_M(x_1, x_2, \ldots, x_m)$. $H(L, M)$ is called the *cross-entropy*, and is always greater than or equal to the entropy $H(L)$, as this is based on the best possible language model, the source itself:

$$H(L) \leq H(L, M). \tag{8}$$

The *cross-entropy of a language* L given model M is defined by taking the limit as the length of the message grows (as for Equation 4). If L is ergodic, we have:

$$H(L, M) = \lim_{m \to \infty} -\frac{1}{m} \log p_M(x_1, x_2, \ldots, x_m). \tag{9}$$

Combining Equations 4 and 8, we see that:

$$H(L) \leq \lim_{m \to \infty} -\frac{1}{m} \log p_M(x_1, x_2, \ldots, x_m). \tag{10}$$

The cross-entropy is relevant as it provides a measure of how well the approximate model is doing—the closer it is to $H(L)$, the more accurate the model. It gives a useful yardstick for comparing the accuracy of competing models. This is done by computing the cross-entropy for each model, and the model with the smallest cross-entropy is inferred to be the "best".

Entropy and cross-entropy can be directly related to text compression. The entropy $H(L)$ is a lower bound on the average number of bits per symbol required to encode a long string of text drawn from L (Brown et al., 1992a). Bell et al. show that arithmetic coding (Witten et al., 1987, Rissanen and Langdon, 1979), a method of assigning codes to symbols with a known probability distribution, achieves an average code length arbitrarily close to the cross-entropy (Bell et al., 1990). Hence, text compression can be used directly to estimate an upper bound to the entropy: if $b_M(x_1 x_2 \ldots x_m)$ is the number of bits required to encode the string $x_1 x_2 \ldots x_m$ using model M, then

$$H(L, M) = \lim_{m \to \infty} \frac{1}{m} b_M(x_1 x_2 \ldots x_m) \tag{11}$$

where $H(L, M)$ is the number of bits per symbol required to encode a long sequence of text drawn from L.

4. The entropy of English

In a classic paper published in 1951, Shannon estimated the entropy of English[5] to be approximately 1 bit per character (Shannon, 1951). Shannon's method of estimating the entropy was to have human subjects guess upcoming characters based on the immediately preceding text. He chose 100 random samples taken from Dumas Malone's *Jefferson the Virginian* (Malone, 1948). From the number of guesses made by each subject, Shannon derived upper and lower bound estimates of 1.3 and 0.6 bits per character ("bpc").

Cover and King (1978) noted that Shannon's guessing procedure only gave partial information about the probabilities for the upcoming symbol. A correct guess only tells us which symbol a subject believes is the most probable, and not how much more probable it is than other symbols. They developed a gambling approach where each subject gambled a proportion of their current capital on the next symbol. Using a weighted average over all the subjects' betting schemes,[6] they were able to derive an upper bound of 1.25 bpc. The performance of individual subjects ranged from 1.29 bpc to 1.90 bpc.

Cover and King discuss the meaning of the phrase "the entropy of English": "It should be realized that English is generated by many sources, and each source has its own characteristic entropy. The operational meaning of entropy is clear. It is the minimum expected number of bits/symbol necessary for the characterization of the text." They also point out that although the true entropy is strictly less than the upper bound, the lower bound is really a "lower bound to an upper bound" and is therefore of limited meaning.

Both Shannon's and Cover and King's approaches were based on human subjects guessing a text. Other approaches have been based on the statistical analysis of letter and word frequencies derived from text, and on the performance of computer models using text compression algorithms. English entropy estimates obtained by various researchers for these different approaches are summarized in Table 1. The estimates are shown in the first column, with a reference and description of the model used given in the remaining two columns. (See Cover and King's paper for an extensive bibliography on related research.)

Shannon also made estimates based on the statistical analysis of letter and word frequency data available at the time. He showed that, as longer and longer sequences of characters are considered, the estimate of the upper bound decreases from 4.03 bpc for unigraphs down to 3.1 bpc for trigraphs (see Table 1). His estimate based on unigrams (words) was slightly better at 2.14 bpc. The amount of statistical data available to Shannon was limited, so he was unable to extend his analysis to higher order models. In a more recent report, statistical analysis for n-graphs suggests 1.5 bpc as the asymptotic limit for 26-letter English (Tilbourg, 1988).

Table 1 also lists estimates from a progression of successively more compact computer coding schemes. (This is based on a table in (Church and Mercer, 1993).) The standard method for representing ASCII text is to use 8 bits to encode each character. Church and Mercer state that this would be a perfect code if the source produced each symbol equally often and independently of context; however, as

Table 1. English entropy estimates using different language models

Entropy (bpc)	Reference	Model
	Models based on human performance:	
1.3	(Shannon, 1951)	From experiments with subjects' best guesses
1.25	(Cover and King, 1978)	From experiments with subjects using gambling
	Models based on statistical analysis of text:	
4.03	(Shannon, 1951)	Unigraphs
3.32	(Shannon, 1951)	Bigraphs
3.1	(Shannon, 1951)	Trigraphs
2.14	(Shannon, 1951)	Unigrams (words)
1.5	(Tilbourg, 1988)	n-graphs
	Models based on computer performance:	
8	(Church and Mercer, 1993)	ASCII
5	(Church and Mercer, 1993)	Huffman code each character
4.06	(Bell et al., 1990)	Unix `compress`
2.48	(Bell et al., 1990)	PPMC (maximum order 3)
2.3	(Witten, Moffat and Bell, 1994)	PPMC (maximum order 5)
2.1	(Church and Mercer, 1993)	Unigram (Huffman code each word)
1.75	(Brown et al., 1992a)	Trigrams (words)

they point out, "English is not like this" (Church and Mercer, 1993). For English, the average code length can be reduced substantially by assigning shorter codes to frequently occurring symbols and longer codes to less frequent ones. One way of doing this is to use a Huffman code (Bell et al., 1990). The Unix `compress` algorithm based on the Lempel-Ziv compression scheme (Ziv and Lempel, 1978) gives slightly better compression. Bell, Cleary and Witten compare the performance of various text compression algorithms at compressing different texts (Bell et al., 1990), and the best performed of the algorithms for English text is PPMC (Moffat, 1990), an implementation of the PPM text compression scheme (described in Section 5). An order 0 word model using Huffman coding achieves slightly better performance, but the best computer-based result of 1.75 bpc reported to date is by Brown et al. (1992a) who adopted the word trigram model developed for speech recognition. They constructed a model from 583 million words of training text, and then used that to estimate the entropy on the 1 million word Brown Corpus.

A cautionary note should be made concerning these results. There is a disparity in the alphabet size and source texts used for the different experiments which makes it difficult to compare the performance of the different approaches. The statistical and human based experiments were based on an alphabet of 27 characters (the 26 letters plus space),[7] whereas all the computer-based experiments were based on the full ASCII character set (256 characters). The entropy also varies widely depending on the style, subject, period and authorship of the text being examined. For this reason, the same source text should be used if at all possible to make a fair comparison. However, the source texts used for the different computer-based experiments vary widely, although all are based on large samples of text. In comparison, the human estimates were based on samples of at most a few hundred letters. Jelinek

(1990, page 476) highlights problems with Shannon's methodology because of this: "His results would be valid only if the text they were based on was representative and large enough for his statistical estimates to converge. Representativity of English can never be established, and the text passage Shannon chose was much too short."

5. PPM based language models

The best algorithms for lossless compression of text are those which adapt to the text being compressed (Bell et al., 1990). Two classes of adaptive techniques are commonly used. One class matches the text against a dictionary of strings seen so far and transforms it into a list of indices into the dictionary. These techniques are usually formulated as a variant on Ziv-Lempel (LZ) compression (Ziv and Lempel, 1977, Ziv and Lempel, 1978). While LZ compressors do not give the best compression they are widely used because of their simplicity and low execution overhead.

The best compression is obtained by another class of compressors which use adaptive statistical modeling. These split compression into two steps. The first step accumulates a statistical model of the symbols seen so far in the input text. As each symbol is encoded, this model is used to generate a probability distribution over those symbols which can occur next. Arithmetic coding is then used to optimally encode the symbol which actually does occur with respect to this distribution (Bell et al., 1990).

Prediction by partial matching, or PPM, has set the performance standard in lossless compression of text throughout the past decade. The original algorithm was first published in 1984 by Cleary and Witten (1984), and a series of improvements was described by Moffat (1990), culminating in a careful implementation, called PPMC, which has become the benchmark version. This still achieves results superior to virtually all other compression methods, despite many attempts to better it (Cleary and Teahan, 1997, Bunton, 1996).

PPM is an n-graph model which uses the last few characters in the input stream to predict the upcoming one (see Section 2.3). It employs a suite of "finite context" models of order k (where k is the number of preceding symbols used) up to some pre-determined maximum. For each model, note is kept of all characters that have followed every length-k subsequence observed so far in the input, and the number of times that each has occurred. Prediction probabilities are calculated from these counts. The probabilities associated with each character that has followed the last k characters in the past are used to predict the upcoming character. Thus from each model, a separate predicted probability distribution is obtained.

These distributions are effectively combined into a single one, and arithmetic coding is used to encode the character that actually occurs, relative to that distribution. The combination is achieved through the use of "escape" probabilities. Recall that each model has a different value of k. The model with the largest k is, by default, the one used for coding. However, if a novel character is encountered in this context, which means that the context cannot be used for encoding it, an

Table 2. PPMC model after processing the string *abracadabra* (maximum order 2)

Order $k=2$					Order $k=1$					Order $k=0$				Order $k=-1$			
Predictions			c	p	Predictions			c	p	Predictions			c	p	Predictions		c	p
ab	→	r	2	$\frac{2}{3}$	a	→	b	2	$\frac{2}{7}$	→	a	5	$\frac{5}{16}$	→ A		1	$\frac{1}{\lvert A\rvert}$
	→	*Esc*	1	$\frac{1}{3}$		→	c	1	$\frac{1}{7}$	→	b	2	$\frac{2}{16}$				
						→	d	1	$\frac{1}{7}$	→	c	1	$\frac{1}{16}$				
ac	→	a	1	$\frac{1}{2}$		→	*Esc*	3	$\frac{3}{7}$	→	d	1	$\frac{1}{16}$				
	→	*Esc*	1	$\frac{1}{2}$						→	r	2	$\frac{2}{16}$				
					b	→	r	2	$\frac{2}{3}$	→	*Esc*	5	$\frac{5}{16}$				
ad	→	a	1	$\frac{1}{2}$		→	*Esc*	1	$\frac{1}{3}$								
	→	*Esc*	1	$\frac{1}{2}$													
					c	→	a	1	$\frac{1}{2}$								
br	→	a	2	$\frac{2}{3}$		→	*Esc*	1	$\frac{1}{2}$								
	→	*Esc*	1	$\frac{1}{3}$													
					d	→	a	1	$\frac{1}{2}$								
ca	→	d	1	$\frac{1}{2}$		→	*Esc*	1	$\frac{1}{2}$								
	→	*Esc*	1	$\frac{1}{2}$													
					r	→	a	2	$\frac{1}{3}$								
da	→	b	1	$\frac{1}{2}$		→	*Esc*	1	$\frac{1}{3}$								
	→	*Esc*	1	$\frac{1}{2}$													
ra	→	c	1	$\frac{1}{2}$													
	→	*Esc*	1	$\frac{1}{2}$													

escape symbol is transmitted to signal the decoder to switch to the model with the next smaller value of k. The process continues until a model is reached in which the character is not novel, at which point it is encoded with respect to the distribution predicted by that model. To ensure that the process terminates, a model is assumed to be present below the lowest level, containing all characters in the coding alphabet. This mechanism effectively blends the different order models together in a proportion that depends on the values actually used for escape probabilities. Formally, given $S = c_1, c_2, \ldots, c_m$, then

$$p(S) = \prod_{i=1}^{m} p'(c_i | c_{i-5}, c_{i-4}, c_{i-3}, c_{i-2}, c_{i-1})$$

where p' gives the probabilities returned by the order 5 PPM character model.

As an illustration of the operation of PPM, Table 2 shows the state of the four models with $k = 2$, 1, 0, and -1 after the input string *abracadabra* has been processed. For each model, all previously-occurring contexts are shown with their associated predictions, along with occurrence counts c and the probabilities p that are calculated from them. By convention, $k = -1$ designates the bottom-level model that predicts all characters equally; it gives them each probability $\frac{1}{|A|}$ where A is the alphabet used.

Some policy must be adopted for choosing the probabilities to be associated with the escape events. There is no sound theoretical basis for any particular choice in

Table 3. Encodings for three sample characters using the model in Table 2

character	probabilities encoded (without exclusions)	probabilities encoded (with exclusions)	codespace occupied
c	$\frac{1}{2}$	$\frac{1}{2}$	$-\log_2 \frac{1}{2} = 1$ bit
d	$\frac{1}{2}, \frac{1}{7}$	$\frac{1}{2}, \frac{1}{6}$	$-\log_2(\frac{1}{2} \cdot \frac{1}{6}) = 3.6$ bits
t	$\frac{1}{2}, \frac{3}{7}, \frac{5}{16}, \frac{1}{\|A\|}$	$\frac{1}{2}, \frac{3}{6}, \frac{5}{12}, \frac{1}{\|A\|-5}$	$-\log_2(\frac{1}{2} \cdot \frac{3}{6} \cdot \frac{5}{12} \cdot \frac{1}{251}) = 11.2$ bits

the absence of some *a priori* assumption on the nature of the symbol source; some alternatives are evaluated in (Witten and Bell, 1991, Bunton, 1996). The method used in the example, commonly called "Method C," gives a count to the escape event equal to the number of different symbols that have been seen in the context so far (Moffat, 1990); thus, for example, in the order-0 column of Table 2 the escape symbol receives a count of 5 because five different symbols have been seen in that context. Other methods for estimating the escape probability are described in Teahan (1997).

Sample encodings using these models are shown in Table 3. As noted above, prediction proceeds from the highest-order model ($k = 2$). If the context successfully predicts the next character in the input sequence, the associated probability p is used to encode it. For example, if c followed the string *abracadabra*, the prediction *ra*→*c* would be used to encode it with a probability of $\frac{1}{2}$; that is, in one bit.

Suppose instead that the character following *abracadabra* was *d*. This is not predicted from the current $k = 2$ context *ra*. Consequently, an escape event occurs in context *ra*, which is coded with a probability of $\frac{1}{2}$, and then the $k = 1$ context *a* is used. This does predict the desired symbol through the prediction *a*→*d*, with probability $\frac{1}{7}$. In fact, a more accurate estimate of the prediction probability in this context is obtained by noting that the character *c* cannot possibly occur, since if it did it would have been encoded at the $k = 2$ level. This mechanism, called "exclusion," corrects the probability to $\frac{1}{6}$ as shown in the third column of Table 3. Finally, the total number of bits needed to encode the *d* can be calculated to be 3.6.

If the next character were one that had never been encountered before, say *t*, escaping would take place repeatedly right down to the base level $k = -1$. Once this level is reached, all symbols are equiprobable—except that, through the exclusion device, there is no need to reserve probability space for symbols that already appear at higher levels. Assuming a 256-character alphabet, the *t* is coded with probability $\frac{1}{251}$ at the base level, leading to a total requirement of 11.2 bits including those needed to specify the three escapes. This example highlights how the probabilities allocated to different symbols can be either arbitrarily small or large. In comparison, an order 0 static model that assigns the same probability $\frac{1}{256}$ to each character in the alphabet requires 8 bits to encode each of them.

6. Estimating the entropy of English using PPM based models

A series of carefully crafted improvements have been made to the PPMC implementation, culminating in an 8% improvement for the latest implementation (Teahan, 1997). These implementations use a fixed upper bound to the length of the context models. Another approach is to use unbounded length context models (Cleary and Teahan, 1997). Burrows and Wheeler's block-sorting algorithm (1994) also uses unbounded contexts but unlike the PPM methods is non-adaptive. Experiments based on the Calgary Corpus (Bell et .al, 1990) show that for current implementations, performance of these approaches are similar (Fenwick, 1996, Teahan, 1997). However, for English text fixed length PPM models of order 5 perform up to 5% better.

These improvements raise the question of how much further the PPM models can be improved and whether the margin to the human entropy estimates can be closed. The first step towards this is to note that the human estimates were done using 27 character English (letters plus space) whereas the machine models are for full ASCII text. A simple experiment of converting text to this format (by replacing all sequences of non-letters with a single space, and converting all letters to lower case) show a 10% improvement in compression. Also, the text that Shannon used in his experiments, Dumas Malone's *Jefferson the Virginian*, is not the same as those in the computer experiments, the Brown Corpus (used in Brown *et al.*'s (1992a) experiments) and Thomas Hardy's book (*book1* in the Calgary Corpus). Indeed, the latter contains numerous typographical errors which have been removed in the latest electronic text available from Project Gutenberg. This corrected version improves compression by 3% to 2.24 bpc. In comparison, the entropy for the text Shannon used is in fact a great deal lower, requiring just 2.01 bpc to compress the entire volume (Teahan and Cleary, 1996).

These musings led us to investigate ways in which PPM's performance could be improved for English text. Two such methods are described in the next two sections.

6.1. The use of training text to improve PPM models

One of the drawbacks with PPM is that it performs relatively poorly at the start. This is because it has not yet built up the counts for the higher order context models, so must resort to lower order models. To overcome this, a simple expedient is to use training text to prime PPM.

This raises the question of which and how much training text to use. Obviously, finding training text that is related to the text being compressed is important, preferably by the same author. Boggess et al. (1991) highlight some of the problems: "we have noted that the body of work of a single writer differs significantly both from other writers and from norms for English text derived from large, multiple-source corpora."

One approach would be to find the greatest amount of text written by the same author, preferably of the same style, and about the same subject, and use this for

Table 4. Dumas Malone's epic work *Jefferson and His Time*

vol.	pub. date	no. of pages	no. of words	no. of chars.	title
1	1948	423	154035	905790	*Jefferson the Virginian*
2	1951	488	182633	1081029	*Jefferson and the Rights of Man*
3	1962	506	186852	1114406	*Jefferson and the Ordeal of Liberty*
4	1970	485	175441	1058016	*Jefferson the President First Term 1801-1805*
5	1974	668	235622	1418505	*Jefferson the President Second Term 1805-1809*
6	1977	499	174940	1034505	*The Sage of Monticello*
Total		3069	1109523	6612253	*Jefferson and His Time*

training text. The complete works of Jane Austen is now available in the public domain: the six novels *Emma*, *Mansfield Park*, *Northanger Abbey*, *Persuasion*, *Pride and Prejudice* and *Sense and Sensibility*. Their total combined size is over 4 million characters or nearly 720,000 words. Experiments with the novel *Emma* show that compressing the last chapter using the remaining chapters plus the other five novels as training text improves compression by nearly 47% from 2.93 bpc down to 1.56 bpc. This can be further improved down to 1.48 bpc using the methods described in the next section.

However, there is a danger in all this. This text is not the same as that used by Shannon. Shannon himself found that the entropy was greater for different texts such as newspaper writing, scientific work and poetry. Indeed, any comparison made between Brown et al.'s result, which is based on the Brown Corpus, and Shannon's results is circumspect because of the different source texts involved. Still, the results with *Emma* were very promising, and gave us enough encouragement to try to repeat them with Shannon's source *Jefferson the Virginian*. This text is the first volume in a series of six titled *Jefferson and His Time* (Malone, 1977) which was awarded the Pulitzer Prize for History in 1975. The first volume, *Jefferson the Virginian*, was published in 1948, three years before Shannon published his paper. Fortuitously (at least for our experiments), Dumas Malone then proceeded to write and publish the remaining five volumes over a period of 26 years.

In order to gain access to these one million words of text, all six volumes were scanned into the computer. Numerous errors made by the OCR software were corrected, and all footnotes, headings, page numbers and end-of-line hyphens were removed. Table 4 summarizes information about the resulting text (which we call the "Jefferson Corpus"). We estimate that the percentage of incorrect words remaining is about 0.2% based upon the number of errors found in the last chapter of the first volume. Our experiments with this text shows that over 30% improvement in compression is possible using the other five volumes as training text for the first.

The question of how much training text should be used is an important one. Brown et al. in their experiments with trigram models used a training corpus containing over 580 million words. In comparison, the training text we use for our method is just 1.1 million words for *Jefferson the Virginian* or 0.7 million words for

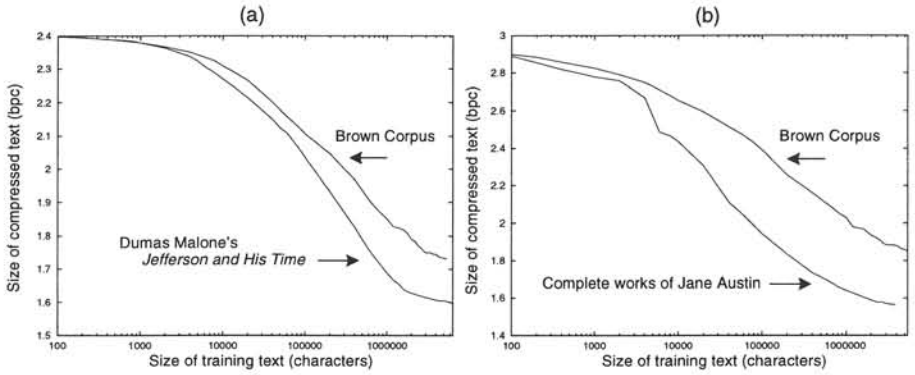

Figure 1. How training text improves compression for PPM (a) for the last chapter of Dumas Malone's *Jefferson the Virginian* (b) for the last chapter of Jane Austen's *Emma*

Emma, although admittedly, this training text is far better tuned to the text being modeled. Experiments using training text from different authors show a marked reduction in performance, and a substantial increase in the size of the training text would be required to overcome this. This is shown in Figure 1 which illustrates how the compression of the last chapters of *Jefferson the Virginian* and *Emma* improves for different sized training texts taken from different sources, in this case the Brown Corpus, and the remaining text from *Jefferson and His Time* and the complete works of Jane Austen.

Clearly, related text performs as a better training vehicle although it seems that large amounts of unrelated text can do as well. All the curves show a quantitatively similar shape with a steep reduction between 10^4 and 10^6 characters and rather flatter beyond this. It is unclear whether this is a general result and what performance might be expected for substantially larger training texts, for example, the corpus used by Brown et al.

These results pose two related questions: "how many words does it take to 'train' a human model?" and "how many words has a human encountered in their lifetime?" Siskind estimates from language learning research that children hear between $3,200$ and $50,400$ words per day (Siskind, 1995) . Assuming a constant rate, then a 30 year-old listener will have heard between 35 and 550 million words in their lifetime, values beyond those used in our experiments but comparable to the corpus used by Brown et al. It is unclear to what extent the difference between spoken and written English would affect human performance.

6.2. Improving the PPM model by enlarging the alphabet

In English, some bigraphs occur more frequently than individual letters. This led us to investigate the effect of replacing such frequent bigraphs by new unique symbols. For example, one of the most frequent bigraphs in English is *th*. Performance of the

Table 5. The 110 most frequent non-space bigraphs from the Brown Corpus

th	he	in	er	an	re	on	at	en	nd
ed	or	es	te	ti	it	is	to	st	ar
of	ng	ha	ou	al	as	nt	hi	le	se
ve	me	ne	co	ea	de	ro	ri	io	ll
ic	li	ra	be	ce	ch	ma	om	el	ho
si	wa	ca	la	ur	ta	fo	ly	no	pe
il	us	ut	di	ns	ad	ac	et	ot	ee
rs	un	ec	wh	wi	so	pr	lo	we	tr
ss	ge	nc	ct	sh	ow	ol	ai	em	ie
mo	ul	id	rt	im	ni	po	ir	ld	mi
ts	pa	na	ke	am	oo	os	gh	su	ig

PPM encoder improves by up to 1% if all occurrences of *th* in the text are replaced with a single uniquely identifiable character. Curiously, a look at the history of English spelling (Scragg, 1974) helps us understand why. In Old English, *th* was written as two single letters—the thorn and the eth, corresponding to the unvoiced and voiced sounds in *that* and *thin*. Caxton, the first English printer, sometimes used thorn's nearest typographical equivalent *Y* as there was no equivalent letter in the continental type used then by the printing presses. This may still be seen today in old-style signs such as *Ye Olde Gifte Shoppe*.

To encode text using this idea, it is scanned from left to right, with the upcoming bigraph replaced by a unique symbol if it occurs in a lookup table. This presents two immediate problems: which bigraphs are the most frequent, and how many bigraphs should be replaced? Both problems can be solved by using frequency counts gleaned from a large corpus of representative English text (e.g. the Brown Corpus) to select the N most frequent bigraphs, with N being adjusted to find the maximum compression (typically when $N \approx 100$).

Experimental results show that replacing bigraphs containing spaces degrades performance. This is similar to the effect that removing spaces from English text has on compressed output size – the compressed output actually increases even though as much as 15% of the text may have been removed! Consequently, the bigraph encoding method replaces only the most frequent bigraphs that do not contain spaces. For example, the three most frequent non-space bigraphs are *th*, *he* and *in*. Table 5 is the list of bigraphs we used in our experiments (sorted in decreasing order of frequency from left to right).

For example, when the string "*the rights of man*" is bigraph-encoded, it reduces from 17 characters down to 11 symbols: *th*, *ri*, *gh*, *ts*, *of* and *ma* are all replaced by separate symbols in the expanded alphabet. Note that the bigraph *he* in the word *the* is not replaced, despite it being the second most frequently occurring non-space bigraph in the Brown Corpus. This is because the bigraph *th* just before it is replaced first.

In compression experiments on a diverse range of English text, we have found that bigraph encoding typically improves PPM compression on English text by up to 7% (Teahan and Cleary, 1996). We have also experimented with other alphabet

Table 6. Compression ratios for the last chapter of *Jefferson the Virginian*

encoding method	untrained PPM (bpc)	trained PPM (bpc)	improvement (%)
unencoded text	2.402	1.598	33.5
bigraph encoding	2.415	1.488	38.4

Table 7. Compression ratios for various texts using untrained PPM

source text	size of text (chars)	PPM (bpc)	bigraph encoding (bpc)	improvement (%)
Jefferson Corpus	6448777	1.597	1.513	5.26
Brown Corpus	5766822	1.938	1.874	3.30
LOB Corpus	5636660	1.911	1.860	2.67
Wall Street Journal	15398849	1.725	1.602	7.13
King James Bible	4139727	1.574	1.464	7.04
Complete works of Shakespeare	4607009	1.914	1.889	1.31
Complete works of Jane Austen	3872447	1.659	1.603	3.36

replacement algorithms (such as replacement of commonly occurring vowel, consonant and word sequences) but in all cases, the best improvement in compression occurs with the simple bigraph replacement method.

6.3. An estimate of the entropy of English

The performance of bigraph encoding on the last chapter of Dumas Malone's *Jefferson the Virginian* is compared with PPM on the unencoded text in Table 6. The table lists the compression ratios for both trained and untrained PPM (using fixed models of order 5). The text was pre-converted to 27 character English for these experiments, and consists of 7985 words and 46137 characters. The results in the table graphically illustrate the importance of using training text to pre-load PPM's context models, with an improvement of over 30%. The training text used in this case was the remaining 5 volumes of Dumas Malone's work, plus all but the last chapter of the first volume, containing 1.1 million words or 6.4 million characters.

To further test the robustness of bigraph encoding, a number of other 27 character texts[8] were bigraph encoded and compressed using untrained PPM. Table 7 shows that in all cases there was an improvement over unencoded text of 2% to 7%.

Shannon for his experiments used 100 samples each containing 100 characters taken at random from the text. However, his paper does not indicate which passages were chosen so we cannot make a direct comparison with this text. Cover and King instead chose a passage of 1490 characters (260 words) to predict the next 75 characters. Using trained PPM with bigraph encoding on this short piece of text

Table 8. A collection of writings by Thomas Jefferson

title	no. of words	no. of chars.
A Summary View of the Rights of British America	7012	41177
Addresses, Messages, and Replies	20874	126942
Autobiography	40902	239219
First Inaugural Address	1724	10332
Indian Addresses	5847	31559
Letters	310445	1785007
Miscellany	50043	290524
Notes on the State of Virginia	66050	404735
Public Papers	58409	348392
Second Inaugural Address	2164	13034
Total	563470	3290921

results in an estimate of 1.726 bpc. In comparison, individual subjects in Cover and King's experiments obtained between 1.29 and 1.90 bpc.

This compares with 1.488 bpc for our larger sample of 46137 characters taken from the last chapter. This result improves further if additional related training text is available. For example, the collection of writings by Thomas Jefferson shown in Table 8 is now available in the public domain. The addition of this collection to the training text improves the estimate down to 1.461 bpc. In comparison, unrelated text does not do as well, with the addition of the entire Brown Corpus (almost doubling the size of the training text) only improving the estimate down to 1.482 bpc.

7. Word and tag based language models

The problem of compressing English text differs from the general compression problem because much is known *a priori* about the structure of English. It should be possible to use this structure to achieve better compression. One way that this has been done is to use the fact that English can be segmented into words and to use words rather than characters as the fundamental unit of compression. This is found to be faster (because less encoding operations are necessary) and to achieve up to 4% better compression than purely character based models (Teahan, 1997).

Another approach to modeling we adopt (Teahan and Cleary, 1997) is to use parts-of-speech (tags) such as *noun*, *verb*, and *adjective*. The idea is that knowing the tag of a word helps in predicting it. The advantage of using the tag is that it may have occurred many times previously. Hence, a good representative sample of what is likely to follow it has been built up. By contrast, an individual word may have occurred only a small number of times. Traditional language modeling approaches (for example, used in speech recognition and machine translation) have been either word or part of speech based (Brown et al., 1992b, Jelinek, 1990,

Kuhn and DeMori, 1990). Results with these models have shown that the word based approach generally performs better.

This section describes two adaptive models, one word based and the other tag based, that have been found to perform better than other models in practice. Results of experiments with compressing English text using these models are discussed in the next section. These are split into two subsections—results with manually tagged texts, and results with texts automatically tagged by computer. A revised estimate of the entropy of English using these models is then given.

There are two major issues with using tags: first, the words in the text must have tags assigned to them somehow; and second, the tags need to be encoded in the models along with the text itself. This has the potential for increasing the size of the compressed text. However, the extra contextual information provided by the tags more than compensates for this and we will see that the total result is slightly better than pure word based coding.

For the models we are concerned with, we assume that the text has already been tagged (for example, using the tag sets shown in Tables 10 and 11) and we wish to explicitly encode and decode these tags along with the words.

Both the tag and word based models recommended here exploit the blending mechanism of the PPM text compression scheme. Higher order contexts are tried first, but if the next word has not been seen before in this context, then a lower order context is used instead. So that the decoder knows which context to use, an "escape" symbol is transmitted to signal that the prediction should be done with a lower order context.

Experiments reported in (Teahan and Cleary, 1997) show that a simple escape strategy performs best in most cases. This method estimates the probability of the escape symbol as being proportional to the number of words in the context which have occurred only once *i.e.* the number of singletons. Performance of these models can be improved further by two simple exclusion mechanisms adapted from PPM (see Section 5)—the first, called *update exclusions*, updates the counts only in contexts that actually make the prediction. The second, called *full exclusions*, excludes words already predicted by higher order contexts. Both update and full exclusions typically improve the compression by a few per cent.

The representation of the two best performed models experimented with are shown in Figure 2. The order 1 word model (labeled "WW") is an n-gram model which first predicts the word using just the previous word, but escapes to an order 0 model if the word is not predicted, then to a character based model (a fixed order PPM model) if the word is not predicted at all. In the diagram, the symbol $\hookrightarrow$ represents the escape process. For a sequence S of length m words, the WW model is represented by the following formula:

$$p(S) = \prod_{i=1}^{m} p'(w_i|w_{i-1})$$

where p' gives the probabilities returned by the WW model in Figure 2.

The second model shown (labeled "WTW") first predicts the word using the current tag and the previous word. If this is unsuccessful, it tries based on the current

WW model	WTW model	TTWT model
$p(w_i \mid w_{i-1})$	$p(w_i \mid t_i w_{i-1})$	$p(t_i \mid t_{i-1} w_{i-1} t_{i-2})$
$\hookrightarrow p(w_i \mid)$	$\hookrightarrow p(w_i \mid t_i)$	$\hookrightarrow p(t_i \mid t_{i-1} w_{i-1})$
$\hookrightarrow$ *character model*	$\hookrightarrow$ *character model*	$\hookrightarrow p(t_i \mid t_{i-1})$
		$\hookrightarrow p(t_i \mid)$
		$\hookrightarrow p_{eq}(t_i \mid)$

Figure 2. Some models for predicting words and tags

tag only, otherwise it escapes down to the character based model. This model must include some mechanism for predicting the tags as well as the words. The best model we have found for this based on results from compression experiments is labeled "TTWT"—it first uses the prior tag, the prior word and the tag preceding that to predict the tag. If unsuccessful, it uses the escape hierarchy shown.

The WTW model combined with the TTWT model forms an n-pos model (see Section 2.2); it is represented by the following formula:

$$p(S) = \prod_{i=1}^{m} p'(t_i | t_{i-1}, w_{i-1}, t_{i-2})\ p''(w_i | t_i, w_{i-1})$$

where p' gives the probabilities returned by the TTWT model and p'' gives the probabilities returned by the WTW model shown in Figure 2.

These models are described in more detail in (Teahan and Cleary, 1997). An efficient trie-based data structure that maintains the cumulative frequency counts required for arithmetic coding for these models is also described there.

8. Estimating the entropy of English using word and tag based models

Compression experiments for these models were conducted on various texts (both tagged and non-tagged). All experiments were with texts converted to 27 character English—26 letters plus space. For these experiments, a "word" was considered to be any consecutive sequence of letters between spaces. Tags were assigned to words using the corresponding tag in the tagged text—if a word was split into more than one part (if the original word was hyphenated, for example), then the same tag was assigned to each new part *e.g.* *Vice-Chairman*/**NN** becomes *vice*/**NN** *chairman*/**NN**.

The compression experiments are split into two sections—experiments with tagged corpora where the tags have been manually checked; and experiments with texts where the tags have been assigned automatically by computer. Compression ratios are shown in bits per character (bpc). More details of these and other experiments may be found in Teahan (1997).

Table 9. How well the models compress the manually tagged texts

Tagged text	Number of words	PPMD5 + bigraph (bpc)	WW (bpc)	WTW + TTWT (bpc)
LOB Corpus:				
• *all* 5636660 *characters*	1021049	1.860	1.783	**1.781**
• *last* 10001 *chars. using the preceding text for training*	1912	1.863	1.809	**1.784**
Wall Street Journal:				
• *all* 15398849 *characters*	2614956	1.602	**1.539**	1.547
• *last* 10010 *chars. using the preceding text for training*	1736	1.553	**1.490**	**1.490**

8.1. Experiments with manually tagged texts

Compression results on manually tagged corpora are summarized in Table 9. Two pre-tagged text corpora were obtained for the experiments—the LOB Corpus and the Wall Street Journal. Table 9 compares how well the tag and word based models perform at compressing these texts with the best of the character based models, labeled "PPMD5+bigraph." It combines an order 5 PPM model with bigraph coding. Also listed are results for a sample taken from the last 10,000 characters or so[9] of each text using the preceding text for training.

The results show that performance of the tag based methods is comparable with the best word based models. This is surprising as the tag-based model has to encode *both* the tags and the words—the tags are produced by the decoding process at no extra cost. No attempt was made to optimize the performance of the tag based models. The tags used were designed primarily for linguistic purposes, and further gains should be possible by optimizing the tag set to improve compression performance. Both the word and tag based models are better than the character based method by 3 to 4%.

Figure 3 shows how training text improves compression for the WTW+TTWT model for the sample text at the end of the the Wall Street Journal. Also included are the curves (shown by the dashed lines) for the best of the character and word based models. They show that the initial performance of the tag based model is poor in comparison to the other two models but its rate of improvement is better, gaining parity with the word model at about the 10^5 character mark. It is unclear whether this trend would continue if more training text was available.

Three further curves are plotted that track the costs of encoding the tags (TTWT model), the previously seen words (WTW model) and the previously unseen words (character model). When added together, these three costs equal the overall cost for the WTW+TTWT model. The main contributing factor to the improvement in compression is the reduction in the number of unseen words with larger training texts. Consistent but slow improvement throughout is apparent for the TTWT model. The curve for the WTW model on the other hand steadily increases before plateauing out beyond 2×10^5 characters, and marginally decreasing towards the

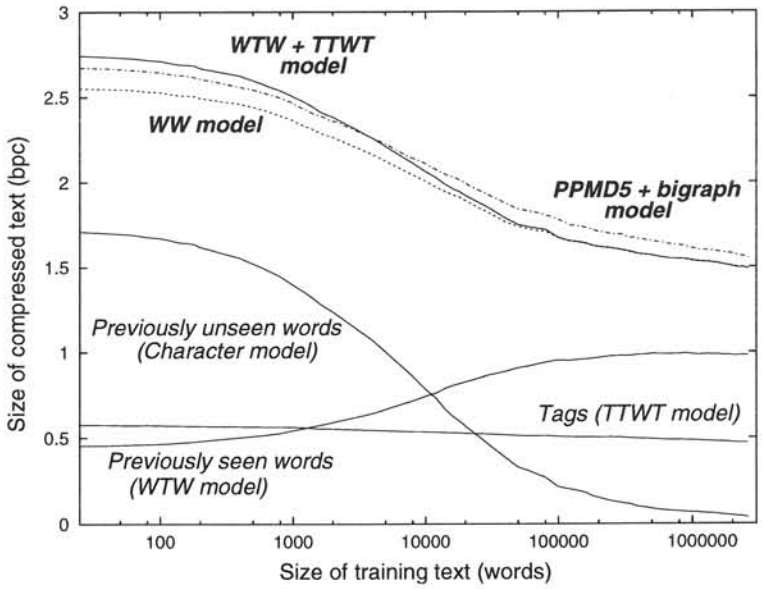

Figure 3. How training improves compression for the last 10010 characters of the Wall Street Journal

end. Experiments with other tag/word models show that the WW and WTW models are consistently the best two models for predicting the words. (For a comparison of the performance of several other models, see (Teahan, 1997).) Experiments also show that performance degrades with higher order models. It is unclear how much larger training texts and the addition of the blending mechanisms described in (Bunton, 1996) will affect these results.

8.2. Experiments with computer tagged texts

These results raise the question of how well the the models would perform if the models were tagged automatically using a computer tagger. An accuracy level of 95–97% is typically reported for computer taggers (Brill, 1994, Charniak, 1993, Church and Mercer, 1993).[10] These taggers perform well at automatically tagging text—most of the tagging errors are caused by words that are unseen or rarely seen in the training corpora. Consequently, computer tagged texts should work well with these models, and we will see that this is true in practice.

An important issue with using computer tagged texts is the tag set chosen to tag the text. The AMALGAM[11] project (Atwell et al., 1994) has developed a tagging program, accessible over the Internet, to tag text with up to 8 annotation schemes. Table 10 provides a description, the number of tags for each and an identifying label (used for reference in the following discussion). These represent the main tag sets that have been adopted in various research corpora.

Table 10. Tag sets provided by the AMALGAM tagger

Label	Tags	Description/reference
BROWN	226	Brown Corpus tag set
ICE	205	International Corpus of English tag set
LLC	210	London-Lund Corpus tag set
LOB	153	Lancaster-Oslo/Bergen Corpus tag set
PARTS	19	Tag set used by the UNIX `parts` program
PENN	45	University of Pennsylvania Corpus tag set
POW	66	Polytechnic of Wales Corpus tag set
SEC	150	Spoken English Corpus tag set

In an experiment, the AMALGAM tagger was used to tag the Jefferson Corpus text with each of the eight tag sets. The tags assigned by the tagger to the opening line of Dumas Malone's *Jefferson the Virginian* are shown Table 11. Four of the tag sets are similar—BROWN, LOB, PENN and SEC—as the latter three were based on the first. The SEC tag set is essentially the LOB tag set with minor changes. The other tag sets are noticeably different from these four. The PARTS tag set, for example, is based on the tag set devised for the UNIX `parts` program.

Each of the tagged texts was compressed using the WTW+TTWT model with results shown in Table 12. Compression ratios (sorted in ascending order) are listed for two configurations: first, for all the text found in the six volumes of Dumas Malone's work (6448790 characters and 1113235 words) without any training text; and second, for the last chapter of *Jefferson the Virginian* (46142 characters and 7984 words) trained on the remaining text from Dumas Malone's work. These results are compared at the bottom of the table with the best character and word models (PPMD5+bigraph and WW) and the standard PPMD5 model on the untagged text.

The results show that a number of the tag sets (BROWN, LOB, PENN and SEC) outperform the word based model, and all of them are better than the character based model. The best result (shown in bold font) of 1.412 bpc for the BROWN tag set compares favorably with Shannon's human-based entropy estimate of 1.3 bpc. The performance of this tag set is surprising—it was designed before the other three (which supposedly have all been "improved" linguistically). Also interesting is the performance of the SEC tag set compared to the LOB tag set. The SEC tag set was trained solely on transcriptions of spoken English. Atwell et al. (1994) point out that consequently "phenomena that are used primarily for English in its written form will not be found in SEC" (one example being abbreviations). The difference in quality of training data may also account for the differences observed between the other models.

Also shown in the table are the actual number of distinct tags found in the tagged texts (these are all notably less than the possible number of tags given in Table 10), and the compression ratios for encoding the three separate components of the overall model: the previously seen words (WTW model), the tags (TTWT model) and the previously unseen words (character model).

Table 11. Tagging the opening line to Dumas Malone's *Jefferson the Virginian*

Tag Set	AMALGAM-tagged text
Untagged	thomas jefferson was born in a simple wooden house in what is now albemarle county virginia
BROWN	thomas/**NP** jefferson/**NP** was/**BEDZ** born/**VBN** in/**IN** a/**AT** simple/**JJ** wooden/**JJ** house/**NN** in/**IN** what/**WDT** is/**BEZ** now/**RB** albermarle/**NP** county/**NN** virginia/**NP**
ICE	thomas/**N**(prop, sing) jefferson/**N**(prop, sing) was/**V**(cop,past) born/**ADJ**(edp) in/**PREP**(ge) a/**ART**(indef) simple/**ADJ** wooden/**ADJ** house/**N**(com,sing) in/**PREP**(ge) what/**PRON**(nom) is/**V**(cop,pres) now/**ADV**(ge) albermarle/**N**(prop,sing) county/**N**(prop,sing) virginia/**N**(com,sing)
LLC	thomas/**NP** jefferson/**NP** was/**VB+5** born/**VA+N** in/**PA** a/**TF** simple/**JA** wooden/**JA** house/**NC** in/**PA** what/**GC** is/**VB+3** now/**AC** albermarle/**NP** county/**NP** virginia/**NP**
LOB	thomas/**NP** jefferson/**NP** was/**BEDZ** born/**VBN** in/**IN** a/**AT** simple/**JJ** wooden/**JJ** house/**NN** in/**IN** what/**WDT** is/**BEZ** now/**RN** albermarle/**NP** county/**NPL** virginia/**NP**
PARTS	thomas/**adj** jefferson/**noun** was/**be** born/**adj** in/**prep** a/**art** simple/**adj** wooden/**adj** house/**noun** in/**prep** what/**pron** is/**be** now/**adv** albermarle/**adj** county/**noun** virginia/**noun**
PENN	thomas/**NNP** jefferson/**NNP** was/**VBD** born/**VBN** in/**IN** a/**DT** simple/**JJ** wooden/**JJ** house/**NN** in/**IN** what/**WP** is/**VBZ** now/**RB** albermarle/**NNP** county/**NNP** virginia/**NNP**
POW	thomas/**HN** jefferson/**HN** was/**OM** born/**AX** in/**P** a/**DQ** simple/**H** wooden/**AX** house/**H** in/**AX** what/**HWH** is/**OM** now/**AX** albermarle/**HN** county/**HN** virginia/**HN**
SEC	thomas/**NP** jefferson/**NP** was/**BEDZ** born/**VBN** in/**IN** a/**AT** simple/**JJ** wooden/**JJ** house/**NN** in/**IN** what/**WDT** is/**BEZ** now/**RN** albermarle/**NP** county/**NP** virginia/**NP**

To further test the robustness of these results, other texts were tagged using the three best performing tag sets (BROWN, PENN and LOB) and then compressed. Table 13 shows that the performance of the tag models with the BROWN and PENN tag sets are comparable with the best word model (WW) over a diverse range of texts. The results for the LOB tag set are slightly worse, but all three tag sets are better than the best character model (PPMD5+bigraph). The difference, however, between the best and worst models averaged over all the texts is still less than 4%. (The best result for each text is shown in bold font.)

An important application of the tag based models is their ability to compare the performance at word prediction of different taggers and tag sets (whether the tags are assigned manually or by computer). For example, an interesting comparison can be made between the results for the computer and manually tagged texts. The manually tagged LOB corpus requires 1.781 bpc to compress it; in comparison,

Table 12. Comparing the WTW+TTWT model performance for different tag sets on the AMALGAM-tagged text of the Jefferson Corpus

Tag Set	Number of tags	WTW model (bpc)	TTWT model (bpc)	Char. model (bpc)	*All the text* (bpc)	*Last chapter* (bpc)
BROWN	102	0.775	0.549	0.109	**1.433**	**1.412**
PENN	34	0.864	0.465	0.112	1.441	1.426
LOB	119	0.745	0.586	0.114	1.445	1.427
SEC	101	0.741	0.592	0.121	1.454	1.432
LLC	123	0.730	0.614	0.121	1.465	1.442
ICE	141	0.733	0.611	0.126	1.470	1.446
POW	55	0.887	0.461	0.122	1.470	1.448
PARTS	12	0.938	0.409	0.128	1.475	1.458
PPMD5					1.620	1.598
PPMD5+bigraph model					1.513	1.487
WW model					1.455	1.442

Table 13. Comparing model performance on various AMALGAM-tagged texts

AMALGAM-tagged text	**PPMD5 +bigraph** (bpc)	**WW model** (bpc)	**WTW+TTWT model** *BROWN* (bpc)	*PENN* (bpc)	*LOB* (bpc)
Jefferson Corpus	1.513	1.455	**1.433**	1.441	1.445
Brown Corpus	1.874	**1.797**	1.805	1.809	1.829
LOB Corpus	1.860	1.784	**1.781**	1.784	1.784
Wall Street Journal	1.603	1.542	1.539	**1.536**	1.554
King James Bible	1.464	1.439	1.423	**1.422**	1.432
Complete works of Shakespeare	1.931	**1.846**	1.892	1.888	1.933
Complete works of Jane Austen	1.607	1.534	**1.519**	1.523	1.531
Average	1.693	1.628	**1.627**	1.629	1.644
Weighted average	1.676	1.612	**1.611**	**1.611**	1.627

the results for the computer tagged text (using the same LOB tag set) is only slightly worse (1.784 bpc). Even more interesting is the comparison for the Wall Street Journal—the computer tagged text (with the PENN tag set) compresses better than the manually tagged text (1.536 bpc compared to 1.547 bpc). These comparisons, however, are slightly biased in that part of the training corpora used to train the AMALGAM tagger includes parts of the texts being compressed (for example, the training corpus used to train the tagger for the LOB tag set includes 20% of the LOB corpus itself). Even so, the computer tagged models still do remarkably well compared with the manually tagged models.

These results show that models based on parts-of-speech (tags) can perform as well as word based models. No attempt was made to optimize the performance of these models. As the tags used were designed primarily for linguistic purposes, further gains should be possible by optimizing the tag set to improve compression performance. Jelinek (1990) describes several tag and word based language mod-

els designed for speech recognition. He reports significant improvement in model performance using automatically derived parts of speech.

9. Summary and conclusions

A number of English texts have been compressed by machine models, with results that can be reasonably compared with previous estimates of the "entropy of English" made by Shannon and Cover and King using human subjects. Results show that machine models can perform in the range achieved by humans. In contrast to previous machine-based experiments, these estimates are based on the same source text and size of alphabet that Shannon used (Shannon, 1951). The best estimate of 1.41 bpc was obtained using a tag based model and is within 8% of Shannon's upper bound estimate of 1.3 bpc. It is expected that with larger texts, machine models can perform even better than this; for example, we project that the text contained on the World Wide Web (75×10^9 bytes of text as of late 1996) could be used to construct a language model that could compress English text down to below 1 bit per character (Teahan and Cleary, 1997).

The importance of this goes beyond the incremental improvement in the size of compressed text. These improved models have been successfully applied in novel ways to a wide variety of problems in natural language processing, such as word segmentation and OCR spelling correction (Teahan et al., 1998), identifying language dialects and ascribing authorship (Teahan, 1997), and solving cyphers (Irvine, 1997). For example, a variant of the PPM text compression algorithm applied to the problem of segmenting words in English text achieved an accuracy of 99%. The use of the character based model required substantially less training text than other methods; for example, the 5.6 Mbyte Brown Corpus was found to perform better than a previously published word-based model (Ponte and Croft, 1996) trained on 1 Gbyte of text. By applying the PPM character-based model to OCR spelling correction of text produced by a state-of-the-art OCR software, simple errors in the text were able to be corrected at the rate of approximately 10 errors per page.

Notes

1. PPM stands for "prediction by partial match."
2. The Brown Corpus (Brown et al., 1992a) is a million word collection of diverse samples taken from a variety of American English sources.
3. To simplify this and subsequent equations, it is assumed that the sequence of words is padded by "pseudo-words" $w_0, w_{-1}, \ldots$ so that probabilities such as $p(w_1|w_0)$ make sense.
4. To simplify notation, the sums shown in this and subsequent formulas are assumed to be made over all possible sequences.
5. Strictly speaking, what Shannon estimated was the "cross-entropy" of English as defined in Section 3. The entropy estimates given in the remainder of the paper are also cross-entropy estimates, but for reasons of succinctness will be referred to simply using the term "entropy."
6. All the subjects' betting schemes were combined together to form a "committee" estimate with the weighting factor being in proportion to the money won by each subject. This explains why the derived upper bound estimate is lower than the individual estimates.

7. Tilbourg's estimate was for the 26 letters only.
8. The LOB Corpus (Johansson, 1978) is a million word corpus of British English text. The Wall Street Journal (ACL-DCI CD-ROM 1, 1991) is available from the Data Collection Initiative (Liberman, 1989). The King James Bible and the Complete works of Shakespeare are both available in the public domain.
9. The size of the sample was increased slightly (to 10,001 characters for the LOB Corpus, and 10,010 characters for the Wall Street Journal) to ensure that the first character in the sample was at the beginning of a word.
10. Samuelson and Voutilainen (1997) highlight problems with using accuracy levels to compare taggers. They claim higher levels of accuracy are possible, but they acknowledge that Church disputes this. He argues that the 97% level is an "upper bound" because linguists performing the task manually disagree in at least 3% of all cases.
11. The acronym stands for "Automatic Mapping Among Lexico-Grammatical Annotation Models." The tagger is based on Brill's rule-based tagger (Brill, 1993).

References

ACL-DCI CD-ROM 1. (1991). Association for Computational Linguistics Data Collection Initiative CD-ROM 1.

Atwell, E., Hughes, J. & Souter, C. (1994). AMALGAM: Automatic mapping among lexico-grammatical annotation models. *Proceedings of ACL workshop on The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, edited by Klavans, J., pages 21–28. New Mexico State University, Las Cruces, New Mexico.

Bahl, L. R., Jelinek, F. & Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2: 179–190.

Bahl, L. R., Brown, P. F., deSouza, P. V., and Mercer, R. L. (1989). A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:1001–1008.

Bell, T. C., Cleary, J. G., and Witten, I. H. (1990). *Text compression*. Prentice Hall, NJ.

Boggess, L., Agarwal, R., and Davis, R. (1991). Disambiguation of prepositional phrases in automatically labelled technical text. *AAAI'91*, pages 155–159.

Brill, E. (1993). *A corpus-based approach to language learning*. Ph.D. thesis, University of Pennsylvania.

Brill, E. (1994). Some advances in rule-based part of speech tagging. *Proceedings of the Twelth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, Washington.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Lai, J. C., and Mercer, R. L. (1992a). An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40.

Brown, P.F., Della Pietra, V.J., deSouza, P.V., Lai, J.C. and Mercer, R.L. (1992b). Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Bunton, S. (1996). *On-line stochastic processes in data compression*. Ph.D. thesis, University of Washington.

Burrows, M., and Wheeler, D. J. (1994). A block-sorting lossless data compression algorithm. Technical report, Digital Equipment Corporation, Palo Alto, California.

Charniak, E. (1993). *Statistical language learning*. MIT Press, Cambridge, MA.

Chen, S. F. (1996). *Building probabilistic models for natural language*. D.Phil. thesis, Harvard University.

Church, K. W. & Mercer, R. L. (1993). Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics*, 19(1):1–24.

Cleary, J. G., and Teahan, W. J. (1997). Unbounded length contexts for PPM. *Computer Journal*, 40(2/3):67–75.

Cleary, J. G. & Witten, I. H. (1984). Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402.

Cole, R. A., Mariani, J., Uszkoreit, H., Zaenen, A. & Zue, V. (editors) (1995). *Survey of the state of the art in human language technology*. National Science Foundation, Directorate XIII-E

of the Commission of the European Communities Center for Spoken Language Understanding, Oregon Graduate Institute.

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

Fenwick, P. (1996). The Burrows-Wheeler Transorm for block sorting text compression: principles and improvements. *Computer Journal*, 39(9):731–740.

Francis, W. and Kučera, H. (1982). *Frequency analysis of English usage.* Houghton Mifflin, Boston, 1982.

Ganeson, R. and Sherman, A. T. (1993). Statistical techniques for language recognition: an introduction and guide for cryptanalysts. *Cryptologia*, 17(4):321–366.

Garside, R., Leech, G. and Sampson, G. (editors) (1987). *The computational analysis of English.* London. Longman.

Irvine, S. (1997) *Compression and cryptology.* D.Phil. thesis, Univ. of Waikato, Hamilton, New Zealand.

Jelinek, F. (1990). Self-organized language modeling for speech recognition. In A. Waibel and K. Lee, editors, *Readings in speech recognition*, pages 450–506. Morgan Kaufmann Pub. Inc.

Johansson, S. (1978). *Manual of information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital computers.* Department of English, University of Oslo, Norway.

Kuhn, R. & DeMori, R. (1990). A cache-based natural language model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

Liberman, M. (1989). Text on tap: the ACL/DCI. *Proceedings of the DARPA Speech and Natural Language Workshop*, San Mateo, California. Morgan Kaufman.

Malone, D. (1948). *Jefferson the Virginian.* Little Brown and Co., Boston.

Malone, D. (1977). *Jefferson and his time.* Little Brown and Co., Boston.

Moffat, A. (1990). Implementing the PPM data compression scheme. *IEEE Transactions on Communications*, 38(11):1917–1921.

Ponte, J.M. and Croft, W.B. (1996). USeg: A retargetable word segmentation procedure for information retrieval. *Fifth Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, Nevada.

Rissanen, J. and Langdon, G.G. (1979). Arithmetic coding. *IBM Journal of Research and Development*, 23:146–162.

Ristad, E.S. and Thomas, R.G. (1995). "New techniques for context modeling." *Proceedings of the 33rd Annual Meeting of the ACL*, Cambridge, Massachusetts, June 27-30.

Samuelsson, C. & Voutilainen, A. (1997). Comparing a linguistic and a stochastic tagger. *Proceedings of the 35th Annual Meeting of the ACL*, Madrid, Spain, 7-12 July.

Schmid, H. (1994). Part of speech tagging with neural networks. *Proceedings of the International conference on Computational Linguistics, 1994.*

Scragg, D. G. (1974). *A history of English spelling.* Manchester University Press, Oxford Road, Manchester.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656.

Shannon, C. E. (1951). Prediction and entropy of printed English. *Bell System Technical Journal*, pages 50–64.

Siskind, J.M. (1995). Private communication, 1995.

Teahan, W. J. (1997) *Modelling English text.* D.Phil. thesis, Univ. of Waikato, Hamilton, New Zealand.

Teahan, W. J. and Cleary, J. G. (1996). The entropy of English using PPM-based models. *Proceedings DCC'96*, edited by Storer, J.A. & Cohn, M., IEEE Computer Society Press.

Teahan, W. J. and Cleary, J. G. (1997). Models of English text. *Proceedings DCC'97*, edited by Storer, J.A. & Cohn, M., IEEE Computer Society Press.

Teahan, W. J., Inglis, S., Cleary, J. G. and Holmes, G. (1997). Correcting English text using PPM models. Technical report 97/26, Univ. of Waikato, Hamilton, New Zealand.

Tilbourg, H. C. A. (1988) *An introduction to cryptology.* Kluwer Academic Publishers.

Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L. & Palmucci, J. (1993). Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):359–382.

Wilson, W. J. (1994). Chinks in the armor of public key cryptosystems. Technical Report 94/3, University of Waikato, Hamilton, New Zealand, 1994.

Witten, I. H. and Bell, T. C. (1991). The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

Witten, I. H., Moffat, A. and Bell, T. C. (1994). *Managing gigabytes*. Von Nostrand Reinhold, New York.

Witten, I. H., Neal, R. M. & Cleary, J. G. (1987). Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540.

Ziv, J. and Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343.

Ziv, J. and Lempel, A. (1978). Compression of individual sequences via variable rate-coding. *IEEE Transactions on Information Theory*, 24(5):530–536.