Working Paper Series
ISSN 1170-487X

Correcting English Text Using PPM Models

by W J Teahan, S Inglis, J G Cleary and G Holmes

Working Paper 97/26
November 1997

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

Correcting English text using PPM models

W. J. Teahan, S. Inglis, J. G. Cleary & G. Holmes

Department of Computer Science
University of Waikato
Hamilton, New Zealand

{*b.teahan, s.inglis, j.cleary, g.holmes*}*@cs.waikato.ac.nz*

An essential component of many applications in natural language processing is a language modeler able to correct errors in the text being processed. For optical character recognition (OCR), poor scanning quality or extraneous pixels in the image may cause one or more characters to be mis-recognized; while for spelling correction, two characters may be transposed, or a character may be inadvertently inserted or missed out.

This paper describes a method for correcting English text using a PPM model. A method that segments words in English text is introduced and is shown to be a significant improvement over previously used methods. A similar technique is also applied as a post-processing stage after pages have been recognized by a state-of-the-art commercial OCR system. We show that the accuracy of the OCR system can be increased from 95.9% to 96.6%, a decrease of about 10 errors per page.

1 Motivation

In order to fully evaluate the performance of text compression algorithms, large bodies of material have to be made available in machine readable form. Optical character recognition (OCR) systems provide a fast and relatively inexpensive option for acquiring such text. However, when confronted with digitizing over 3,000 pages of Dumas Malone's *Jefferson and his time*, we found that the time taken to correct the text once it had been processed by an OCR system was prohibitive. Some common mistakes made by the OCR software were the letter *c* being confused with the letter *e* (for example, the word *thc* occurring instead of *the*, and *Jcffcrson* instead of *Jefferson*), problems with the letters *m* and *w*, *l*'s being replaced by *i*'s, the mis-recognition of *question* as *guest ion* and upper case characters appearing in the middle of words. The task of correcting these errors added substantially to the time required to complete the project. (On average, a further three minutes was required to correct each page.)

This paper describes a method of correcting these errors. The next three sections describes the theoretical background to the approach we adopt. After that, we describe how the method can be applied to two specific problems—automatic segmentation of words in text, and improving OCR output.

2 The noisy channel model

A common framework for the statistical modeling of natural language is based on a theory developed by Shannon (1948) at AT&T Bell Laboratories to model a noisy communication channel such as a telephone line. The adoption of the noisy channel model (also referred to as the *source-channel* model), was pioneered by the IBM Research Group at Yorktown Heights, New York, who applied it to the problem of continuous speech recognition (Bahl *et al.*, 1983). Since then the model has been applied to the problems of machine translation (Brown *et al.*, 1990), automatic spelling correction (Mays, Damerau & Mercer, 1990), and many other applications such as part-of-speech tagging, OCR and handwriting recognition (Chen, 1996).

In this approach, a sequence of text, S, is sent into a communications channel and the noisy text, O, comes out. For example, in spelling correction, the noisy text corresponds to output from a typist who makes spelling errors, and in machine translation, it corresponds to text in another language. The problem is to recover the original input text from the output text. This can be done by hypothesizing all possible input texts, S, and then selecting the most probable input $\hat{S}$ given the output O:

$$\hat{S} = \arg\max_S p(S|O)$$

where $\arg\max_S f(x)$ is the value of x that maximizes $f(x)$. Applying Bayes' theorem, we can rewrite this as:

$$\hat{S} = \arg\max_S \frac{p(S)\,p(O|S)}{p(O)} = \arg\max_S p(S)\,p(O|S)\,. \tag{1}$$

Hence, the most probable sequence depends both on the *prior* probability $p(S)$ that the sequence S will occur, and on the *observation* (or channel) probability $p(O|S)$ that the output O will be observed given that the sequence S has occurred. The latter depends on the application—for example, in speech recognition, the output *two* might be observed given the input *too*; in spelling correction, the output *percieve* might be likely if the input was *perceive*. The prior probability $p(S)$ is usually not available, so a model is used instead. The next section describes one such model based on the PPM text compression scheme (Cleary & Witten, 1984; Cleary, Teahan & Witten, 1995).

3 PPM 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 probabilities are estimated by collecting frequency statistics from a large corpus of text, which is called the *training text*. The size of the training text is usually large containing many millions (and in some cases billions) of words.

Classes of language models include n-gram models (which base the probability of a word on the preceding n words), n-pos models (which base the probability on

the preceding words and parts of speech), and n-graph models (models based on characters). In an n-graph model, for example, 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 conditioning context to just a few of the previous characters; for example, 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 & Cleary, 1997; Ristad, 1995; Bell, Cleary & Witten, 1990), some other examples being cryptography (Irvine, 1997), language identification (Ganeson & Sherman, 1993) and various applications for automatically correcting words in texts such as OCR and spell-checking (Kukich, 1992).

Teahan & Cleary (1996) show how the best performed text compression scheme PPM (for "prediction by partial matching") can now predict English text almost as well as humans. They performed experiments on the same text that Claude E. Shannon used in a famous experiment to estimate the entropy of English (Shannon, 1951). Shannon's estimates were based on humans guessing the upcoming text, letter by letter. Teahan and Cleary used the PPM scheme to build a computer based model, and found that performance was close to, and in some cases, superior to human results.

PPM models are based upon varying length contexts of prior characters. For example, a particular context may be the letters "thei". All the characters that have followed this context are counted, and updated progressively throughout the text. The next time the letters "thei" occur in the text, the counts are used to estimate the probability for the upcoming character. The PPM technique blends together the context models for varying lengths (such as "hei" and "ei") to arrive at a final overall probability distribution. Compressed text can then be constructed and later decoded using asymptotically optimal techniques (Bell *et al.*, 1990).

Experiments with English text show that PPM models with an upper bound of five characters in their context perform competitively. Performance of these models can be substantially improved by training them on large amounts of related text, but even unrelated text works well. The potential of character based methods is as yet untapped, especially if you consider the more than 15 billion words currently accessible on the World Wide Web as a potential source for training text.

4 PPM BASED TEXT CORRECTION

In order to find the most probable correct text given the observed (incorrect) input text, we used a variation of the Viterbi dynamic programming method (Viterbi, 1967). The Viterbi algorithm is commonly used in hidden Markov models for part

of speech tagging programs to assign the most likely sequence of tags to words in a sentence (Charniak, 1993).

Applying the noisy channel model to the problem of PPM based text correction, we wish to find the sequence of text, $\hat{S}$, that maximizes the prior and observation probabilities (see Equation 1). The prior probability $p(S)$ is estimated by training a PPM model on English text. 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.

The observation probability $p(O|S)$ is also estimated from training data—this requires two training texts: the first consists of typical output from the application containing a representative sampling of errors, and the second the corrected text. From this we can estimate the error probability—for example, in an OCR application, the probability that the letter c were mistaken for the letter e, or that the letters lc was mistaken for the letter k (some other examples are shown in Table 3). Formally, given the OCR text $O = t_1 \circ t_2 \circ \ldots t_m$, where $\circ$ denotes concatenation, and each of the T_i is a transformed character sequence, then

$$p(O|S) = \prod_{i=1}^{m} p''(t_i|c_i)$$

where $p''(t|c)$ is the probability computed from a confusion table that the OCR sequence t was reported in place of the true character c.

The key idea behind the Viterbi algorithm is that at any point in the search we need only retain, for all the possible active contexts, the sequence of text with the minimum entropy with poorer performing alternatives discarded. Thus, with a Markov based model where the maximum length of the contexts are fixed (as with PPM models), the number of alternatives being searched is kept within manageable limits.

5 Results

The method of correcting text described above has been applied to two problems—word segmentation and correction of OCR text. Experimental results for these two problems are described in the next two sections. These results were obtained by training an order 5 PPMD language model on the text in the million-word (5.6 Mbyte) Brown Corpus (Francis & Kučera, 1982).

Word segmentation models are evaluated in the literature by the two measures *recall* and *precision*. When comparing two strings, they can only differ by the number of spaces present. The prediction of spaces is determined by three classes: correct space prediction a, spurious space prediction b, and missed space c. Recall is defined as $recall = 100 \times a/(a + c)$ and precision is defined as $precision = 100 \times a/(a + b)$. A perfect model will have recall and precision of 100%.

A more general measure is available when evaluating text correction models, and the processed text can be compared directly with the original (correct) text. The

Original text:	the unit of New York-based Loews Corp that makes Kent cigarettes stopped using crocidolite in its Micronite cigarette filters in 1956.
With spaces removed:	theunitofNewYork-basedLoewsCorpthatmakesKentcigarettesstoppedusingcrocidoliteinitsMicronitecigarettefiltersin1956.
Ponte & Croft's method:	the unit of New York-based Loews Corp that makes Kent cigarettes stopped using c roc id o lite inits Micron it e cigarette filters in 1956.
PPM-based method:	the unit of New York-based LoewsCorp that makes Kent cigarettes stopped using croc idolite in its Micronite cigarette filters in 1956.

Table 1: Segmenting words in English text

difference between these two texts can be determined by the *edit distance* between them (Cormen *et al.*, 1990). The edit distance between two strings, x and y, is defined to be the minimum transformation sequence that converts x into y. We define *accuracy* to be $100 - 100 \times e/m$, where e is the edit distance between the corrected and original text, and m is the number of characters in the original text. The transformation operations are: delete a character, insert a new character, and change a character into another. For example, the edit distance between *Eloplcinsons* and *Hopkinsons* is 4. A sequence of operations that performs this transformation is: delete *E*, change *l* to *H*, delete *l*, change *c* to *k*.

5.1 Word segmentation

English has an alphabetic orthography and word spacing, unlike other languages such as Japanese and Chinese, so it is relatively easy to adopt the "word" as a basic unit by using blank space and various punctuation marks as delimiters.

Word segmentation is an important task required for applications that do not start with an orthographic representation, such as speech recognition, or the automated transcription of Morse code. Ponte & Croft (1996) introduced a method (USeg) for predicting space positions and examined its performance using a 500 KB extract from the Wall Street Journal. USeg is a word-based model that was trained on 1 Gigabyte of text, and produced a recall of 93.54% and precision of 90.03%.

Using our PPM character-based method on the same corpus produces both recall and precision rates of 99.52%, with an edit distance accuracy of 99.04%. This improvement over Ponte and Croft's results used only a small fraction of their training text.

Table 1 shows an example used by Ponte & Croft with the addition of the predictions made by PPM. The improvements that the PPM model provides are evident in this small example. Although the word *Micronite* does not occur in the Brown Corpus, the word was correctly segmented using PPM. Likewise, *inits* was correctly segmented into *in* and *its*. In this example, PPM made two mistakes. The space in *Loews Corp* was not predicted because *LoewsCorp* required 54.3 bits to encode the

Original image	*OmniPage text*	*PPM corrected text*
of the American	of the Americam	of the American
he became forty-one	he became fotty-one	he became forty-one
was concerned this	v as concetntd this	v as concerntd this
accepted election as	accepted elecclon as	accepted electlon as
however, that the	however, chat the	however, that the
taking his seat, he	taking his seaL he	taking his seat, he
for Annapolis	for Ammapolis	for Annapolis
Hopkinsons and	Eloplcinsons and	Hopkinsons and
generally but because	generally hut because	generally but because
her future responsibilities.	her fumre responsi...	her future responsi...
that in marriage	that m marriage	that in marriage

Table 2: PPM correction of OCR text

text while the original required 55.0 bits. Similarly, an extra space was added in *crocidolite* because the space reduced the number of bits to encode it from 58.7 to 55.3.

5.2 Correcting OCR text

By comparison, correcting output that was generated by an OCR system is a more difficult problem. In this section, we apply the PPM model to the output of the OmniPage 7.0 commercial character recognition system, in an attempt to correct simple mistakes. Ideally, the PPM model would be embedded in the OCR system, so that access to the complete probability distribution across the different characters is provided. Unfortunately, as we are analyzing the output as a post-processing stage, this information is unavailable.

The images required for this experiment were digitized at 150 dpi from an original copy of *Jefferson the Virginian* printed in 1948 (the first volume of Malone's *Jefferson and his time*). The pages used to train the confusion models were taken from the five chapters (up to 89 pages) immediately prior to the last chapter in *Jefferson the Virginian*. These pages contained 32,000 words (185 KB). The last chapter was used as testing text and consisted of 21 pages containing 8,000 words (46 KB). An order 5 PPM model trained on the Brown Corpus text was used to train the English language model.

We define a *confusion* to be the transformation required to correct a sequence of a small number of characters in the text. We use *observed* $\rightarrow$ *corrected* to denote a confusion transformation from the observed to the corrected text: for example, $lc \rightarrow k$ denotes that the bigraph *lc* is corrected to the letter *k*. To limit the searching required by the Viterbi algorithm, the confusions that occurred only once, or contained a space were discarded.

A sample of errors found on the first three pages of the last chapter is shown in

Confusion	*freq.*	*Confusion*	*freq.*	*Confusion*	*freq.*	*Confusion*	*freq.*
El → El	4	e → o	2	l → I	5	m → rn	29
El → H	2	e → rt	3	l → l	2943	m → rs	9
L → L	50	e → se	2	lc → k	2	m → ru	7
L → i	2	e → t	7	lc → lc	5	m → ta	5
L → t,	2	e → th	4	m → an	8	m → te	3
c → c	2293	e → tr	4	m → ct	3	m → ti	2
c → e	7	e → tt	3	m → in	65	m → to	104
c → t	22	h → b	12	m → m	2082	m → tr	2
e → a	4	h → h	4727	m → n	6	m → ts	13
e → at	2	h → i	6	m → ni	5	m → tt	4
e → c	18	h → it	14	m → nt	5	m → tu	25
e → e	11180	h → t	2	m → nx	3	m → ut	3
e → i	4	h → th	29	m → ra	18	t → r	3
e → ie	2	h → ti	6	m → ral	2	t → t	6527
e → is	4	l → -l	2	m → rm	3		

Table 3: A sample of confusions generated from OmniPage output

Table 2. (These are shown in the order that they occur in the text). Extracts from part of these images that are relevant to the examples are shown in the first column. Some of the images are slightly skewed, a side-effect of how the pages were placed on the scanner.

Table 3 lists a sample of the confusions that were learnt from the confusion training data. The frequencies for both the incorrect and correct transformations are shown. For example, the letters *El* occurred six times in the training data; twice it was incorrectly replaced by the single letter *H*, and four times it was correctly identified. These confusions relate to the example corrections shown in Table 2. The correction of *Americam* to *American*, *fotty-one* to *forty-one*, *chat* to *that*, *Ammapolis* to *Annapolis* and *hut* to *but* all stem from single letter confusions (i.e. $m \rightarrow n$, $t \rightarrow r$), whereas the corrections *seaL* to *seat,* (including the extra comma), *Eloplcinsons* to *Hopkinsons*, *fume* to *future*, and *m* to *in* derive from double letter confusions. The correction of "*v as concetntd*" (line 3 in the diagram) and "*elecclon*" (line 4), was only partially successful because the confusions required to correct these errors (for example, $t \rightarrow e$ and $l \rightarrow i$) had not been seen at least twice in the training data (and therefore do not occur in Table 3).

Figure 1 shows how the number of confusions increases as the number of training pages increases. Figure 2 shows how the edit distance accuracy between the PPM corrected output and the original correct text varies with the number of training pages. After 60 pages, the edit distance with respect to the "correct" text had decreased from 1757 down to 1558, an increase in accuracy from 96.28% to 96.70%.

Figure 3 shows how the edit distance accuracy between the corrected and correct text decreases for order 4 and 5 models and for different sized training text.[1] The

[1]The number of pages used to train the confusion model was limited to thirty to conserve memory.

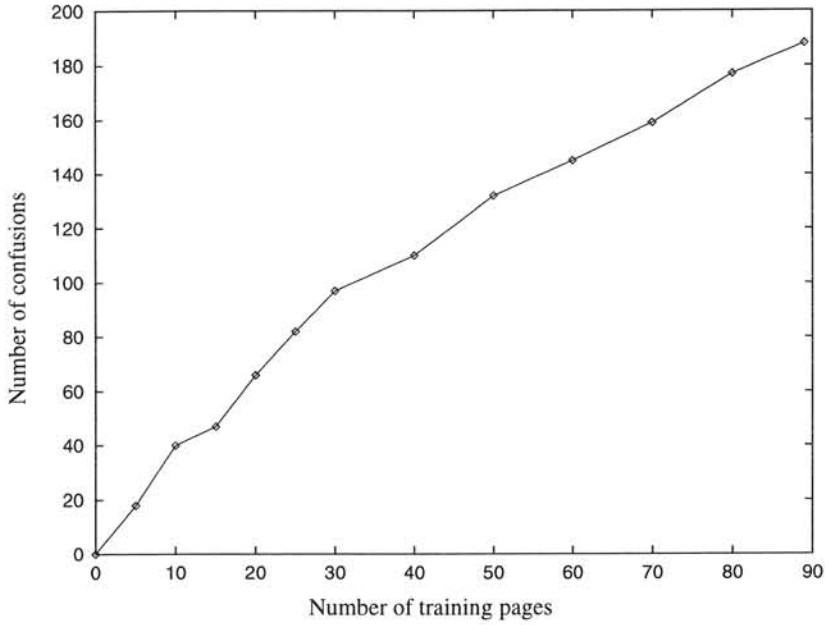

Figure 1: How training on errors produced by OmniPage output affects the number of confusions

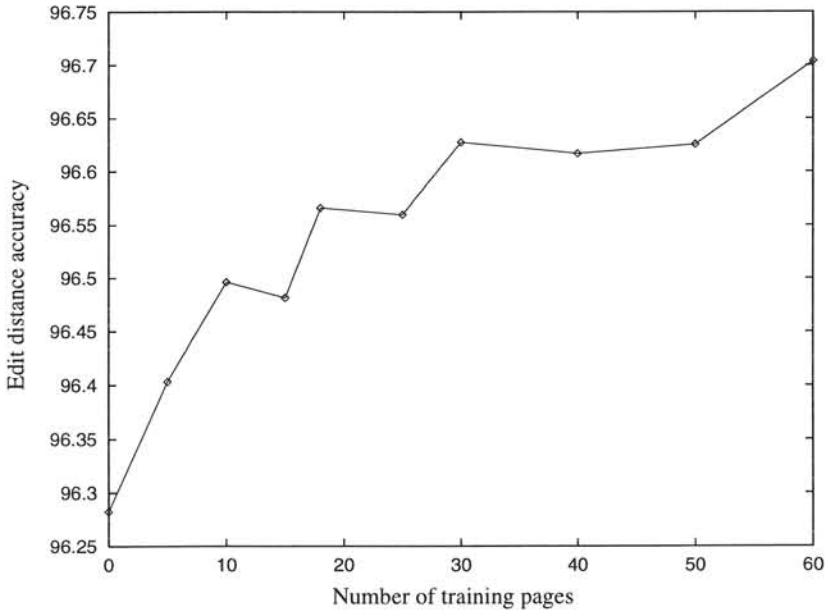

Figure 2: How training on errors produced by OmniPage output affects the accuracy of the PPM corrected output

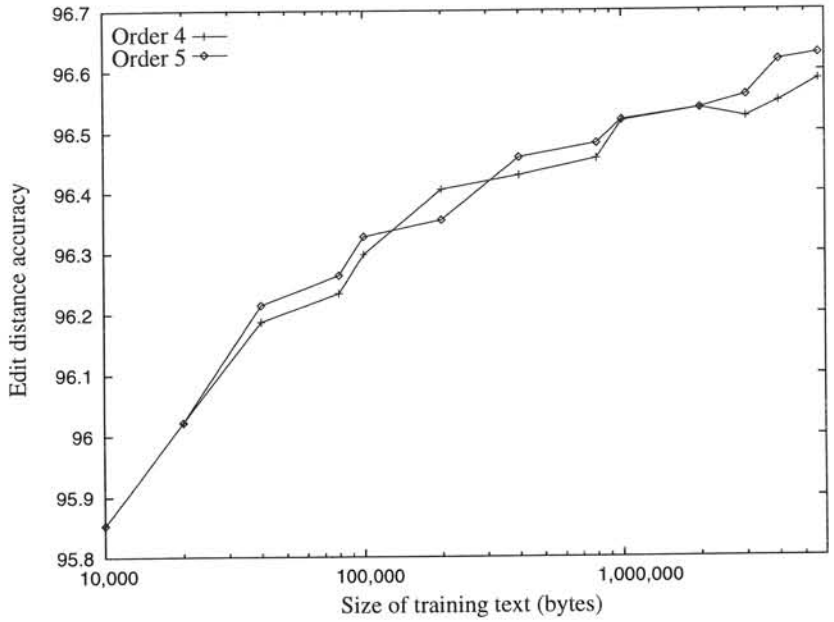

Figure 3: How the edit distance accuracy varies for different order models and different size training texts

x axis is plotted using a logarithmic scale. The graph shows that the order 5 model achieves slightly better error correction than the order 4 model. The improvement is about 1% after training on 5.6 Mb.

6 Summary and conclusions

We have introduced a method for correcting errors in English text based on a PPM model. This method was applied to two problems—segmenting words in English text, and improving the output from a commercial OCR system. The accuracy of the PPM word segmenter was 99.04% with a recall and precision of 93.54%. The use of the character based model required substantially less training text than other methods; for example, the 5.6 Mb Brown Corpus was found to perform better than a previously published model trained on 1 Gbyte of text. By applying the PPM character-based model to OCR spelling correction, we have been able to improve edit distance accuracy from 95.9% to 96.6%, which is a decrease of 10 errors per page.

Previous experience with models for text compression indicates that these results can be substantially improved by using more training text. For OCR, it will be particularly important to increase the size of the confusion training data. The elimination of confusions that occurred only once or contained spaces was done mainly to reduce the space and execution time of the Viterbi algorithm. We are working on improving the efficiency of our implementation and assume that using more confusion data will further improve the accuracy.

7 REFERENCES

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

Bell, T.C., Cleary, J.G. & Witten, I.H. 1990. *Text compression.* Prentice Hall, New Jersey.

Brown, P.F., Cocke, J., Della Pietram, S.A., Della Pietra, V.J., Jelinek, F., Lafferty, J.D., Mercer, R.L. & Roossin, P.S. 1990. "A statistical approach to machine translation." *Computational Linguistics*, **16**(2): 79–85.

Charniak, E. 1993. *Statistical language learning.* MIT Press, Cambridge, Massachusetts.

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

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

Cleary, J.G., Teahan, W.J. and Witten, I.H. 1995. "Unbounded length contexts for PPM," *Proceedings DCC'95.* IEEE Computer Society Press, 52–61.

Cormen, T.H., Leiserson, C.E. and Rivest, R.L. 1990. *Introduction to algorithms.* MIT Press, Cambridge, Mass.

Francis, W.N. and Kučera, H. 1982. *Frequency analysis of English usage: Lexicon and grammar.* Houghton Mifflin, Boston.

Irvine, S.A. 1997. *Compression and Cryptology.* D.Phil. thesis, Univ. of Waikato, N.Z.

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

Mays, E., Damerau, F.J. & Mercer, R.L. 1990. "Context-based spelling correction" in *Proceedings, IBM Natural Language ITL*, France, 517–522.

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.

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

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*, 50–64.

Teahan, W.J. 1997. *Modelling English text.* D.Phil. thesis, Univ. of Waikato, N.Z.

Teahan, W.J. and Cleary, J.G. 1996. "The entropy of English using PPM-based models," *Proceedings DCC'96.* IEEE Society Press, 53–62.

Viterbi, A.J. 1967. "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Transactions on Information Theory*, **13**, 260–269.