Natural Language Engineering **14** (3): 417–430. © 2007 Cambridge University Press
doi:10.1017/S1351324907004597 Printed in the United Kingdom

A new PPM variant for Chinese text compression

PEILIANG WU and W. J. TEAHAN

School of Informatics, University of Wales Bangor,
Dean Street, Bangor, Gwynedd LL57 1UT, UK
email: {perry,wjt}@informatics.bangor.ac.uk

(*Received 19th September 2006; revised and accepted 15 March 2007; first published online 1 October 2007*)

Abstract

Large alphabet languages such as Chinese are very different from English, and therefore present different problems for text compression. In this article, we first examine the characteristics of Chinese, then we introduce a new variant of the Prediction by Partial Match (PPM) model especially for Chinese characters. Unlike the traditional PPM coding schemes, which encodes an escape probability if a novel character occurs in the context, the new coding scheme directly encodes the order first before encoding a symbol, without having to output an escape probability. This scheme achieves excellent compression rates in comparison with other schemes on a variety of Chinese text files.

1 Introduction

Text compression for Chinese characters is problematical because of its large alphabet size. Today, few Chinese texts are translated into electronic form. It is estimated that almost 70% of Chinese paper-based information is urgently required to be transformed into electronic form (Vines and Zobel 1998). However, very little work has been done in the area of Chinese text compression. Of the results that have been published, the researchers have chosen to experiment with a small number of files they have collected themselves. They have also used different compression measures to compare results. For example, Lua (1994) in his article reported on an *n-gram* scheme to encode Chinese characters and achieved the compression rate of 6.44 bits per Chinese character (bpCc) on 14-bit Chinese characters. Gu (1995; 2005) describe a Prediction by Partial Match (PPM)-like predictive scheme for Big5 encoded Chinese text, which reduces a text file's size to 5.424 bpCc. Vines and Zobel (1998) modified the PPM model for Chinese characters by changing the unit of encoding from 8 to 16 bits, and achieved 6.2 bpCc. Cheng *et al.* (1999) introduce a word-based dictionary coding algorithm that use both the LZW and Huffman coding schemes. It is reported that the compression results are comparable to that of the traditional PPM-based schemes. Most recently, Ong and Ng (2005) announced the compression rate of 7.925 bpCc for Dynamic Markov Compression. However, these results cannot be directly compared because they have been obtained under entirely different experimental conditions (such as testing corpus). A primary purpose of this

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

article is to provide a standardized framework under which such experiments can be done (see Section 5).

Text compression is a useful vehicle for comparison of different language models for natural language processing (NLP) applications. Text mining is one application that can make use of text compression models to achieve better NLP performance by using the Viterbi algorithm and variations to annotate the text (Witten *et al.* 1999). On the other hand, we have found that understanding gained from applying text compression models to NLP applications can lead to further improvements in text compression itself (Teahan and Harper 2001). The work described here, although focussed on the problem of text compression, is an important precursor future work we wish to perform in the area of NLP for Chinese and other Asian languages.

In this article, we examine the characteristics of Chinese language text in Section 2, then introduce the traditional PPM model in Section 3. Section 4 discusses our new adaptive PPM variant, dubbed PPMO (for PPM with *order stream*). Section 5 discusses the Bangor Mandarin Chinese Corpus (BMCC), which we have used in our experimental analysis. Section 6 reports the significant experimental results we have obtained with our corpus. A summary and discussion of the implications of the results is provided in Section 7.

2 Chinese text

The modern Chinese dictionary lists about 8,000 Chinese characters. Unlike English, which is usually encoded in the computer by the ASCII character set, Chinese is usually encoded by using a more specific character set such as GB2312-80 and GB13000.

For example, the table for the GB2312-80 character set has 94 rows (sections) and 94 columns (positions), which contains 682 symbols, 6,763 Chinese characters, and some user-defined characters (Figure 1). Both row and columns are encoded as an ASCII character. The whole table is divided into four parts.

As the above figure shows, every Chinese character can be encoded by using the GB2312-80 character set, which uses a two-dimensional table. Because the GB Chinese code is a two-byte encoding scheme, the high byte is encoded by a row, or 'section code', and the low byte is encoded by a column, or 'position code'.

Unlike English words, which are separated by spaces, Chinese text is not naturally segmented by space or any other symbol. Therefore, it is not clear which is the most appropriate symbol to use for encoding in Chinese. In English experiments, for example, it has been found that word-based symbols are significantly better than character-based ones (Teahan 1998), but it is not clear whether the same would occur with Chinese text.

3 Prediction by Partial Match

PPM is a finite-context model because the predictions are based on a finite number of preceding symbols. It predicts upcoming symbols by employing a suite of context models of different orders, from some pre-defined maximum down to a default

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

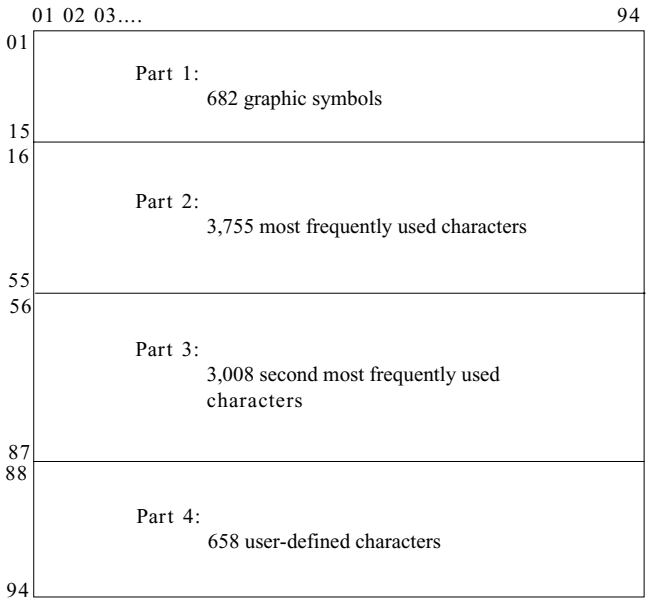


Fig. 1. Chinese GB2312-80 coding table.

−1 model (Cleary and Witten 1984; Bell *et al.* 1990). If a novel symbol occurs in the context, an escape probability will be assigned. PPM achieves excellent compression rates, although this model seems expensive both in terms of memory and execution speed. There are several variations for PPM, such as PPMA, PPMB, PPMC, PPMD, PPM* (Cleary and Witten 1984; Moffat 1989; Cleary and Teahan 1997; Shkarin 2002), which are mainly named according to the escape method they employ. Moffat's PPMC, a well-crafted implementation that uses the update exclusion mechanism (Moffat 1990), has become the benchmark version (Cleary and Teahan 1997). In a recent article, Dmitry Shkarin introduced PPMII, which stands for PPM with Information Inheritance. It takes advantage of the similarity of distribution functions in parent and child contexts and sets the initial value of the generalized symbol frequency in the child context with regard to information about this symbol gathered in the parent context (Shkarin 2002). PPMII achieves a good compression rate as well as excellent compression speed. PPM-Ch, introduced by Wu and Teahan (2005), adapts the PPM model especially for Chinese text by using pre-processing prior to encoding and uses 16 bits symbol schemes, plus frequency sorting techniques, as well as achieves excellent compression results.

To illustrate the traditional PPM model, we use PPMC as an example. Let A be a discrete alphabet consisting of $|A| > 2$ symbols and D be the maximum order of the model, where $d \leq D$ be the current coding order of a model. The probability of an upcoming symbol $x_{n+1} = \varphi, \varphi \in A$ depends on the *current context* $s_d = x_n, \ldots, x_{n-d+1}$. Let $c_d(\varphi)$ denote the number of times that the symbol φ occurs in the context s_d. Let t_d be the total number of unique symbols that occur after the context s_d. Let $T_d = \Sigma\, c_d(\varphi)$, which denotes the total number of times that the context s_d has been. Then, the probability of a symbol φ occurring in the context s_d

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

Table 1. *PPMC model after processing the string '中国的中是中华中间的中'*
z

Order $d=2$			Order $d=1$			Order $d=0$			Order $d=-1$		
Predictions	$c_d(\varphi)$	$p_d(\varphi)$	Predictions	$c_d(\varphi)$	$p_d(\varphi)$	Predictions	$c_d(\varphi)$	$p_d(\varphi)$	Predictions	$c_d(\varphi)$	$p_d(\varphi)$
中国 →的	1	$\frac{1}{2}$	中 →国	1	$\frac{1}{8}$	→ 中	5	$\frac{5}{17}$	→ A	1	$\frac{1}{\lvert A\rvert}$
→*esc*	1	$\frac{1}{2}$	→是	1	$\frac{1}{8}$	→ 的	2	$\frac{2}{17}$			
			→华	1	$\frac{1}{8}$	→ 国	1	$\frac{1}{17}$			
国的 →中	1	$\frac{1}{2}$	→间	1	$\frac{1}{8}$	→ 是	1	$\frac{1}{17}$			
→*esc*	1	$\frac{1}{2}$	→*esc*	4	$\frac{1}{2}$	→ 华	1	$\frac{1}{17}$			
						→ 间	1	$\frac{1}{17}$			
的中 →是	1	$\frac{1}{2}$	国 →的	1	$\frac{1}{2}$	→*esc*	6	$\frac{6}{17}$			
→*esc*	1	$\frac{1}{2}$	→*esc*	1	$\frac{1}{2}$						
中是 →中	1	$\frac{1}{2}$	的 →中	1	$\frac{1}{2}$						
→*esc*	1	$\frac{1}{2}$	→*esc*	1	$\frac{1}{2}$						
是中 →华	1	$\frac{1}{2}$	是 →中	1	$\frac{1}{2}$						
→*esc*	1	$\frac{1}{2}$	→*esc*	1	$\frac{1}{2}$						
中华 →中	1	$\frac{1}{2}$	华 →中	1	$\frac{1}{2}$						
→*esc*	1	$\frac{1}{2}$	→*esc*	1	$\frac{1}{2}$						
华中 →间	1	$\frac{1}{2}$	间 →的	1	$\frac{1}{2}$						
→*esc*	1	$\frac{1}{2}$	→*esc*	1	$\frac{1}{2}$						
中间 →的	1	$\frac{1}{2}$									
→*esc*	1	$\frac{1}{2}$									
间的 →中	1	$\frac{1}{2}$									
→*esc*	1	$\frac{1}{2}$									

is:

$$p_d(\varphi) = \frac{c_d(\varphi)}{t_d + T_d}, c_d(\varphi) > 0.$$

The probability of an *escape* occurring in the context s_d is:

$$p_d(esc) = \frac{t_d}{t_d + T_d}.$$

The overall conditional probability for any symbol φ is:

$$P_{PPMC}(\varphi) = \left[\prod_{i=d+1}^{D} p_d(esc|s_i)\right] * p_d(\varphi|s_d).$$

To illustrate the operation of PPM, Table 1 shows the state of seventeen conditioning classes with $d = 2, 1, 0$ and -1 after the input string '中国的中是中华中间的中' has been processed. Note that for PPM algorithms, each symbol is usually directly encoded using arithmetic coding (Moffat 1990; Moffat *et al.* 1998) based on the probability suggested by the model. The PPM encoding algorithm proceeds from the highest order model $d = 2$; if the context successfully predicts the next character

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

φ, the associated probability $p_d(\varphi)$ is used to encode it. For example, if '是' followed the input string '中国的中是中华中间的中', the probability of $\frac{1}{2}$ would be used because a successful prediction '的中' →'是' has been made in the order 2 model.

Suppose that the character following the input string '中国的中是中华中间的中' is '国', which is not predicted by the current context '的中'. Consequently, an *escape* probability $\frac{1}{2}$ would be encoded by using the arithmetic coding algorithm and the encoding process downgrades from order 2 to order 1 model, $d = 1$ context '中' is used. Then the desired prediction '中' →'国' is selected with probability $\frac{1}{8}$. The total probability to encode the character '国' is $\frac{1}{2} \times \frac{1}{8} = \frac{1}{16}$, which is 4 bits. We can improve this further if we use the '*exclusion*' mechanism, since '中' →'是' cannot be encountered because if it did, it would have been encoded at the order 2 context model. The probability $\frac{1}{7}$ can be used in the current order 1 model for the character '国', which saves coding space in comparison with probabilities encoded without exclusions.

If a novel character '正' follows the input string, the encoding process would escape right down to the base level $d = -1$. In this level, all characters would be encoded as a probability of $\frac{1}{|A|}$, where $|A|$ is the alphabet size of the non-encoded symbols. For Chinese, $|A| \approx 8,000$.

4 PPMO: A new PPM variant for Chinese text

As mentioned above, for Chinese characters, the size of each character requires a two-byte scheme, or 16 bits, to encode it. The main problem of using the traditional PPM algorithm for Chinese text is that the proportion of escape probabilities that need to be encoded is much higher than for English text, which leads to wasted code space. In addition, because of the large alphabet and the large proportion of escapes, performing full exclusions can significantly reduce execution speeds. Lower order models also tend to perform better. Experiments show that on average Chinese text gets its best compression results by using an order 2 PPM model as its maximum order. This compares to order 5 for English (Teahan 1998).

To overcome the disadvantages of the traditional PPM algorithm, we separate the coding process into two parts, namely, *order stream* and *symbol stream*. The order stream outputs a coding order for each coding symbol, whereas the symbol stream outputs the coding symbol itself on the baris of the context for the particular order. To illustrate this algorithm, we use the same definitions that we mentioned in the previous section.

- If a symbol φ has been found in the context s_d, the current maximum order d from the order stream is encoded first, followed by the symbol itself from the symbol stream, which is encoded using the following probability:

$$P_s(\varphi) = \frac{c_d(\varphi)}{T}, \qquad c_d(\varphi) > 0.$$

- If a symbol φ has not been found in the context s_d, it will downgrade to the lower order, ultimately backing off to the default order -1 model if needed. To make sure a symbol can be found in the order -1 model, we allocate one

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

Table 2. *PPMO model when processing the string 中国的中是中华中间的中*

Character	中	国	的	中	是	中	华	中	间	的	中
Order Stream	-1	-1	-1	0	-1	0	-1	0	-1	0	1
Symbol Stream	$\frac{1}{\|A\|}$	$\frac{1}{\|A\|}$	$\frac{1}{\|A\|}$	$\frac{1}{3}$	$\frac{1}{\|A\|}$	$\frac{2}{5}$	$\frac{1}{\|A\|}$	$\frac{3}{7}$	$\frac{1}{\|A\|}$	$\frac{1}{9}$	1

Table 3. *PPMO model after processing the string 中国的中是中华中间的中*

Symbol stream												
Order $d=2$			Order $d=1$			Order $d=0$			Order $d=-1$			
Predictions	$c_d(\varphi)$	$P_s(\varphi)$	Predictions	$c_d(\varphi)$	$P_s(\varphi)$	Predictions	$c_d(\varphi)$	$P_s(\varphi)$	Predictions	$c_d(\varphi)$	$P_s(\varphi)$	
中国 →的	1	1	中 →国	1	$\frac{1}{4}$	→ 中	5	$\frac{5}{11}$	→ A	1	$\frac{1}{\|A\|}$	
			→是	1	$\frac{1}{4}$	→ 的	2	$\frac{2}{11}$				
国的 →中	1	1	→华	1	$\frac{1}{4}$	→ 国	1	$\frac{1}{11}$				
			→间	1	$\frac{1}{4}$	→ 是	1	$\frac{1}{11}$				
的中 →是	1	1				→ 华	1	$\frac{1}{11}$				
			国 →的	1	1	→ 间	1	$\frac{1}{11}$				
中是 →中	1	1										
			的 →中	1	1							
是中 →华	1	1										
			是 →中	1	1							
中华 →中	1	1										
			华 →中	1	1							
华中 →间	1	1										
			间 →的	1	1							
中间 →的	1	1										
间的 →中	1	1										

count to each symbol in the alphabet. Once the model that will be used to encode the symbol has been selected, the order of the model is encoded first, then the model is used to encode the symbol.

Table 2 shows the order and symbol probabilities used when processing the string 中国的中是中华中间的中.

Table 3 illustrates PPMO model after processing the same string 中国的中是中华中间的中. Suppose '是' followed the input string, the encoding process outputs 2 in the order stream because '是' would be encoded in the order 2 model, then outputs a probability of $\frac{1}{2}$ since a successful prediction '的中'→'是' has been made in the order 2 model.

Suppose instead that the character following the input string '中国的中是中华中间的中' is '国', which is not predicted by the current context '的中'. Rather than output an *escape* probability, the encoding process downgrades from order 2 to the order 1 model and the $d = 1$ context '中' is used. Then a 1 is output in the order stream because the desired prediction '中'→'国' is selected with probability $\frac{1}{8}$.

If a novel character '正' follows in the input string, the encoding process would back off to the default order $d = -1$. A -1 would be output in the order stream and all characters would be encoded as a probability of $\frac{1}{A}$.

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

```
31101201111101232123312101112101221112111101001112 11
23312111112232312123311212321111111231112212330121 11
23312001211210101231233123210101211011211212333333 33
33312312111111211201101211122331221211212121101211 22
```

Fig. 2. Sample output of the order stream for encoding the Chinese Bible.

As Table 3 shows, all the coding probabilities in the order 2 context model and mostly in the order 1 model are 1, $P_s(\varphi) = 1$. This compares with the probability distributions in Table 1, where all probabilities must be smaller than 1 ($p_d(\varphi) < 1$) by the nature of the traditional PPM encoding scheme.

Moveover, if the total number of unique symbols that occur after the context is 1 ($t_d = 1$), the encoding probability would be 1, which implies the symbol stream need not output anything when encoding. In the above example, '是' follows the input string, the encoding process only outputs 2 in the order stream and nothing in the symbol stream because in the order 2 context model, the encoding context '的中' only has one unique symbol that '是' occurs after it.

Obviously, there is the additional expense of encoding the order stream. The trade-off between our new method and the traditional PPM method is between encoding the order stream separately versus additional code space from encoding escapes and backing off to lower orders. Interestingly, at least for Chinese text, we have found the new method outperforms the traditional approach, but understandably, it is dependent on how efficiently the order stream can be encoded. We now discuss one approach on how this can be done.

Figure 2 shows a sample file of the order stream for encoding the Chinese Bible in Section 5 SBMCC.

It is clear that the order stream is highly predictable because of its small alphabet size and repetitive nature. For example, if the maximum coding order of the symbol stream is 2 for a model, there are only four possibilities in the order stream, which are $\{-1, 0, 1, 2\}$. (In Figure 2, we rescale the order stream to $\{0, 1, 2, 3\}$ by adding 1 to each order.) We can regard the order d as separate entities to be encoded and use the traditional PPM algorithm to code the order.

The conditional probability for order d is:

$$P_o(d) = \left[\prod_{i=d^*+1}^{D^*} p_{d^*}(esc|s_i^*) \right] \times p_{d^*}(d|s_d^*),$$

where D^* denotes the maximum coding order of the order stream; $d^* \leq D$ denotes the current coding order; and s_i^* denotes the current order stream coding context.

The overall coding probability for a symbol φ is:

$$P_{PPMO}(\varphi) = P_o(d) \times P_s(\varphi).$$

When decoding, we first decode the order stream to find out which order the symbol has been encoded with, and then decode the symbol stream on the basis of the order of the model.

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

Although PPMO is a two-step encoding method, it is still possible to take advantage of the exclusion mechanism that is so effective for the standard PPM algorithm. For example, if '国' follows the input string in Table 2, we can exclude the symbol '是' in the order 1 context model since '是' has not been encoded in the order 2 '的是' context model. The probability of $\frac{1}{3}$ would be used in comparison with the probability of $\frac{1}{4}$ when no exclusion mechanism is used.

PPMO requires a two-step encoding for each symbol, but the traditional PPM algorithm may require many escapes for each symbol, and if the alphabet is large and escaping is frequent, as the case with Chinese text, it is not clear which approach will have better speed performance.

5 Bangor Mandarin Chinese Corpus

To standardize future experimental results, we have set up a corpus named BMCC for experimental usage. It can be downloaded from the following Internet site: http://www.informatics.bangor.ac.uk/~wjt/AIIA/BMCC/BMCC.htm.

According to the file size, the whole corpus is divided into two subcorpora. One is called the *Small Bangor Mandarin Chinese Corpus,* or SBMCC, the other is the *Large Bangor Mandarin Chinese Corpus,* or LBMCC.

SBMCC contains all files smaller than 800 KB. We select some news, novels, and fictional articles. We also include a balanced Mandarin Chinese Corpus named Lancaster Corpus of Mandarin Chinese (LCMC) (McEnery and xiao 2004), which is constructed by the Linguistics Department of Lancaster University and can be obtained from http://bowland-files.lancs.ac.uk/corplang/lcmc/. It contains fifteen different text categories. The corpus comprises Mandarin Chinese text published in Mainland China, including newspapers, books, as well as text from the Xinhua News Agency. All the text is segmented by space according to the Chinese word as judged by a native speaker, but for the purposes of our experimental corpus, we have removed these spaces. We have also removed the XML tags that accompany the LCMC files.

LBMCC contains all files larger than 800 KB. We select the Chinese Bible, and include some TREC Mandarin Corpus (2000) files and the PH Corpus (Jin 1992). All TREC files come from the Xinhua News Agency and were collected for the Text Retrieval Conference. We have concatenated all the TREC files into one big file, removed all the XML tags, and selected five partitions from the head of that file (these are split according to story boundaries).

6 Experimental results

We use some other compression tools for comparison. *gzip* and *bzip2* are the two popular compressors under Unix/Linux environment. *WinRAR* and *ABC2.4* are two compressors under the Microsoft Windows environment. *ABC2.4* uses the Burrows-Wheeler Compression Algorithm (Burrows and Wheeler 1994). PPMC, PPMD (Moffat 1990), and PPMII (Shkarin 2002) are several variations of the PPM model. *PPM-Ch* is 'PPM Chinese model' (Wu and Teahan 2005) and *PPMO* is our

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

Table 4. *Chinese text compression for Small Bangor Mandarin Chinese Corpus using different compressors (the best result for each file is shown in bold)*

File	Size (bytes)	*gzip* (bpCc)	*bzip2* (bpCc)	*WinRAR* (bpCc)	*ABC* (bpCc)	*PPMC* (bpCc)	*PPMD* (bpCc)	*PPMII* (bpCc)	*PPM-Ch* (bpCc)	*PPMO 2 + 2* (bpCc)
Chinese news1	15,371	9.144	8.992	9.098	8.856	8.426	8.434	8.238	7.910	**7.750**
Chinese news2	26,195	9.602	9.384	9.546	9.126	8.936	8.956	8.648	8.336	**8.162**
Chinese article1	19,245	9.766	9.574	9.726	9.314	9.058	9.100	8.680	8.496	**8.292**
Chinese article2	38,724	9.410	8.802	9.32	8.516	8.458	8.456	8.166	8.042	**7.910**
Chinese article3	80,760	8.472	7.772	8.316	7.544	7.432	7.412	7.142	7.028	**6.908**
Chinese book1	53,706	8.932	8.406	8.790	8.166	8.064	8.072	7.760	7.656	**7.520**
Chinese book2	436,656	9.340	8.026	8.770	7.660	7.744	7.714	7.378	7.474	**7.354**
LCMC-A	271,053	9.526	8.706	9.036	8.446	8.394	8.386	8.030	7.926	**7.778**
LCMC-B	170,469	8.826	8.170	8.442	7.930	7.814	7.818	7.512	7.420	**7.268**
LCMC-C	112,681	8.204	7.530	7.854	7.380	7.18	7.166	6.91	6.808	**6.696**
LCMC-D	103,031	8.956	8.476	8.672	8.226	8.112	8.110	7.800	7.624	**7.500**
LCMC-E	218,473	9.002	8.564	8.646	8.286	8.246	8.248	7.892	7.778	**7.628**
LCMC-F	266,266	9.310	8.632	8.922	8.372	8.332	8.326	7.966	7.878	**7.740**
LCMC-G	452,511	9.626	8.652	9.048	8.316	8.322	8.312	7.962	7.896	**7.776**
LCMC-H	208,035	7.016	6.238	6.458	6.090	5.996	5.976	5.750	5.756	**5.666**
LCMC-J	509,095	8.220	7.352	7.694	7.104	7.058	7.034	6.754	6.802	**6.702**
LCMC-K	159,118	9.656	9.004	9.332	8.632	8.692	8.688	8.296	8.094	**8.030**
LCMC-L	137,743	9.530	8.916	9.236	8.626	8.620	8.614	8.238	8.006	**7.934**
LCMC-M	35,159	9.472	9.150	9.326	8.946	8.756	8.786	8.464	8.086	**8.020**
LCMC-N	155,897	9.482	8.870	9.138	8.530	8.560	8.564	8.176	8.016	**7.940**
LCMC-P	273,109	9.460	6.256	5.430	6.038	6.394	6.424	5.892	5.856	**5.482**
LCMC-R	49,289	9.254	8.822	9.126	8.546	8.484	8.490	8.162	7.848	**7.802**
Average		9.100	8.378	8.634	8.120	8.050	8.05	7.718	7.578	**7.350**

experimental model. We use an order 2 model as the maximum encoding order for both the symbol stream and the order stream.

The source code of our implementation is available at http://www.informatics.bangor.ac.uk/ ~wjt/AIIA/BMCC/PPMO.tar.gz. The arithmetic coder for our implementation was obtained from RWTH Aachen University (Bodden *et al.* 2004). The configuration of our test machine is Intel Pentium M 1.7 GHz, with 512 MB internal memory. We also use the measure *bits per Chinese character byte* (bpCc) to compare results, as the notion of a Chinese character is problematical and is dependent on the coding scheme used.

Table 4 shows the results for the SBMCC when using different compressors. It is clear that our variant PPMO achieves the best compression rate in all cases. On average, it is about 3% to 4% better than the second-place compressor PPM-Ch and about 5% to 6% better than the third-place compressor PPMII.

Table 5 shows the compression results for the LBMCC. PPMO still performs best on average. It is approximately 1% better than the second place compressor PPMII. Experimental results show that if we use an order 3 context model as a maximum order for the order stream, we achieve further improvements, with the average compression rate reducing to 5.860 bpCc.

Table 6 shows the compression results by using different orders for the order stream in the PPMO model. For example, PPMO 2+4 shows the compression result

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

Table 5. *Chinese text compression for Large Bangor Mandarin Chinese Corpus using different compressors (best result is shown in bold)*

File	Size (bytes)	*gzip* (bpCc)	*bzip2* (bpCc)	*WinRAR* (bpCc)	*ABC* (bpCc)	*PPMC* (bpCc)	*PPMD* (bpCc)	*PPMII* (bpCc)	*PPM-Ch* (bpCc)	*PPMO* 2 + 2 (bpCc)
TREC1	15,371	8.220	6.382	6.718	6.248	5.890	5.882	5.856	**5.720**	5.814
TREC2	1,348,941	8.374	6.818	7.044	6.292	6.196	6.168	**5.966**	6.026	5.984
TREC3	1,171,931	8.286	6.808	6.972	6.354	6.238	6.226	5.994	6.056	**5.986**
TREC4	1,034,517	8.350	6.788	7.060	6.394	6.268	6.246	6.034	6.076	**6.022**
TREC5	1,478,840	8.204	6.670	6.826	6.174	6.082	6.064	**5.854**	5.924	5.870
Chinese Bible	2,032,066	7.364	6.092	6.288	5.522	5.578	5.552	**5.380**	5.530	5.444
PH1	3,259,284	8.564	7.082	6.802	6.176	6.168	6.136	6.218	6.034	**5.984**
PH2	3,466,810	8.692	7.192	6.924	6.272	6.268	6.236	6.320	6.128	**6.080**
PH Corpus	7,506,581	8.604	7.102	6.650	6.120	5.918	5.880	6.258	6.304	6.256
Average		8.296	6.770	6.810	6.172	6.068	6.044	5.944	5.978	**5.938**

Table 6. *Chinese text compression for selected files from the Bangor Mandarin Chinese Corpus using different orders for the order stream*

File	Size (bytes)	*PPMO* 2+1 (bpCc)	*PPMO* 2+2 (bpCc)	*PPMO* 2+3 (bpCc)	*PPMO* 2+4 (bpCc)	*PPMO* 2+5 (bpCc)
LCMC-A	271,053	7.795	7.778	7.776	7.779	7.792
Chinese Bible	2,032,066	5.454	5.444	5.440	5.438	5.438

Table 7. *Execution times for PPMD, PPMII, and PPMO encode and decode by using exclusions and no exclusion*

		PPMO with exclusions		PPMO no exclusions		PPMD	PPMII
File	Size (bytes)	*Encode* (s)	*Decode* (s)	*Encode* (s)	*Decode* (s)	*Encode* (s)	*Encode* (s)
LCMC-A	271,053	27.5	17.5	4.1	6.8	17.5	0.22
Chinese Bible	2,032,066	185	125	25.9	40.3	70	0.73

by using an order 2 model for the symbol stream and order 4 model for the order stream. We select one small file and one big file to illustrate thus; results are similar for other files in the corpus. It is clear that the compression results are comparable across orders 1–5 for the order stream.

Table 7 shows the encode and decode execution times for PPMD, PPMII, and PPMO by using exclusions and no exclusions. It is clear that PPMII is much faster than any other PPM-based compressors. However, this well-crafted implementation has been heavily optimized for speed, whereas the focus for our implementation

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

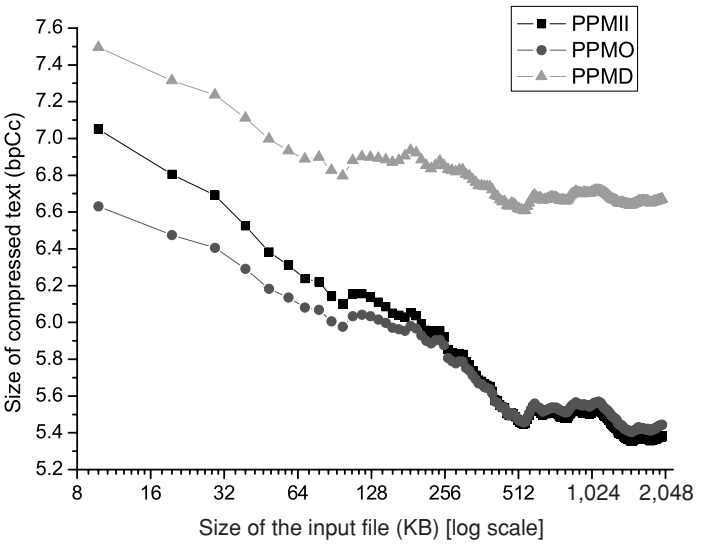


Fig. 3. Size of the output compressed file for PPMO, PPMII, and PPMD as the text grows.

Table 8. *Percentage of the symbol and order stream in the PPMO output compressed file*

File	Original size (bytes)	Compressed size (bytes)	*Symbol stream* (%)	*Order stream* (%)
LCMC-A	271,053	131,767	81.7	18.3
Chinese Bible	2,032,066	691,437	80.6	19.4

has been on compression performance. Nevertheless, the coding speed of PPMO with no exclusions is approximately seven times faster than that of PPMO with exclusions, and approximately three to four times faster than the traditional PPMD model. Moreover, experimental results of PPMO with no exclusions indicate a 2% to 3% drop in compression rate, but this is still much better than most of the other compressors with better execution speed.

Figure 3 shows how the size of the compressed text increases as the input text grows for PPMO, PPMII, and PPMD on the Chinese Bible. In the beginning, the compression rate of PPMO is 7% better than that of PPMII and almost 12% better than that of the PPMD, which shows that the PPMO model adapts more quickly. As the text grows, PPMII's compression rate improves quickly until finally it is slightly better than that of PPMO. Similar results occur for other files in the BMCC, although in most cases, PPMO ends up with the best overall compression (see Tables 4 and 5) unlike for the Bible text file that we have chosen for illustration purposes here.

Table 8 shows the separate percentage costs of encoding the symbol and order streams for the Bible file. It can be seen that the cost of encoding the symbol stream is approximately $\frac{4}{5}$ of the total cost, whereas that of the order streams is $\frac{1}{5}$.

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

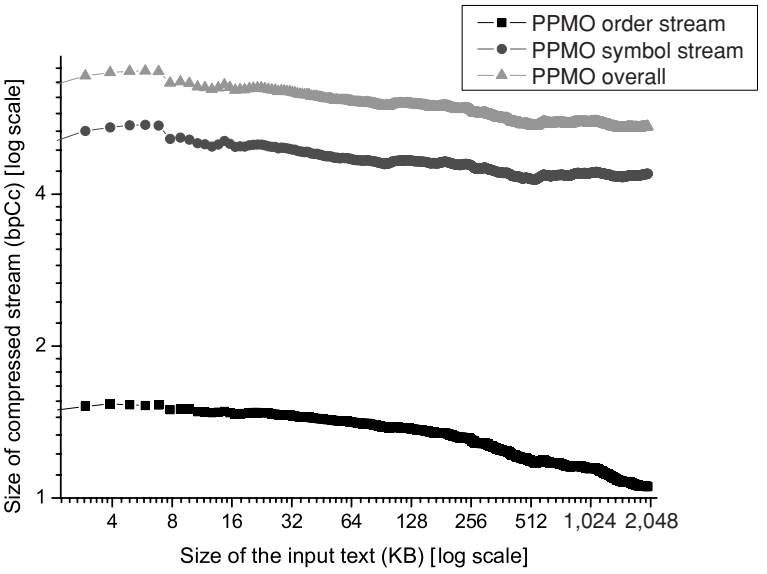


Fig. 4. The size of the symbol and order stream in the PPMO output compressed file as the text grows.

Figure 4 illustrates the size of the compressed symbol and order streams as the text grows. Again this is illustrated for the Chinese Bible text (and similar results are evident for the other files). As we have discussed in Section 4, the output size of the order stream increases much slower than that of the symbol stream because of its small alphabet size and repetitive nature. The graph shows a noticeable dip in the order stream compressed output with larger input text, whereas the symbol stream output and combined overall output has reached a plateau.

7 Discussion

We have described a new variant of the PPM compression algorithm designed especially for the Chinese language. The method uses a two-step encoding process. First, we encode the order of the model directly, then we encode the symbol itself. To our knowledge, this is the first empirical PPM algorithm that does not use the traditional PPM escape probabilities blending algorithm. No other study has successfully used a second-order encoding process to process the text.

We have also created a compression corpus that contains a sampling of GB encoded Chinese text documents. The purpose of the corpus is to standardize future experiments with Chinese text compression. As the experiments show in Section 6, PPMO outperforms other approaches on compressing the text in the corpus. This algorithm gained an excellent compression rate of 7.350 bpCc for the small files and 5.938 bpCc for the large files.

Some other language models such as the *Interpolated Language Model* (Jelinek 1985) are widely used in the speech recognition area. These models assign a weight to each order model, and then calculate the weighted sum of the probabilities.

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

Different interpolation algorithms such as held-out interpolation (Jelinek 1985) and word clustering interpolation (Bassiou and Kotropoulos 2005) have been devised on the basis of this idea. Our new PPMO blending mechanism could be used in this area as well. Hence, it is not only of interest to the PPM text compression and language modeling community but also of interest to the wider language modeling community where the adaptive mechanism for encoding the orders may be applicable. The advantage of the coding method that PPMO uses is that it is adaptive—in other words, the 'weights' are computed on the fly and so a streaming process is possible rather than an off-line process being required to compute the weights via some optimization method.

In addition, this algorithm should achieve similar improvements in other large alphabet- size languages such as Japanese, Korean, and Thai.

Further work in this area will include an optimization of the coding scheme. It is not clear how the improvements we have detailed here would affect other PPM-based implementations. But it does highlight that there are alternatives to the traditional PPM coding algorithm that may be more appropriate for text (such as Chinese text) that has different properties in the standard text (i.e, English) usually reserved for experimental evaluation.

References

Bassiou, N., and Kotropoulos, C. 2005. Interpolated distanced bigram language models for robust word clustering. In *International Workshop on Nonlinear Signal and Image Processing (NSIP 2005)*.

Bell, T. C., Cleary, J. G., and Witten, I. H. 1990. *Text Compression*. Upper Saddle Rivee, NJ: Prentice Hall.

Bodden, E., Clasen, M., and Kneis, J. 2004. *Arithmetic Coding revealed. Germany: RWTH Aachen University*.

Burrows, M., and Wheeler, D. J. 1994. A block-sorting lossless data compression algorithm. *Technical Report 124, Digital Equipment Corporation, Palo Alto, CA*.

Cheng, K.-S., Young, G. H., and Wong, K.-F. 1999. A study of word-based and integral-bit chinese text compression algorithms. *Journal of the American Society for Information Science* **50**(3):218–228.

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

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

Gu, H. Y. 1995. Large-alphabet Chinese text compression using adaptive Markov model and arithmetic coder. *Computer Processing of Chinese and Oriental Languages* **9**(2):111–124.

Gu, H.-Y. 2005. A large-alphabet-oriented scheme for Chinese and English text compression. *Software—Practice and Experience* **35**:1027–1039.

Jelinek, F. 1985. Self-organized language modeling for speech recognition. In *Readings in Speech Recognition, A. Waibel and K. Lee (eds.),* Morgan Kaufmann, Weshington, DC vol. 28, pp. 2591–2594.

Jin, G. 1992. PH Corpus of Mandarin Chinese. ftp://ftp.cogsci.ed.ac.uk/pub/chinese. Date accessed 30 June, 2007.

Lua, K. T. 1994. Compression of Chinese text. In *International Conference on Chinese Computing*. pp. 367–375.

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press

McEnery, T., and Xiao, R. 2004. The Lancaster Corpus of Mandarin Chinese. European Language Resources Association. http://bowland-files.lancs.ac.uk/corplang/lcmc/. Date accessed 30 June, 2007.

Moffat, A. 1989. Word-based text compression. *Software Practice and Experience* **19**(2):185–198.

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

Moffat, A., Neal R., and Witten, I. 1998. Arithmetic Coding Revisited. *ACM Transactions on Information Systems* **16**(3):256–294.

Ong, G. H., and Ng, J. P. 2005. Dynamic Markov Compression using a crossbar-like tree initial structure for Chinese texts. In *ICITA '05: Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)*, vol. 2, pp. 407–410.

Shkarin, D. 2002. PPM : One step to practicality. In *Data Compression Conference 2002*. pp. 202–211.

Teahan, W. J. 1998. *Modelling English Text, PhD thesis*. New Zealand: University of Waikato.

Teahan, W. J., and Harper, D. J. 2001. Combining PPM models using a text mining approach. In *Data Compression Conference 2001* pp. 153–162.

TREC Mandarin Corpus. 2000 Text Retrieval Conference test data. http://www.ldc.upenn.edu. Date accessed 30 June, 2007.

Vines, P., and Zobel, J. 1998 Compression techniques for Chinese text. *Software—Practice and Experience* **28**(12):1299–1314.

Witten, I. H., Bray, Z., Mahoui, M., and Teahan, W. J. 1999. Text mining: A new frontier for lossless compression. In *Data Compression Conference 1999*. pp. 198–207.

Wu, P., and Teahan, W. J. 2005. Modelling Chinese for text compression. In *Data Compression Conference, 2005, Proceedings. DCC 2005*. p. 488.

https://doi.org/10.1017/S1351324907004597 Published online by Cambridge University Press