Adaptive Models of Chinese Text

Submitted by Peiliang Wu

for the degree of

Doctor of Philosophy

University of Wales, Bangor 2007

To my Parents and Rui

Abstract

This thesis is concerned with how to build adaptive language models of Chinese text, which can be used in different Chinese natural language processing (NLP) applications.

The Prediction by Partial Matching (PPM) language model has been widely used in many NLP areas. To apply this model for Chinese, many problems arise that originate from the language's large alphabet. In this thesis, the PPM-ch model is introduced first to improve the traditional PPM model, by first using preprocessing techniques, then a frequency sorting technique and a variation of PPM that performs no exclusions.

PPMO, a novel variant of the PPM model is then proposed. Unlike traditional PPM models, which output an escape symbol when a novel symbol occurs in a context model, PPMO separates the coding process into two streams, named the orders stream and the symbols stream. This algorithm is the first PPM variant that does not use the escape mechanism and it achieves the best compression results.

We have also investigated Chinese Word segmentation by using our PPM-ch and PPMO model. Although our PPMO models havenot been carefully crafted for segmentation usage, we still achieve satisfactory results.

Acknowledgments

My deep gratitude is dedicated to my supervisor Dr. W. J. Teahan, for his excellent supervision, enthusiasm and encouragement, also for his patience in allowing me many research freedoms to try some different "strange" ideas. I feel extremely fortunate to work with Dr. Teahan, especially after having some difficulties in the first year of my study. Without his devoted effort, completion of this work would not have been possible.

I would also like to thank the school of Informatics for providing some financial support for my study.

A large number of individuals in the School of Informatics, who have contributed either directly or indirectly to the completion of this work, are sincerely thanked. My special thanks are due to Dr. Jianming Tang for his valued suggestions on writing up the thesis.

I would like to take this opportunity to thank my parents, for their love, understanding and encouragement during the years when I was pursuing the PhD degree. Deepest emotions are to my girlfriend Rui; thank you very much, for your best love. We now both have finished our PhD degree, and will marry this spring.

Contents

Figure

Table

1 Introduction

Contents

1.1 Background & Motivation

In this information revolution age, huge amounts of information are stored and processed electronically by the computer. The need for processing languages using computers has never been more urgent.

Natural language processing (NLP) is a science of automatically generating and understanding natural human languages using computers. There are many NLP research areas such as text compression, speech recognition, word segmentation and machine translation and so on. Normally, language models will be developed for different research usages and these models are regarded as the real essential ingredient for natural language processing.

There are two main schemes to building models for natural language. One is a dictionary approach; the other is a statistical approach. To evaluate the performance of the models, text compression is one excellent measure. Experiments show, although the program execution speed of dictionary approaches is fast, the compression rate is much worse than that of the programs that use statistical algorithms. That is why in this thesis we only focus on the statistical algorithms for Chinese language, as we are interested in finding the models with the best predictive performance (as measured by compression codelength) as this usually correlates with better application performance in natural language processing applications. One technique that has achieved excellent results is Prediction by Partial Matching model, or PPM [CW84], an adaptive statistical language model. Unlike many static models that use dictionary algorithms, an adaptive model is built and upgraded dynamically according to the input stream of characters.

The Chinese language has its own characteristics that differ from that of the English language. When processing Chinese language with traditional statistical models such as PPM [CW84], there are several drawbacks for this algorithm:

Firstly, each English symbol requires an 8-bits ASCII character to encode it on the computer; that is, its alphabet size is only 256. However, for the Chinese language, each character requires 16-bits, with an alphabet size up to 65536. Traditional PPM algorithms use the smaller 256 alphabet size, and require two bytes to encode each Chinese character. Therefore this scheme does not capture the characteristics of the larger Chinese alphabet.

Secondly, the PPM model uses the *escape* mechanism to avoid sparse data problems (see section 3.5), which needs to estimate the probability of an escape to a lower order model, and we have found that for Chinese this actually lowers the prediction accurateness. Our experiments show that this greatly reduces the overall compression rate, especially for a large alphabet size language (i.e. Chinese).

Thirdly, the PPM execution time for the Chinese language is much slower on average compared to English. Simply enlarging the alphabet size for PPM algorithms significantly reduces the program execution speed as well as the overall compression result due to the need to perform exclusion.

1.2 Thesis Aims & Objective

The broad aim of this thesis is to investigate how to set up adaptive computer models of Chinese text that are effective in terms of compression performances, execution speed and memory requirements. As we have discussed in the previous section, although existing adaptive statistical language models based on the text compression schemes (such as PPM algorithms [CW84]) achieve excellent results, there are several notable drawbacks when applying for Chinese language. These drawbacks originate from the linguistic differences between Chinese and English language, since Chinese has its own specific characteristics.

By customizing the models for Chinese text, one would expect, significant improvements over current implementations are possible. Several potential benefits for the Chinese language models are therefore gained, which yields a better performance in many Chinese NLP applications such as text compression and word segmentation.

The specific objectives of this thesis will be to seek answers to the following questions:

- What is the best computer model for compressing Chinese text?
- What are the disadvantages of the existing models when they are applied to Chinese text?
- What is the best way of dealing with the large alphabet size problem in Chinese?
- How well do the best models perform in several natural language processing applications?

1.3 Thesis Contributions

This thesis is concerned with adaptive models of Chinese language. The main contribution of this thesis is proposing two new language models designed especially for the Chinese language. These models are named PPMO and PPM-ch respectively. Significantly, PPMO is the first adaptive PPM model without using the traditional PPM escape mechanism.

The objectives listed in the previous section are achieved via the development of the PPM-ch and PPMO language models especially for Chinese text (see chapter 4 & 5). Additionally, we show that the PPM-ch and PPMO models work well for the Chinese word segmentation and have great potential to be used in many other Chinese NLP applications such as text categorization and text mining.

Many Chinese NLP applications can therefore make use of these algorithms to potentially improve application performance. Moreover, the new algorithm has a great potential to be applied to other languages and for word-based English compression.

In the following, the significance of the results obtained from the investigations are described.

1. We introduce preprocessing techniques for Chinese text. Unlike the traditional PPM model, which uses 8 bit ASCII characters to code the 16 bit Chinese text, we pre-process the Chinese characters into 16 bit integers before encoding them. This part of the work is discussed in

Chapter 4 and has been published in DCC2005 [WT05].

2. For standardizing further experiments in Chinese text compression, we have set up a standard Mandarin Chinese Corpus, which is publicly available at *http://aiia.cs.bangor.ac.uk/BMCC*. This work is introduced in Chapter 2 [WT05].

3. We adapt the PPM model for the Chinese language, by using such techniques such as character pre-processing, frequency sorting and no exclusions, achieving competitive compression results for Chinese text. This work is discussed in Chapter 4 [WT05].

4. In Chapter 5, we introduce a new PPM variant, dubbed PPMO, which separates the coding stream into an orders stream and symbols stream. It achieves excellent experimental results. To our knowledge, this is the first empirical PPM algorithm that does not use the traditional PPM escape probabilities blending algorithm. It has not been reported by the other study, which successfully used a secondary order encoding process to process the text. This part of the work is discussed in Chapter 5 [WT07].

5. We introduce word and tag based model for Chinese text, by using our PPMO and PPM-ch model. Our experiments show that character-based

PPMO still achieves the best performance. We discuss this in Chapter 7.

6. We investigate a natural language processing application — Chinese word segmentation — that uses our new PPMO models. We discuss this in Chapter 7.

Summary and conclusions are discussed in the final chapter along with suggestions for future work.

2

Chinese Language Overview

Contents

2.1 Chinese: The World's Most Widely Spoken Language

Chinese (汉语 hàn yǔ) is a primary language which is part of the Sino-Tibetan language family. It is the most widely spoken language in the world. According to the Guinness World Records 2006 [Fol06], more than 1.4 billion, or one-fifth of the people in the world speak some form of Chinese as their native language. This is mainly because China is the most populous nation in the world.

In general, all varieties or dialects of Chinese are tonal and analytic. However, spoken Chinese shows a high level of internal diversity. Regional variation between different variants is comparable in many respects to the Romance language family; many variants of spoken Chinese are different enough to be mutually incomprehensible [DeF84].

Depending on different classification schemes, there are six to twelve main regional groups of Chinese, of which the most populous is Mandarin (普通话 pǔ tōng hùa c. 800 million), followed by Wu (吴 wú c. 90 million), and Cantonese (粤 yùe; c. 70 million).

The standardized spoken Chinese, or *Standard Mandarin* (普通话 pǔ tōng hùa 国语 gúo yǔ 华语 húa yǔ), is based on the Beijing dialect, a member of the Mandarin group. Standard Mandarin is the official language of the People's Republic of China and the Republic of China on Taiwan, as well as one of four official languages of Singapore (together with English, Malay, and Tamil). Standard Mandarin Chinese is one of the six official languages of the United Nations (alongside English, Arabic, French, Russian, and Spanish). Spoken in the form of Standard Cantonese, Chinese is one of the official languages of Hong Kong (together with English) and of Macaw (together with Portuguese).

Position	Language	Script used	Speakers (millions)	Major Spoken region
1	Mandarin	Chinese Characters	1051	People's Republic China (Mainland China, Hongkong, Macau) Republic of China(Taiwan)
2	English	Latin	508	USA, UK, Australia, Canada, New Zealand
3	Hindi	Devanagari	497	North & Central India
4	Spanish	Latin	392	The Americas,Spain
5	Russian	Cyrillic	277	Russia,Central Asia
6	Arabic	Arabic	246	Middle East, Arabia North & Southern Africa
7	Bengali	Bengali	211	Bangladesh, Eastern India
8	Portuguese	Latin	191	Brazil, Portugal
9	Malay, Indonesian	Latin	159	Indonesia, Malaysia
10	French	Latin	129	France, Canada

Tab. 2.1: The 10 most widely spoken languages in the world.

The data in Table 2.1 shows those languages that are spoken by the most people in the world[Soy05]. Mandarin, a chief dialect of China with over one billions speakers, is the world's most wildly spoken language. English is the second most widely spoken language in the world after Mandarin.

The majority of Chinese speakers live in eastern and southern Asian areas, such as the People's Republic of China (Mainland China, Hong Kong, Macau) and Republic of China (Taiwan and nearby islands). There are also many Chinese communities in Western Asia, the Americas, Africa, Europe and Pacific.

2.2 Chinese Characters: Written Language for Chinese

Chinese characters are one of the oldest surviving writing systems, and its history can be traced back into at least 3,200 years ago [Wil86]. Archaeologists have also found various Neolithic scripts in China, which date back as early as the 7th millennium BC, and these scripts are normally regarded as the genesis of the Chinese characters. Various Asian countries such as Japanese, Korean, and Vietnamese adopt Chinese characters as part of their own written languages.

A Chinese character is a logogram and is normally categorized into

3 different types according to its character making mechanism: pictographic character(象形字 xiàngxíngzì), logical aggregate character(会意字 huìyìzì) and pictophonetic character(形声字 xíng-shēngzì).

About four percent of Chinese characters are pictographic characters, which have originated from individual pictograms. Most Character characters are pictophonetic, which means the character contains two parts: one indicating a general category of meaning and the other representing the sound or pronunciation of this character. Some other characters are logical aggregates, which are characters combined from multiple parts indicative of meaning [Wil86].

In the Chinese writing system, each individual single-syllable morpheme corresponds to a single character. Some of these single-syllable morphemes can stand alone as individual words, but most words in the modern spoken Chinese varieties are in fact multisyllabic, consisting of more than one morpheme, usually two, but there can be three or more. Although Chinese language shows a great diversity in the spoken language, different Chinese people with different dialects can still use the same written language to communicate without any problems.

"Square-Block Characters" (方块字 fāngkuàizì) is normally used to illustrate a Chinese character because in written Chinese a character made up of multiple parts compacts these parts together in order to maintain a uniform size and shape.

The actual structure of many Chinese characters varies in different cultures. Since 1956, P.R. China (Mainland China) uses simplified characters as its written language; however traditional Chinese characters are still used in Taiwan and Hong Kong areas.

2.3 Chinese Encoding Method

The Chinese language employs Chinese characters (汉字 hànzì) as its written language. Each character represents a morpheme (a meaningful unit of language), as well as one syllable. The written language can thus be termed a morpheme-syllabic script.

Mainly, there are three Chinese coding standards to represent Chinese Characters in the computer – GB2312-80 and Big5 as mentioned, and Unicode.

For computers, Chinese characters are usually encoded by using a more specific character set such as GB2312-80 or Big5, which maps between numbers and Chinese characters. This is comparable with the 7-bit ASCII character set used for English, the major difference being that the Chinese character set is much larger than that of the English's. ($\sim$ 8,000 compared to 128.)

2.3.1 GB2312-80 and GBK Coding Standard

The *GB2312-80* standard font was published in 1981 by the Standardization Adminstration of the People's Republic of China. It is a national standard encoding character set for Simplified Chinese and used in Mainland China and Singapore. GB2312-80 includes 7,445 Simplified Chinese characters and symbols. Every Chinese character or symbol is represented by 2 bytes and each byte is from 0xA0 to 0xFE in the ASCII table.

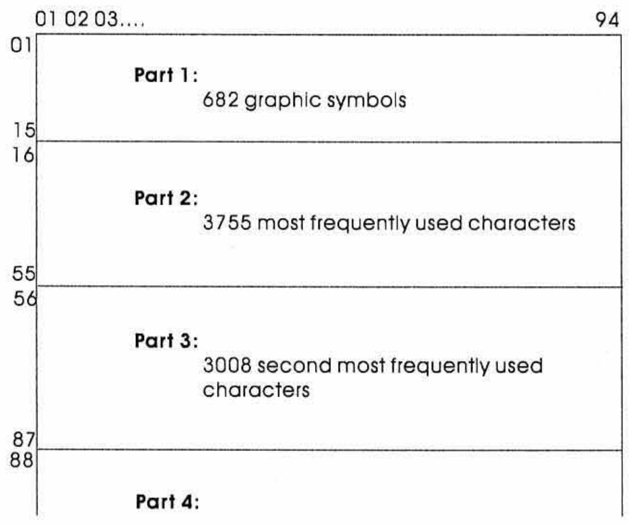
01 02 03....
94
01
Part 1:
682 graphic symbols
15
16
Part 2:
3755 most frequently used characters
55
56
Part 3:
3008 second most frequently used characters
87
88
Part 4:

Fig. 2.1: Chinese GB2312-80 Coding table

Figure 2.1 shows the GB2312-80 character set. It has 94 rows (high byte) and 94 columns (low byte), which is divided into four parts. Part one includes 682 graphic symbols, part two contains 3,755 Simplified Chinese characters sorted by usage frequency in Chinese text. Part three has 3,008 characters which is sorted by their stroke count in Chinese language. Part four is a user-defined characters for the further extension.

The *GBK* standard font is an extension of GB2312-80 which has 20,902 Chinese characters and symbols. It is upward compatible with the GB 2312-80 standard.

2.3.2 Big5 Coding Standard

The *Big5* standard font was defined by the Institute for Information Industry of Taiwan (Republic of China) in 1984 and mainly used in Taiwan and Hong Kong regions as a character encoding set for Traditional Chinese characters. It contains 13,868 Traditional Chinese characters and symbols. Every character is also encoded by 2 bytes, the coding range is from 0xA1 to 0x9 for the first byte, and from 0x40 to 0x7E as well as from 0xA1 to 0xFE for the second byte.

Similar to the GB2312-80 coding standard, the Big5 character set is divided into seven parts including some spaces reserved for user-defined characters. All the characters are sorted by their usage frequency.

2.3.3 Unicode Coding Standard

The Unicode coding standard was defined by the International Organization for Standardization (ISO) in 1991, and allows text and symbols from all of the writing systems of the world to be consistently represented and manipulated by computers.

This encoding character has defined 20,902 CJK (Chinese, Japanese and Korean) characters. It is a superset of the characters in GB2312-80 and Big5. The advantage of using this standard is that you can display Simplified Chinese characters, Traditional Chinese characters, Korean characters and Japanese characters on the same Web page. No other encoding standards is able to support that at the moment. However, the Unicode coding standard is still not widely being used in different countries.

2.4 Chinese Language Corpora

A corpus (plural *corpora*) is a body "of natural language material (whole texts, sample from texts, or sometimes unconnected sentences), which are stored in machine-readable form" [Tea98]. Statistical natural language processing uses corpora to do statistical analysis, checking occurrences or validating linguistic rules on specific domains.

In this section, we first review several Chinese language corpora, then introduce our Bangor Mandarin Chinese Corpus, an experimental corpus that will be used in the experiments of this thesis.

2.4.1 The Lancaster Corpus of Mandarin Chinese

The Lancaster Corpus of Mandarin Chinese (LCMC) is a publicly available *balanced* corpus of Mandarin Chinese, which was built up by Tony McEnery and Richard Xiao from Lancaster University in the United Kingdom. A balanced corpus includes a range of the different text types of the language, with their proportions of the corpus reflecting, in some more-or-less principled way, their levels of use in the language community at large.

LCMC contains 15 different text categories. The corpus comprises Mandarin Chinese text published in Mainland China, including newspapers, books, and also text from the Xinhua News Agency.

Words in the LCMC have been have been encoded into Unicode, tagged and stored in the XML files. For our experimental usage, we have made some changes and formatted the files as several normal flat text files.

Further discussion of this is provided in section 2.4.4.

2.4.2 PH Mandarin Corpus

The Chinese PH Corpus is a Mandarin Chinese corpus originally constructed by Guo Jin [Jin93]. It is not a balanced corpus because the source of this corpus is only news text from the Xinhua news agency, which was written between January 1990 and March 1991.

The corpus is encoded by the GB2312-80 coding standard. Hockenmaier and Brew produced a cleaned up segmented version of it [HB98]. The corpus contains 2,447,719 words and 3,753,291 characters, 492,875 of which are paragraph delimiters. The corpus is freely downloadable (ftp://ftp.cogsci.ed.ac.uk/pub/chinese/).

2.4.3 Text Retreival Conference Mandarin Corpus

The TREC (Text REtreival Conference) Mandarin Corpus [Rog00] is available from the Linguistic Data Consortium (LDC), which is an open consortium of universities, companies and government research laboratories. The catalogue number is LDC2000T52 with ISBN 1-58563-178-7. These documents were used for the Chinese task in the Text Retrieval Conference (TREC) 5 and 6 and consist of approximately 170 megabytes of articles drawn from the Xinhua News agency as well as the People's Daily newspaper (Rogers 2000). The text is GB2312-80 encoded Mandarin Chinese.

The collection of text was gathered by LDC, and then adapted by the National Institute of Standards and Technology (NIST) for use in the TREC Mandarin evaluation program.

2.4.4 Bangor Mandarin Chinese Corpus

In order to standardize future text compression experimental results in our thesis, we have reformed the above corpora and included them with a corpus named *Bangor Mandarin Chinese Corpus* (or BMCC) which we have set up for experimental usage. It can be downloaded from the following Internet address: *http://www.informatics.bangor.ac.uk/~wjt/AIIA/BMCC/BMCC.htm.*

According to the file size, the whole corpus is divided into two sub corpora. 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 under 800 KB. As shows in table 2.2, we select some news, novels, and fictional articles. We also include a balanced Mandarin Chinese Corpus named LCMC [MX04], 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 15

Name	**Category**	**Size**
	Small Bangor Mandarin Chinese Corpus	(bytes)
Chinese news1	Political News from XinHua news agency	15,371
Chinese news2	Sport News from XinHua news agency	26,195
Chinese article1	Article about military affairs from Internet	19,245
Chinese article2	Article about computer from Internet	38,724
Chinese article3	Article about arts from Internet	80,760
Chinese book1	Novel by Lu Xun	53,706
Chinese book2	Novel by Qian Zhongshu	436,656
LCMC-A	Press: reportage	271,035
LCMC-B	Press: editorials	170,469
LCMC-C	Press: reviews	112,681
LCMC-D	Religion	103,031
LCMC-E	Skills, trades and hobbies	218,473
LCMC-F	Popular lore	266,226
LCMC-G	Biographies and essays	452,511
LCMC-H	Miscellaneous: reports and official documents	208,035
LCMC-J	Science: academic prose	509,095
LCMC-K	General fiction	159,118
LCMC-L	Mystery and detective fiction	137,743
LCMC-M	Science fiction	35,159
LCMC-N	Adventure and martial arts fiction	155,897
LCMC-P	Romantic fiction	273,109
LCMC-R	Humour	49,289

Tab. 2.2: Small Bangor Mandarin Chinese Corpus.

different text categories. The corpus comprises Mandarin Chinese text published in Mainland China, including newspapers, books, and also 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.

Name	**Category**	**Size**
Large Bangor Mandarin Chinese Corpus		(bytes)
TREC1	Text Retrieval Conference test data	818,786
TREC2	Text Retrieval Conference test data	1,348,941
TREC3	Text Retrieval Conference test data	1,171,931
TREC4	Text Retrieval Conference test data	1,034,517
TREC5	Text Retrieval Conference test data	1,478,840
Chinese Bible	Bible, Chinese version data	2,032,066
PH1	Part of PH Corpus	3,259,284
PH2	Part of PH Corpus	3,466,810
PH Corpus	PH Corpus	7,506,581

Tab. 2.3: Large Bangor Mandarin Chinese Corpus.

Table 2.3 shows Large Bangor Mandarin Chinese Corpus. It contains all files above 800 KB. We select the Chinese Bible, and include some TREC Corpus [Rog00] files and the PH Corpus [Jin93]. All TREC files come from the XinHua news agency that were collected for the Text Retrieval Conference. We have concatenated all the TREC files into a single file, then removed all the XML tags, and selected 5 partitions from the head of that file (these are split according to story boundaries).

2.5 Theoretical Models For Chinese Text

In this section, we review some theoretical models for Chinese text. We examine the characteristics of Chinese text by looking at the frequency of Chinese characters and show how Zip's Law can be used to predict the number of character occurrences for a given size of Chinese text.

2.5.1 Character Frequency

Compared with 26 letter English alphabets, many non-native language speakers might be surprised that the alphabet size of Chinese language is more than 80,000 Chinese characters. However, today most of these Chinese characters are rarely used; now only about 8,000 characters will be normally used in modern communications.

Chinese Characters	**Coverage Rate**
Most frequently used 1,000 characters	$\sim$ 90%
Most frequently used 2,500 characters	98%
Most frequently used 3,500 characters	99.5%

Tab. 2.4: The coverage rate of the most frequently used characters in Chinese text.

Table 2.4 shows the coverage rate of the most frequently used characters in Chinese text. Statistics show that the most frequently used 3500 characters cover 99.5% of the Chinese characters commonly used

[Abo07].

PH Corpus			*Chinese Bible*			*LCMC*		
character	frequency	%	character	frequency	%	character	frequency	%
的	94052	1.25293	的	46367	2.21709	的	54272	1.30322
国	39623	0.52784	他	20147	0.96335	一	20901	0.50189
一	29453	0.39236	你	18957	0.90645	是	16947	0.40694
在	27622	0.36797	我	18630	0.89081	不	14916	0.35817
和	25030	0.33344	们	18173	0.86896	了	14714	0.35332
了	24111	0.3212	人	16415	0.7849	在	13000	0.31217
中	23561	0.31387	在	12365	0.59125	人	12652	0.30381
年	23458	0.3125	和	11735	0.56112	有	12117	0.29096
人	23249	0.30971	是	11292	0.53994	这	9241	0.2219
会	20634	0.27488	耶	9810	0.46908	我	9169	0.22017
大	18474	0.2461	说	9096	0.43493	他	8460	0.20315
有	17517	0.23336	不	8991	0.42991	个	8285	0.19895
是	15939	0.21233	要	7951	0.38019	上	8256	0.19825
工	15501	0.2065	有	7903	0.37789	和	8046	0.19321
为	15023	0.20013	就	7878	0.3767	中	8037	0.19299
上	14270	0.1901	以	7798	0.37287	大	7599	0.18247
这	14086	0.18765	了	7525	0.35982	来	7185	0.17253
业	13574	0.18083	为	7524	0.35977	地	7124	0.17107
民	13389	0.17836	子	7274	0.34781	为	6910	0.16593
地	13379	0.17823	一	6898	0.32984	国	6734	0.1617
Overall		6.42029			12.56268			6.1128

Tab. 2.5: 20 most frequently used Chinese Characters from 3 Chinese Corpora

Table 2.5 shows the 20 most frequently used Chinese Characters from 3 Chinese corpora. We select the PH corpus, Chinese Bible and LCMC from our BMCC (see Section 2.4.4). It is clear that “的” (*prep.* of) is the most frequently used Chinese character in all three corpora. Also, several characters such as "一" (*num.* one), "人" (*noun* people) also occur in all three corpora, but compared to English, the most notable difference is that there are a significant number of characters that only occur in one corpus even in such a short list of 20 as here (5 in PH corpus, 8 in Chinese Bible, 2 in LCMC).

For the Chinese Bible, the cumulative frequency of the first 20 most frequently used characters is 12.56%. For the PH corpus and LCMC, the 20 most frequently used characters occupy 6.42% and 6.11% respectively.

2.5.2 Zipf's law

Zipf's law was first introduced by George Zipf [Zip49]. This law describes a statistical regularity in the distribution of words used in any large text and is normally regarded as an empirical rule for characterizing any natural languages.

Let f be the frequency of a word in a given text, and r be the rank of the word according to the frequency f of its occurrence. So, the most frequently occurring word would be given the rank $r = 1$, the next most frequently occurring word, $r = 2$ and so on. Zipf's law states that

$$f(r) \propto \frac{1}{r}$$

or, in other words

$$f(r) \times r = K$$

where K is a constant number.

This power law has been tested over a large volume of literature and also different languages and is found to be accurate for words whose rank is not too low or too high.

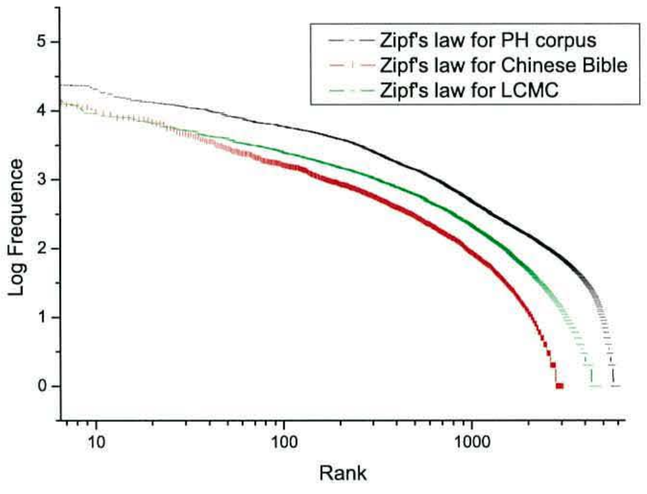

Fig. 2.2: Zipf's Law for Chinese corpora

For example, Figure 2.2 shows rank on the X-axis versus frequency on the Y-axis, using logarithmic scales. For all three Chinese corpora, the rank and frequency shows a linear slope within the first 1,000 Chinese characters, which indicates Chinese language initially follows the Zipf's law. The rank-frequency ratio decreases much faster after the first 1,000 characters. This is because, as we mentioned in the previous Section 2.5.1, the most frequently used 1,000 characters covers 90% of the whole corpus, but those ranked greater than 1,000 might only occur one or two times.

2.6 Conclusion

In this chapter, we reviewed several fundamentals of Chinese language. One of the most important differences between Chinese and English is that the alphabet of Chinese is much larger than that of English. The large alphabet leads to many problems for Chinese NLP. We will examine these problems in the next few chapters.

We introduced a Chinese language corpora, named BMCC. This corpus can be found at *http://www.informatics.bangor.ac.uk/~wjt/AIIA/BMCC/BMCC.htm*. The corpus provided samples for standardizing further experiments in Chinese text compression. We will use this corpus as our experimental corpora in this thesis.

We examined two theoretical models for Chinese text in the last section. According to the statistics, the most frequently used 3500 characters cover 99.5% of the Chinese characters commonly used and same as

English, the Chinese language also follows the Zip'f law.

3

Statistical Natural Language Processing

Contents

3.1 Introduction

Natural Language Processing (NLP) is an interdisciplinary field of artificial intelligence and linguistics that uses computer models to process the natural human languages. Normally it is named *computational linguistics* from the linguistical perspective. As a subfield of *applied linguistics*, the philosophical ideas of NLP are heavily influenced by the development of the linguistics science as well as the information science.

Between 1960 and 1985, NLP was typified by a rationalist approach. This approach is characterized by the belief that "a significant part of the knowledge in the human mind is not derived by the senses but is fixed in advance, presumably by genetic inheritance [MS99]." *Generative linguistics*, introduced by Chomsky in 1957 [Cho57], became the fundamental ideas in the rationalist linguistic community. This theory argues that the phrase structure of a language has its own *context-free grammars* and these grammars can be used to produce *formal languages*. Many dictionary based NLP algorithms are based on this approach.

However, in reality using context free grammars to characterize all language phenomena could be problematical because we have not only "well-formed" language, but also "ill-formed" language. An empiricist approach suggests that "we can learn the complicated and extensive structure of language by specifying an appropriate general language model, and then inducing the values of parameters by applying statistical, pattern recognition, and machine learning methods to a large amount of language use [MS99]."

In recent years, empiricist approaches have become more and more popular in many engineering practical solutions. These solutions seek methods that can work on raw text as it exists in the real word and is sometimes named "language engineering" instead of NLP. Statistical NLP characterize linguistic events as probabilistic phenomena, saying sentences are "usual" and "unusual". This approach has many great advantages, because they are "better at automatic learning, better at disambiguation, and also have a role in the science of linguistics. [MS99]."

In this thesis, our approach to processing Chinese language is based on the empiricist approach by using statistical methods. We will focus on how to statistical methods to set up a reasonable Chinese language model, for the usage of Chinese NLP.

Before adopting a Statistical NLP approach for Chinese language, we review several fundamental ideas, which are key to a statistical approach to NLP. We first introduce the statistical language model concept in section 3.2, then examine the ideas of entropy, cross-entropy and perplexity in section 3.3. Section 3.4 discusses n-gram models and Markov models. In Section 3.5 we illustrate the sparse data problem – a ubiquitous problems in statistical language modelling, and the ways to solve this problem. Then we discuss different variable order Markov models in Section 3.6. Section 3.7 reviews the prediction by partial match (PPM) model, one of the most compression-efficient algorithms used by the statistical language modelling community. We will apply this model as a starting point for Chinese NLP. In the last section 3.8, we examine different text compression algorithms.

3.2 Statistical Language Model

A statistical *language model* assigns a probability to a discrete sequence $P(x_1, \cdots, x_n)$ by means of a *probability distribution* [PC98]. One classical application of discrete sequence predictions is *lossless compression*, or more specifically *text compression*; but there are numerous other applications involving sequential data, which can be directly solved based on effective prediction of discrete sequences. Examples of such applications are speech recognition [JBM75, BJM83], machine translation [BCP+90] and automatic spelling correction [KCG90].

In general, comparison of different language models is made with respect to prediction quality as measured by the average compression rate of a discrete text [BYY04]. The best compression rate can be achieved by using *arithmetic coding* [WNC87], which allows the encoding of messages in the number of bits equal to its "information content", or *entropy* (see Section 3.3) with respect to a model.

So the problems of building a language model transform into a problem of how to get the best text compression rate. The better text compression rate you are able to achieve, the better prediction accuracy is made by the language model, which indicates a possible better performance for other NLP applications [BWC89].

A language model is an essential ingredient for statistical NLP. Witten & Bell made a statement about the importance of the language model [WB90]:

> A model of a natural language text is a collection of information that approximates the statistics and structure of the text being modelled. The model may be very simple, e.g. an estimate of the probability of each character; or it may be very complex, such as the model of English language that we carry in our heads,

with which we can spot subtle grammatical errors and spelling mistakes. Such models are of great importance in a number of areas, notably text compression, authorship ascription, and language-processing programs such as spelling-checkers.

3.3 Entropy, Cross-Entropy and Perplexity

The idea of *Entropy* was first introduced by Claude E. Shannon in his historical paper in 1948 [Sha48] and then became the fundamental idea in information theory.

Entropy is a measure of the amount of order in the message. It is a number that is small when there is a lot of order and large when there is a lot of disorder. Ideally, the length of a message after it is encoded should be equal to its entropy [BCW90].

Let $p(x)$ be the probability of a discrete sequence and A be a alphabet of this sequence. The value *entropy* E of X is given by the formula

$$E(X) = -\sum_{x \in A} p(x) \log_2 p(x). \tag{3.1}$$

For example, suppose there is a sequence of text from the alphabet (A, B, C) with the probabilities of $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{4}$. Then the average number of bits required to encode each symbol, or the average entropy is thus:

$$\begin{aligned} E &= -p(A)log_2p(A) - p(B)log_2p(B) - p(C)log_2p(C) \\ &= -\frac{1}{2}log_2\frac{1}{2} - \frac{1}{4}log_2\frac{1}{4} - \frac{1}{4}log_2\frac{1}{4} \\ &= 1.5 \text{ bits} \end{aligned}$$

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 a measure of the information content of the message – more probable messages convey less information than less probable ones.[Tea98]"

For language modelling, let $X = x_1, x_2, \cdots, x_n$ be a sequence of symbols, so equation 3.1 can be reformulated as:

$$E(X) = -\sum_{x_i \in A} p(x_1, x_2, \cdots, x_n) \log_2 p(x_1, x_2, \cdots, x_n) \tag{3.2}$$

and the *per word entropy* is:

$$\frac{1}{n}E(X) = -\frac{1}{n}\sum_{x_i \in A} p(x_1, x_2, \cdots, x_n) \log_2 p(x_1, x_2, \cdots, x_n). \tag{3.3}$$

If we assume that a language L is a stochastic process, then the *entropy of a language* is defined as:

$$E(L) = \lim_{n\to\infty} -\frac{1}{n}\sum_{x_i \in A} p(x_1, x_2, \cdots, x_n) \log_2 p(x_1, x_2, \cdots, x_n). \tag{3.4}$$

Normally, the true probability distribution of a language L is unknown. However, an upper bound to $E(L)$, or named *cross entropy* of a language L is defined by:

$$E(L, M) = \lim_{n \to \infty} -\frac{1}{n} \sum_{x_i \in A} p(x_1, x_2, \cdots, x_n) \log_2 m(x_1, x_2, \cdots, x_n). \tag{3.5}$$

where $m(x_1, x_2, \cdots, x_n)$ is the probabilities estimated by the model M.

The cross-entropy provides a measure of how well the language model is. Lower cross entropy normally leads to better performance in applications. In another words, the closer $E(L, M)$ is to $E(L)$, the less inaccurate the model is [Tea98].

Teahan achieved an average cross-entropy of English by using statistical language model such as PPM (see section 3.7) of 1.48 bpc for Jane Austen's Emma [Tea98]. In the next several chapters, we will examine the entropy of Chinese text by using different PPM-based models (see Section 3.7).

Another measure related to the entropy is called the *perplexity*. The relationship between the *perplexity* and the *cross entropy* is:

$$\text{perplexity}(L, M) = 2^{E(L,M)}. \tag{3.6}$$

As for the cross entropy, a lower perplexity indicates a more accurate language model.

3.4 n-gram Models and Markov Models

The problems of predicting the next symbol x_n can be stated as attempting to estimate the probability function

$$p(x_n|x_1, \cdots, x_{n-1})$$

according to the previous context $x_1, \cdots, x_{n-1}$. The overall probability of a sequence X is given by:

$$\begin{aligned} P(X) &= p(x_1)p(x_2|x_1)p(x_3|x_1,x_2)\cdots p(x_n|x_1,\cdots,x_{n-1}) && (3.7)\\ &= \prod_{i=1}^{n} p(x_i|x_1,x_2,\cdots,x_{i-1}). && (3.8) \end{aligned}$$

Here, $x_1, \cdots, x_{i-1}$ is called the *history*. It is clear that a full-history language model could be computationally expensive. One way to solve this problem is by making the *Markov assumption* [Mar13], which means the history is only equivalent to the previous $n-1$ words. This history is called the *conditioning context*.

For example, if we only use the single previous symbol to condition the probability, we get a bigram model and the equation 3.7 is reduced to:

$$P(X) \approx \prod_{i=1}^{n} p(x_i|x_{i-1}). \tag{3.9}$$

Similarly, a trigram model use the previous two symbols to make a prediction, and the equation 3.7 is reformulated as:

$$P(X) \approx \prod_{i=1}^{n} p(x_i|x_{i-2}, x_{i-1}). \qquad (3.10)$$

Both the bigram model and the trigram model are a *Markov Model* since they are making the Markov assumption when predicting the next word. More generally, an n-gram model is also called an order $n-1$ Markov model. Here, we refer to vanilla Markov models as *Visible Markov Models*.

n-gram language models have been used in a wide range of NLP domains [Che96]. For example, Bahl, Jelinek and Mercer [BJM83] first applied bigram and trigram model in their speech recognition applications. Charniak use a n-gram model for part-of-speech tagging [CHJP93].

There is another type of Markov models named *Hidden Markov Model* (HMM) [RJ86], which differ from visible Markov models because they use a set of unknown parameters to make a Markov assumption. The challenge for a HMM is to determine the hidden parameters from the observable parameters. The extracted model parameters can then be used to perform further analysis, for example for pattern recognition applications [Rab89, Jel90].

3.5 The Sparse Data Problem

As we discussed in the previous section, an n-gram model must perform the prediction task $P(x_n|x_1 \cdots x_{n-1})$. Since:

$$P(x_n|x_1 \cdots x_{n-1}) = \frac{P(x_1 \cdots x_n)}{P(x_1 \cdots x_{n-1})}$$

estimating good conditional probability distribution can be reduced to having good solutions to simply estimating the unknown probability distribution of an n-gram [MS99].

Three approaches are normally used in processing n-gram language models: *estimating*, *smoothing* and *blending*.

Estimating seeks to solve the problem of what probability we should use to estimate the next word. An intuitive way to do this is based on the relative frequencies of context observed in the previous text. A *maximum likelihood estimate* (MLE) for this approach is:

$$P_{MLE}(x_1, \cdots, x_n) = \frac{C(x_1, \cdots, x_n)}{T} \tag{3.11}$$

and

$$P_{MLE}(x_n|x_1, \cdots, x_{n-1}) = \frac{C(x_1, \cdots, x_n)}{C(x_1, \cdots, x_{n-1})} \tag{3.12}$$

where $C(x_1, \cdots, x_n)$ is the frequency of sequence symbols $x_1, \cdots, x_n$ and T is the total number of the symbols in the training text. However, an immediate unpleasant fact for the MLE estimator is that if the conditioning context $C(x_1, \cdots, x_n)$ are extremely rare, or even never actually occurs in the previous text, zero probability is assigned by the estimator and the language model is therefore unable to recognize these novel symbols. The problem of how to handle rare events is referred to as the *sparse data problem*, or *zero frequency problem* if the estimated words are unseen. This problem is ubiquitous in statistical language modelling [Che96].

One approach to overcoming the sparse data problem is called *training* – using as much data as possible to train the language model. Training is a direct way to achieve an improvement in language model quality and it is fundamental to all statistical language model. However, training large amounts of data require enormous computing resources as well as storage resources [Tea98].

Another more clever way is to build more compact and better models for the probability estimation. This process to correct the probability estimate is called *smoothing*. A simply way to avoid the zero frequency problem is to employ *Laplace's law* [Lap14] by adding one to the counts, which is

$$P_{LAP}(x_n|x_1, \cdots, x_{n-1}) = \frac{C(x_1, \cdots, x_n) + 1}{C(x_1, \cdots, x_{n-1})}. \tag{3.13}$$

In 1953, Good introduced the *Good-Turing* estimate [Goo53], which was initially used for estimating the population frequencies of species and then became the basis of many smoothing techniques [CG91]. This method assumes that the determining frequency is binomial.

Let $C(x_1, \cdots, x_n) = r$ be the frequency and N_r be the frequencies of the frequency r, if $r > 0$

$$P_{GT}(x_1, \cdots, x_n) = \frac{r^*}{N} \tag{3.14}$$

where

$$r^* = \frac{(r+1)N_{r+1}}{N_r}. \tag{3.15}$$

If $r = 0$, the discount frequency r^* is therefore reduced to:

$$r^* = \frac{N_1}{N_0} \tag{3.16}$$

and

$$P_{GT}(x_1, \cdots, x_n) = \frac{N_1}{N_0 N} \tag{3.17}$$

From equation 3.16 and 3.17 we can see that the number of unseen words N_0 is actually unknown but a crude estimate can be determined from the size of the vocabulary V, where:

$$N_0 = V^n - \sum_{r>0} N_r.$$

Since $\sum_{r>0} N_r \ll V^n$, then $N_0 \approx V^n$. For example, the Chinese language has about 8,000 characters, so for a bigram model, $V = 8,000$ and $N_0 =$

$V^2 = 6400,0000$. Comparing equation 3.17 with 3.11, we see that the Good-Turing estimator assigns the same adjusted count r^* to all the novel words in n-gram.

By using smoothing, the estimator reserves some probability space for each word in each context. However, this means that initial prediction will have to rely to some extent on the zero frequency estimator rather than on the actual data sequences being coded. Of course we can make the context very short but that leads to poor language modelling in the long term because little of the structure of the sequence data will be available for making predictions [BCW90].

For n-gram models, a process to combine multiple probability estimates from various different context models into a single overall probability is called *blending*. One way to combine these predictions is to assign a weight to each model and calculate the weighted sum of the probabilities. This is usually named *linear interpolation* [Jel90]. For example, interpolating a trigram model is formulated as:

$$P_{li}(x_n|x_{n-2}, x_{n-1}) = \lambda_1 P_1(x_n) + \lambda_2 P_2(x_n|x_{n-1}) + \lambda_3 P_3(x_n|x_{n-2}, x_{n-1}) \tag{3.18}$$

where $0 \leqslant \lambda_i \leqslant 1$ and $\sum_i \lambda_i = 1$.

Of course the weights may be set by hand. However, there are several algorithms for calculating the weight. Baum introduced a *forword-backward* algorithm, which estimates the weights by finding those which maximize the probability for as training sequence [Bau72]. Katz's *Back-off* algorithm allows the estimator to "back off" from a upper n-gram model to a lower model according to the frequency counts of the context [Kat87].

3.6 Variable Order Markov Models

Variable Order Markov Models (VMM) are referred to those adaptive models that learn the probability distribution where the conditioning context varies and an adaptive response is applied to the available statistics in the previous data. Thus, unlike normal n-gram models, which can be regarded as a fixed-order Markov model, VMMs provide the means for capturing both large and small order Markov dependencies based on the observed data [BYY04].

There are many VMM prediction algorithms, of which the context tree weighting (CWT) algorithms [WST95], prediction by partial matching (PPM) algorithms [CW84] and the probabilistic suffix tree [RST96] algorithm are the most well known in the NLP community.

Context Tree Weighting (CTW) is a lossless compression algorithm proposed by Willems in 1995 [WST95]. This algorithm proposed a weighting predictor to assign a weight for every probability that gains from the path nodes of the context tree, and then combines these probabilities into an overall CTW probability. Further improvements have been made by Willems and Tjalkens [WST96, TVW97].

Prediction by partial matching (PPM) is an adaptive n-gram model used initially for text compression (see section 3.7). It was originally introduced by Cleary and Witten [CW84] in 1984 and then Moffat made a series of improvements and developed PPMC, which has become the benchmark version [Mof89]. PPM has been applied to many NLP problems such as cryptology, language identification, text correction [Tea98].

Probabilistic suffix trees (PST) [RST96] are well known in the machine learning community. This algorithm proposes a distribute learning algorithm for variable memory length Markov processes by using a Probabilistic Suffix Automata (PSA).

PPM generally uses for character-based text compression and CTW uses for binary-based compression. In this thesis, our method to compress the Chinese text is based on the Chinese character, we will use PPM method to apply Chinese text compression.

3.7 PPM: An Adaptive Variable Order Markov Model

In this section we will review one particular algorithm for adaptive statistical modelling: the PPM text compression model. We will apply this model to our Chinese language in chapter 4 and develop our new PPM variant in chapter 5.

3.7.1 PPM: 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 -1 model [CW84, BCW90]. 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* [CW84, Mof89, CT97, Shk02], which are mainly named according to the escape method they employ. In a recent paper, 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 [Shk02]. PPMII achieves a good compression rate as well as excellent compression speed.

3.7.2 PPM's Probability Estimation and Smoothing Mechanism

In Section 3.5, we have discussed different mechanisms to address the sparse data problem. PPM's solution is to assign an *escape* probability for a novel symbol, to get around the zero frequency problem. This is similar to Katz's back-off mechanism [Kat87], which assigns the same weight for all counts greater than 0. However, PPM models use *full exclusion* and *update exclusion* mechanisms to assign the weight for the different order models (we will discuss these two mechanisms in Section 3.7.3).

Normally different PPM variants are named according to their escape methods. Several methods have been used to calculate the symbol and escape probabilities for the context model.

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* $s_d = x_n, \cdots, x_{n-d+1}$. Let $c_d(\varphi)$ denotes 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.

Escape Method	**Escape Probability**	**Symbol Probability**	
Method A	$e = \frac{1}{T_d+1}$	$p(\varphi) = \frac{c(\varphi)}{T_d+1}$	[CW84]
Method B	$e = \frac{t_d}{T_d}$	$p(\varphi) = \frac{c(\varphi)-1}{T_d}$	[CW84]
Method C	$e = \frac{t_d}{T_d+t_d}$	$p(\varphi) = \frac{c(\varphi)}{T_d+t_d}$	[Mof90]
Method D	$e = \frac{t_d}{2n}$	$p(\varphi) = \frac{2c(\varphi)-1}{2T_d}$	[How93]

Tab. 3.1: Major PPM variants for Estimating The Probabilities

Table 3.1 shows the probability estimating mechanisms of several major PPM variants. For instance, if the context has occurred 6 times before, with symbol a following 3 times, symbol b following 2 times and symbol c following once, the escape probability e for PPMA, PPMB, PPMC and PPMD will be $\frac{1}{7}$, $\frac{3}{6}$, $\frac{3}{9}$, and $\frac{3}{12}$ respectively.

Method A and B were introduced by Cleary and Witten [CW84] in 1984, Method C was developed in 1990 by Moffat [Mof90] and Method D was proposed by Howard [How93] in 1993. Experiments show the compression result by using PPMD is usually slightly better than for PPMC, but both methods outperform PPMA and PPMB.

Witten also introduced several other escape methods in 1991 [WB91]. Method P estimates the escape probability as:

$$e = \frac{t_1}{T_d} - \frac{t_2}{T_d^2} - \frac{t_3}{T_d^3} - \cdots .$$

where t_i is the number of symbols observed exactly i times in the context s_d.

Method X notices that since T_d is normally very large, the approximate escape probability can be reduced to:

$$e \approx \frac{t_1}{T_d}.$$

It is clear when $t_1 = 0$ or $t_1 = T_d$, method P and X will break down because the novel event probability is 0 or 1 in these cases. To solve this problem, Witten combined this with the Method C to produce Method XC as:

$$e = \begin{cases} \frac{t_1}{T_d} & \text{when } 0 < t_1 < T_d \\ \frac{t_d}{T_d + t_d} & \text{otherwise.} \end{cases}$$

Moffat presented further ways to solve this problem [MSWB94] by adding one to the counts, named Method X1:

$$e = \frac{t_1 + 1}{n + t_1 + 1}$$

and

$$p(\varphi) = \frac{c(\varphi)}{T_d + t_1 + 1}.$$

Another variation of the PPM models is called PPM*, which make use of unbounded length contexts for PPM. It was originally proposed by Cleary & Teahan in 1997 [CT97]. The main idea of this algorithm is to use all substrings of the input string to generate the prediction, rather than only use several substrings that selected from the high order model to the low order model to predict the next symbols as normal PPM algorithm does. However, it requires substantially more resources, both in terms of memory requirements and execution speed – than standard PPM, and therefore we have concentrated on improving PPM rather than PPM* for this thesis.

3.7.3 PPM's Blending Mechanism

Bell, Cleary & Witten [BCW90] show that for PPM the blending probability of symbol φ is given by

$$p(\varphi) = \sum_{i=-1}^{m} w_i p_i(\varphi)$$

where w_i and p_i are the weight and probability assigned by order i context model. To avoid zero probability, a non-zero wights are assigned to the predictions from lower order contexts.

The PPM model uses *exclusion* as its blending strategy. It combines the predictions for all character contexts of which their length are less than or equal to a maximum order m and uses the escape mechanism to exclude lower order predictions from the final probability estimate. Normally a default order -1 model will be used to ensure all the symbols been assigned a finite probability.

Let e_i be the probability of an escape in order i, where $-1 \leqslant i < m$. The weight w_i assigned by PPM model is:

$$w_i = (1 - e_i) \prod_{k=i+1}^{m} e_k \qquad -1 \leqslant i < m$$

and

$$w_m = 1 - e_m$$

The weighted contribution of the model to the blended probability of symbol φ is thus:

$$w_i p_i(\varphi) = (1 - e_i) p_i(\varphi) \prod_{k=i+1}^{m} e_k.$$

As it was stated by Bell, Cleary & Witten [BCW90]: "the advantage of expressing things in terms of escape probabilities is that they tend to be more easily visualized and understood than the weights themselves, which can become small very rapidly. We also see the the escape mechanism is much more practical to implement than weighted blending."

Furthermore, Bell, Cleary & Witten [BCW90] introduce a *full exclusion* scheme to improve the PPM's blending algorithm. This scheme indicates that "symbols predicted by higher order context need not be predicted by lower order ones since they will have already been encoded. These symbols waste codespace since they can be excluded altogether from the prediction with no effect on the outcome [Tea98]." There is some extra computational overhead by using full exclusion because each symbol has to be checked for exclusion.

Another improvement is called *update exclusion*, which is introduced by Moffat in his PPMC algorithm [Mof90]. This algorithm states that the count will only be updated in context levels at or above the context in which it was successfully predicted. In another words, one count is updated only if it is not predicted by any higher order context. On the average, update exclusion improves compression by 2%, and it also speeds up the program execution time because of removing the need to update the counts in lower context levels [BCW90].

In order to illustrate the traditional PPM model, we use PPMC as an example. We use same definition as shown in the Section 3.7.2.

Since the probability of a symbol φ occurring in the context s_d is:

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

and 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 explain the operation of PPM, Table 3.2 shows the state of the 17 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 [WNC87, MNW98] 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 φ, the associated probability $p_d(\varphi)$ is used to encode it. For example, if "是" follows 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" $\rightarrow$"是" has been made in the order 2 model.

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	$\frac{5}{17}$	$\rightarrow A$	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	$\frac{1}{2}$	问 →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	$\frac{1}{2}$									

Tab. 3.2: PPMC model after processing the string "中国的中是中华中间的中".

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, and the $d = 1$ 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” because the character “是” can be excluded, which saves coding space compared to 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, $|A| \approx 8000$.

3.8 Text Compression

3.8.1 Text Compression Theory

For text compression, the main goal is to reduce size of the encoded data without losing any information of the original text, which means the process is reversible, or lossless.

In 1981, Rissanen and Langdon provided the insight to characterize the process of data compression as being separated into two steps: *modelling* to estimate a probability of each character, and an *encoding* that uses a coding algorithm to produce a compressed representation of data in respect to the probability of each of the characters [RL81].

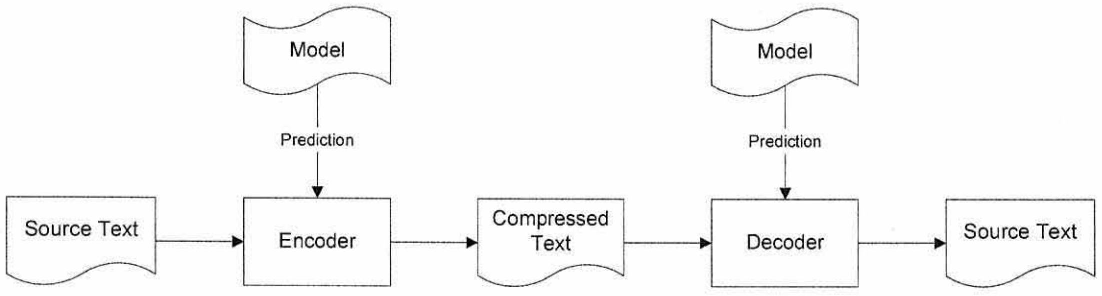

Fig. 3.1: Using a model for compression.

Figure 3.1 [BCW90] shows how a model is used in practice. The encoder and decoder share the same model. The encoder use the probability which is estimated by the model to encode the source into a compressed representation of the data. The decoder uses the same model to estimate the probability to decode the compressed data into the original source file.

Generally speaking, the models for the text compression can be divided into 3 types: *static, semi-adaptive and adaptive*, according to the method of estimating the probability.

The *static model* means a fixed model regardless of the text to be encoded. The encoder and decoder run the same coding algorithm according to the identical codebooks. For instance, in Shannon-Fano coding [Sha48], the probability of each character is fixed based on the probability table that is pre-defined before encoding starts.

Unlike the static model, the *semi-adaptive model* reads the message first and then makes up a codebook according to some criteria such as the frequency of the symbols in the coding message, the encoder then transmits the codebook with the coded message. A disadvantage of this kind of model is that sending a codebook for each message can be time consuming.

For the *adaptive model*, the codebook of both encoder and decoder are updated according to the input stream of the character by using a dynamic codebook.

Predominately, there are two different approaches to achieve an adaptive model: the *dictionary* approach and the *statistical* approach.

The most well known adaptive dictionary model is the Ziv-Lempel compression model, which was introduced by Ziv and Lempel [ZL77] in 1977. The main idea of this scheme is to replace phrases with a pointer if they have occurred earlier in the text. It uses a sliding window to define the length of the *longest match* and a *triple array* to store the compressed data. A series of improvements have been made and different Ziv-Lempel coding variants were developed such as LZ78 [ZL78], LZW [Wel84] etc.

The second approach uses statistical methods to estimate the probability for the coding algorithm. For instance, *Finite-context* models use the preceding few characters to predict the next one [BCW90]. As we have discussed in Section 3.7, Cleary and Witten's PPM [CW84], which use *Arithmetic coding* as it's coding algorithm, is one of the best performed lossless finite-context statistical data compression models [Tea98]. The *Dynamic Markov Model* (DMC), originally described by Horspool [HC86], is a statistical finite-state models [BCW90] that is mainly used to process the binary input. Bell & Moffat shown DMC is actually a finite-context model [BM89].

3.8.2 Text Compression for Chinese

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 [VZ98]. 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 [Lua95] introduced a dictionary-based approach based on minimization of sentence entropy to compress Chinese text. This algorithm first built a frequency dictionary from the original text for all the sub-strings that have 1 to 9 characters, and then used this dictionary to compute a sentence entropy. The sub-string that contributes minimum entropy to the sentence is retained as a code unit. The compression rate of 6.91 *bits per Chinese character* (bpCc) on 14-bit Chinese characters has been achieved for this algorithm.

Gu [Gu95, Gu05] used order 2 Markov model for Big5 encoded Chinese text, and achieved the compression rate of 5.424 bpCc. This approach is similar to PPM's scheme.

Vines and Zobel [VZ98] modified the PPM model for Chinese characters by changing the unit of encoding from 8 bits to 16 bits, and achieved 6.2 bpCc.

Kwok-Shing Cheng and Gilbert H. Young [CYW99] introduce a word based dictionary coding algorithm that uses 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, Ghim-Hwee Ong and Jun-Ping Ng [ON05] announced the compression rate of 7.925 bpCc for Dynamic Markov Compression. A tree structure to use with Dynamic Markov Compression (DMC) for the compression of Chinese text files.

A problem with all these results is that different text corpora were used to perform the testing, as well as different models (e.g. orders, and whether based on characters or words) so they cannot be directly compared. Purpose of the corpus described in Section 2.4.4 is to provide standardization so that further results can be directly compared.

3.9 Conclusion and Discussion

In this Chapter, we review several important theories in statistical NLP. These theories are fundamental for our thesis.

We have explained that a language model to predict probability distribution is the key for statistical NLP. Different NLP applications such as text compression, word segmentation, text categorization have been developed based on different statistical language models.

Text compression is one of the most important NLP applications because the compression rate directly reflects the quality of a language model. Better compression indicates a possible better performance in other NLP applications. For example, text mining is one application which can make use of text compression models to achieve better NLP performance by using the Viterbi algorithm and variations to annotate the text [WBMT99]. On the other hand, it has been found that understanding gained from applying text compression models to NLP applications can lead to further improvements in text compression itself [TH01a].

4 Adapting PPM for Chinese Text Compression

Contents

4.1 Introduction

Although research for many NLP applications have been initially focused on specific languages, mainly English, their language models can still be adapted for a wide variety of other languages.

In this chapter we adapt the traditional PPM model for Chinese text compression. We first examine the characteristics of Chinese language text in section 4.2, then discuss our pre-processing techniques for Chinese characters, which leads to improved overall compression in section 4.3. In section 4.4 we analyze several problems of the traditional PPM model, and presents some important modifications to the PPM model according to the Chinese language's characteristics. Section 4.5 shows the significant experimental results we obtained with our corpus. We summarize our study and discuss important research issues in section 4.6.

4.2 Why Chinese Text is So Different

4.2.1 Large alphabet size

As was pointed out in Section 2.5.1, the alphabet size of Chinese language is large. Of all the published dictionaries in China, the "Chinese Character dictionary" (中华字海) [Len94] lists about 85,000 characters and symbols, and is the biggest dictionary for Chinese characters. Although 90% of these characters are rarely used today, there still are at least 8,000 Chinese characters being used in Chinese language. This alphabet size is still an order of magnitude larger when compared to that of English.

4.2.2 Not naturally segmented

Another problem that was pointed out in Chapter 3 is that Chinese Text is not naturally segmented by space or any other symbol. English words in English text, however, are separated by spaces. Therefore, it is not clear what is the most appropriate symbol to use for encoding in Chinese.

In English experiments, it has been found that word-based symbols are significantly better than character-based ones [Tea98]. Experiments with parts of speech lead to a slight improvement [Tea98] but these are also based on the word being a fundamental unit of encoding. For Chinese, however, word segmentation is problematical. Indeed, some studies suggest that the notion of a word in Chinese is an alien one [TWMW00]. The next section highlights why the issue of what symbol unit to use (be it words, characters, or whatever) is the key to gaining better compression not just for the Chinese language and other non-naturally segmented language (such as Japanese, Korea and Thai) but also for all languages.

4.2.3 Spaces do not help compression

Spaces can help achieve good compression for English [Tea98]. But for Chinese, the opposite affect is observed. Adding spaces to Chinese text reduces the compression result.

To illustrate, we use Guo Jin's Mandarin Chinese *PH corpus* (See section 2.4.2). The segmented PH corpus has spaces between each Chinese word, which have been manually inserted as judged by a native speaker. The unsegmented PH corpus, in contrast, does not contain any spaces. In comparison, we use the Brown Corpus [FK82], which is a million-word balanced corpus of American English text published in 1961. It comprises 500 text samples of approximately 2,000 words distributed over 15 categories.

Table 4.1 contains the compression rates for the Brown Corpus and PH Corpus when they are with space and without space. We use different order PPMD [Tea98] models to compress the files (see section 3.7.2). For the English corpus, we use bit per character (bpc) as its compression rate:

$$bpc = \frac{\text{Bytes in Output File} \times 8}{\text{Bytes in Original File}}$$

However, for Chinese corpus, we use bits per Chinese character (bpCc).

$$bpCc = \frac{\text{Bytes in Output File} \times 8}{\text{Number of Chinese Characters in Original File}}.$$

PPMD model	Brown Corpus	Brown Corpus	PH Corpus	PH Corpus
	With spaces (bpc)	No spaces (bpc)	With spaces (bpCc)	No spaces (bpCc)
Order 0	4.544	4.651	14.422	12.284
Order 1	3.669	3.914	10.906	9.966
Order 2	3.019	3.445	7.872	7.718
Order 3	2.508	3.018	6.650	6.224
Order 4	2.232	2.742	6.050	5.880
Order 5	2.157	2.615	5.856	5.830
Average	3.022	3.398	8.626	7.984

Tab. 4.1: Compression rate comparison for spaced and non- spaced English and Chinese Corpora under different PPMD orders.

It is very clear, from Table 4.1, that for English and for all models of varying order, although there are substantially more characters in the original text, it achieves a better compression rate than the text in which all the spaces have been deleted. On average, the compression rate is 3.022 bpc, which is 11% better than the no space text.

However, for Chinese, the effect is the opposite – the average compression rate of the text with spaces is 8.626 bpCc, which is 8% worse than the no space text. This indicates a number of possibilities: that Chinese language is fundamentally different from English language; that the manually assigned segmentation is inappropriate or incorrect; and/or a word-based alphabet is not the way to model Chinese.

4.3 Pre-processing for Chinese Text

Pre-processing techniques, of which the *Burrows-Wheeler Compression Algorithm* [BW94] is one example, change the text into a different format with the goal being to improve the subsequent compression stage. These preprocessing techniques need to be reversible so that the original file can be recovered after the decoding stage. Abel & Teahan [AT06] describe some pre-processing techniques for text. This section proposes a simple pre-processing technique for Chinese that leads to significant improvement in subsequent compression.

As we introduced in section 2.3, there are two main methods for encoding Chinese – GB code and Big5 code. The following table shows the code range of GB and Big5 code.

	First byte	**Second byte**
GB code	0xA1–0xFE	0xA1–0xFE
Big5 code	0x81-0xFE	0x40-0x7E, 0xA1-0xFE

Tab. 4.2: Code range of GB and Big5 code

Unlike GB code, which uses 0xA1-0xFE for both of the two bytes of a Chinese character, Big5 code uses 0x81-0xFE as the first byte, and 0x40-0x7E, 0xA1-0xFE as the second byte.

Our pre-processing technique is as follows. Rather than encode each 16 bit Chinese character as 2 separate bytes, we convert each 2 bytes into one integer so that it corresponds to a single Chinese character. In theory, for the GB code, the maximum integer will be the square of 256, which means the alphabet size is at least 65536 if there is no pre-processing. This is because the maximum coding range is 256 for both the high and low byte of a GB character.

However, it is possible to reduce the alphabet size substantially. First, since both of the 2 bytes start from 0xA1, we can just subtract 0xA0 (value 160) when we encode a character, and add 0xA0 when we decode it. This change re-scales the coding range from 0x01 to 0x5E.

Second, a constant multiplier can be used to concatenate the high and low byte. We use the following pre-processing method:

$$R = (H - 160) * K + (L - 160) + 256 \tag{4.1}$$

where R is the pre-processing result used when encoding, H is the high byte and L is the low byte of the Chinese character. K is the constant defined as follows. To prevent errors when we decode R, K should satisfy the following condition: $\max(L - 160) < K$. Because $\max(L) = 254$, so $\min(K) = 95$.

Third, in practice, the high byte of the GB code is from 0xA1 to 0xF7 in the GB2312-80 character set, which means it can be re-scaled from 0x01 to 0x57, therefore reducing the overall alphabet size.

And last, 256 characters should be set aside to encode special characters that do not meet the error constraint mentioned above. For example, some GB encoded text may include BIG5 coded characters within it. But we want to restrict the size of our alphabet, so we just skip the pre-processing work when this occurs, and simply encode a character as two separate ASCII bytes. Because we include all 256 ASCII characters, when decoding, if a number is above 256, then this indicates we have used our pre-processing technique. Conversely, if it is smaller than 256, we just output the actual ASCII character.

Simply by using the pre-processing described above on the text before coding, we have found that we can significantly improve text compression for Chinese text. Also, we can reduce the alphabet size to 8615 for the GB code.

For Big5 code, the pre-processing algorithm is:

$$R = (H - 160) * K + L' + 256 \tag{4.2}$$

where L' = L - 98 if the low byte range is from 0xA1 to 0xFE, or L' = L -

98 + 34 if the low byte range is from 0x40 to 0x7E. K = (0xFE - 0x40) - (0xA1 - 0x7E) = 155. The alphabet size for Big5 code is therefore reduced to 14,052.

	No pre-processing	**With our pre-processing method**
	(bpCc)	(bpCc)
Chinese article1	9.100	8.496
Chinese book1	8.072	7.656
Chinese book2	7.714	7.474
Chinese Bible	5.552	5.530

Tab. 4.3: The effect of preprocessing the text prior to encoding when using an order 2 PPMD model to compress Chinese text.

Table 4.3 shows some compression results for Chinese text by using and not using the pre-processing techniques. We use an order 2 PPMD model and select *Chinese article, book* and *Bible* from the SBMCC (See Section 2.4.4) as sample test files.

The first method in column 2 of Table 4.3 does not use any pre-processing; it just encodes and decodes each Chinese character as two separate ASCII bytes. The second method in column 3 uses the pre-processing technique. The result shows that pre-processing improves the compression rate significantly, achieving about 8% better compression than when no pre-processing was done.

4.4 Adapting PPM for Chinese Text Compression

In order to adapt PPM to Chinese text, we have experimented with a number of configurations and have found that the following method works well.

First, although each Chinese character is encoded by 16 bits, the byte-oriented PPM model is not predicting single Chinese characters, but rather halves of Chinese characters. In contrast, we encode whole Chinese characters (i.e 16 bits and a much larger alphabet) as a result of our pre-processing technique. The merit of this change is very clear. 16 bits hold complete Chinese characters, which capture the structure of the language precisely [VZ98].

Second, we sort all the characters in the PPM context models by frequency order. Because of the increased alphabet size, the amount of characters predicted by a context can be very large. For example, PPM order 0 contexts save all the characters that have occurred prior to the point of encoding. For English, there will be roughly 98 printable characters defined by the ASCII character set. But for Chinese, it will be thousands of Chinese characters. Maintaining all these characters in the context in sorted frequency order can significantly reduce the program execution time.

The following table shows the execution time differences between the sorted and unsorted schemes using the PPM model. The configuration of our test machine is Intel Pentium III 1GHz, with 640M internal memory.

	File size (bytes)	**Using Sorted Context** (seconds)	**Using Unsorted Context** (seconds)
Chinese news1	15,372	0.20	1.10
Chinese article1	19,245	0.25	1.50
Chinese book1	53,706	0.61	3.48
Chinese book2	436,656	5.22	29.46
Chinese Bible	2,032,066	17.19	121.23

Tab. 4.4: Execution time for PPM model by using *Sort* and *Unsorted* Schemes under PPMD.

As Table 4.4 shows, by sorting the characters in each context by frequency order, it runs about 6 times faster than the unsorted scheme.

The third innovation is that we do not use exclusions. Exclusions remove all characters for prediction from appearing in a lower order as they already have appeared in a higher order. Exclusions help the compression rate but significantly increases the time consumption for large alphabets. For languages like Chinese, performing no exclusions results in a slightly worse compression result (about 1% worse than with exclusions), but the bonus is that it runs much faster. We will show our experiment result in the next section.

4.5 Experimental Results

We compare our experimental model with other standard compression schemes in this section using files in the BMCC (see 2.4.4).

gzip and *bzip2* are two popular compressors under Unix/Linux environment. *WinRAR* and *ABC2.4* are two compressors under Microsoft Windows environment. *ABC2.4* use Burrows-Wheeler Compression Algorithm [BW94]. PPMC [Mof90], PPMD [How93], PPMII [Shk02] are several variations of the PPM model. *PPM-Ch* for "PPM Chinese model" stands for our experimental model.

Table 4.5 shows the execution time for PPMD and PPM-Ch under order an 2 model. We select *Chinese news, article, books, Bible and PH Corpus* from BMCC (see section 2.4.4) as test files. The configuration of our test machine is Intel Pentium M 1.7 GHz, with 512MB internal memory.

The execution time of PPM-Ch is about 2-4 times faster than PPMD for small files like *Chinese news1, article1, book1* and *book2.* For large files, PPM-Ch is still faster than PPMD, even though PPM-Ch is encoding using a larger alphabet size. The following table is the full test results for the

	File size	**PPMD**	**PPM-Ch**
		Order 2	Order 2
	(bytes)	(seconds)	(seconds)
Chinese news1	15,371	1.10	0.27
Chinese article1	19,245	1.50	0.33
Chinese book1	53,706	3.48	0.82
Chinese book2	436,656	29.46	14.76
Chinese Bible	2,032,066	121.23	22.09
PH Corpus	7,506,581	625.52	546.96

Tab. 4.5: The execution time for PPMD Order 2 and PPM-Ch Order 2.

Bangor Mandarin Chinese Corpus under different compressors. Bold font is used in each line to indicate the best compression result achieved.

PPM-Ch	**Chinese news1**	**Chinese Bible**
	(seconds)	(seconds)
Order 0	0.21	18.06
Order 1	0.27	18.09
Order 2	0.27	22.09
Order 3	0.22	24.04
Order 4	0.30	25.54
Order 5	0.25	26.91

Tab. 4.6: The execution times for PPM-Ch using Order 0 to Order 5.

Table 4.6 illustrates the execution times when using PPM-ch with order 0 to order 5 model. It is clear that the execution time is comparable no matter which order is selected.

Table 4.7 shows, for small files, our variant PPM-Ch achieves the best

file	**size** (bytes)	*gzip* (bpCc)	*bzip2* (bpCc)	*WinRAR* (bpCc)	*ABC* (bpCc)	*PPMC* (bpCc)	*PPMD* (bpCc)	*PPMII* (bpCc)	*PPMD-Ch* (bpCc)
Chinese news1	15,371	9.144	8.992	9.098	8.856	8.426	8.434	8.238	**7.91**
Chinese news2	26,195	9.602	9.384	9.546	9.126	8.936	8.956	8.648	**8.336**
Chinese article1	19,245	9.766	9.574	9.726	9.314	9.058	9.1	8.68	**8.496**
Chinese article2	38,724	9.41	8.802	9.32	8.516	8.458	8.456	8.166	**8.042**
Chinese article3	80,760	8.472	7.772	8.316	7.544	7.432	7.412	7.142	**7.028**
Chinese book1	53,706	8.932	8.406	8.79	8.166	8.064	8.072	7.76	**7.656**
Chinese book2	436,656	9.34	8.026	8.77	7.66	7.744	7.714	**7.378**	7.474
LCMC-A	271,053	9.526	8.706	9.036	8.446	8.394	8.386	8.03	**7.926**
LCMC-B	170,469	8.826	8.17	8.442	7.93	7.814	7.818	7.512	**7.42**
LCMC-C	112,681	8.204	7.53	7.854	7.38	7.18	7.166	6.91	**6.808**
LCMC-D	103,031	8.956	8.476	8.672	8.226	8.112	8.11	7.8	**7.624**
LCMC-E	218,473	9.002	8.564	8.646	8.286	8.246	8.248	7.892	**7.778**
LCMC-F	266,266	9.31	8.632	8.922	8.372	8.332	8.326	7.966	**7.878**
LCMC-G	452,511	9.626	8.652	9.048	8.316	8.322	8.312	7.962	**7.896**
LCMC-H	208,035	7.016	6.238	6.458	6.09	5.996	5.976	**5.75**	5.756
LCMC-J	509,095	8.22	7.352	7.694	7.104	7.058	7.034	**6.754**	6.802
LCMC-K	159,118	9.656	9.004	9.332	8.632	8.692	8.688	8.296	**8.094**
LCMC-L	137,743	9.53	8.916	9.236	8.626	8.62	8.614	8.238	**8.006**
LCMC-M	35,159	9.472	9.15	9.326	8.946	8.756	8.786	8.464	**8.086**
LCMC-N	155,897	9.482	8.87	9.138	8.53	8.56	8.564	8.176	**8.016**
LCMC-P	273,109	9.46	6.256	5.43	6.038	6.394	6.424	5.892	**5.856**
LCMC-R	49,289	9.254	8.822	9.126	8.546	8.484	8.49	8.162	**7.848**
Average		9.1	8.378	8.634	8.12	8.05	8.05	7.718	**7.578**

Tab. 4.7: Chinese text compression for Small Bangor Mandarin Chinese Corpus using different compressors

compression rate in most cases. It is about 3% – 6% better than the second place compressor on average. PPMII achieves the best compression on a few of the files, but in these cases it is just sightly better than PPM-Ch.

file	**size** (bytes)	*gzip* (bpCc)	*bzip2* (bpCc)	*WinRAR* (bpCc)	*ABC* (bpCc)	*PPMC* (bpCc)	*PPMD* (bpCc)	*PPMII* (bpCc)	*PPMD-Ch* (bpCc)
TREC1	818,786	8.22	6.382	6.718	6.248	5.89	5.882	5.856	**5.72**
TREC2	1,348,941	8.374	6.818	7.044	6.292	6.196	6.168	**5.966**	6.026
TREC3	1,171,931	8.286	6.808	6.972	6.354	6.238	6.226	**5.994**	6.056
TREC4	1,034,517	8.35	6.788	7.06	6.394	6.268	6.246	**6.034**	6.076
TREC5	1,478,840	8.204	6.67	6.826	6.174	6.082	6.064	**5.854**	5.924
Chinese Bible	2,032,066	7.364	6.092	6.288	5.522	5.578	5.552	**5.38**	5.53
PH1	3,259,284	8.564	7.082	6.802	6.176	6.168	6.136	6.218	**6.034**
PH2	3,466,810	8.692	7.192	6.924	6.272	6.268	6.236	6.32	**6.128**
PH Corpus	7,506,581	8.604	7.102	6.65	6.12	5.918	**5.88**	6.258	6.304
Average		8.296	6.77	6.81	6.172	6.068	6.044	**5.944**	5.978

Tab. 4.8: Chinese text compression for Large Bangor Mandarin Chinese Corpus using different compressors

Table 4.8 shows, for the large files above 800KB and under 3 MB, PPMII performs best, but still is just 1%-1.5% better than PPM-Ch. For the large files above 3MB, the PPM-Ch obtains the best compression.

Although we do not add exclusions into the PPM-Ch model, PPM-Ch still achieves the best compression result in most of the test files. The average compression rate for PPM-Ch is 6.768 bpCc, which is 5% better than PPMD. The implementation is not yet optimized so its execution speed is not as good as PPMII (also the latter has the advantage that it performs escapes much less regularly).

4.6 Conclusions and Discussion

We have adapted the PPM model especially for Chinese text and achieve good compression results. We highlighted the importance of preprocessing work for Chinese, as unlike naturally segmented language such as English, it is not clear what are the most appropriate symbols to use for encoding. Our experiments with the corpus show that the pre-processing work can improve the compression rate significantly.

We made several changes in the PPM model to adapt specifically to the Chinese language. Changing the symbol encoding unit to 16 bits captures the structure of the language precisely. Sorting all the characters in the context by frequency order improves the program speed significantly and using no exclusions also leads to faster execution speed.

The drawback of this new PPM-Ch model is still the alphabet size of the Chinese language, which results in reduced program speed as well as compression rate. This new PPM-Ch model should also achieve similar improvements in other large alphabet size languages such as Japanese, Korean and Thai.

Further work in this area may include a further optimization of the coding scheme. It is not clear how the improvements we have detailed here would affect other PPM-based implementations. The techniques we present here were arrived at after carefully crafting variations of PPMD especially for the problems associated with Chinese (specifically, the large alphabet size).

5

PPMO:A New PPM Variant for Chinese Text Compression

Contents

5.1 Introduction

In the previous chapter, we adapted the PPM model for Chinese text compression by using various techniques such as character preprocessing and frequency sorting. However, the large alphabet is still one of the main issues for Chinese text compression.

In this chapter, we discuss the further problems of the PPM Models in section 5.2, then our new adaptive PPM variant, dubbed PPMO (for PPM with *orders stream*), is discussed in Section 5.3. The inspiration for this new variant came from the understanding gained from the development of PPM-ch and the problems that were identified in Section 4.2. Section 5.4 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 5.5.

5.2 Further PPM problems for Chinese text

The large alphabet size is one of the most important characteristics for Chinese and this leads to many obstacles for processing natural Chinese language.

One main problem of using the traditional PPM algorithm for Chinese text identified when developing PPM-ch 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.

Table 5.1 shows the proportion that escapes contribute to the overall codelength when using the PPM-ch model with maximum order two for LCMC corpus. On average, about 13.93% of the overall codelength has been taken up with encoding the escape character.

Name	**Size** (bytes)	**Compressed File Size** (bytes)	**Escape Size** (bytes)	**Escape Proportion** %
LCMC-A	271,035	134843	18047	13.38
LCMC-B	170,469	79285	11612	14.65
LCMC-C	112,681	47954	7039	14.68
LCMC-D	103,031	49342	7036	14.26
LCMC-E	218,473	106670	14860	13.93
LCMC-F	266,226	132003	17985	13.62
LCMC-G	452,511	225212	29958	13.30
LCMC-H	208,035	74846	11180	14.94
LCMC-J	509,095	216419	29864	13.80
LCMC-K	159,118	82028	11208	13.66
LCMC-L	137,743	69891	9527	13.63
LCMC-M	35,159	18085	2648	14.64
LCMC-N	155,897	79174	10921	13.79
LCMC-P	273,109	99944	12448	12.45
LCMC-R	49,289	24566	3494	14.22
Average	208125	96018	13189	13.93

Tab. 5.1: Escape Proportion for PPM-ch model, maximum order = 2.

Also, 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 [Tea98] (see Table 4.7 and 4.8 in Chapter 4).

5.3 PPMO: Avoiding escapes by multi-streaming PPM

In order to overcome the disadvantages of PPM-ch and traditional PPM algorithms, we separate the coding process into two parts, which we name the *orders stream* and *symbols stream*. The orders stream outputs a coding order for each coding symbol, whereas the symbols stream outputs the coding symbol itself based on the context for the particular order. To illustrate this algorithm, we use the same definitions which were mentioned in the Chapter 3 Section 3.7.2.

- If a symbol φ has been found in the context s_d, the current maximum order d from the orders stream is encoded first (see below), followed by the symbol itself from the symbols 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 count to each symbol in the alphabet. Once the model which 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.

To illustrate the operation of PPMO, We use the same example that was used for table 3.2.

Table 5.2 illustrates the PPMO model after processing the string "中国的中是中华中间的中" and Table 5.3 shows the order and symbol probabilities used when processing the same string.

Suppose "是" followed the input string, the encoding process outputs 2 in the orders stream because "是" would be encoded in the order 2 model, then outputs a probability of 1 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

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 $\rightarrow$的	1	1	中 $\rightarrow$国	1	$\frac{1}{4}$	$\rightarrow$ 中	5	$\frac{5}{11}$	$\rightarrow A$	1	$\frac{1}{\vert A\vert}$
			$\rightarrow$是	1	$\frac{1}{4}$	$\rightarrow$ 的	2	$\frac{2}{11}$			
国的 $\rightarrow$中	1	1	$\rightarrow$华	1	$\frac{1}{4}$	$\rightarrow$ 国	1	$\frac{1}{11}$			
			$\rightarrow$间	1	$\frac{1}{4}$	$\rightarrow$ 是	1	$\frac{1}{11}$			
的中 $\rightarrow$是	1	1				$\rightarrow$ 华	1	$\frac{1}{11}$			
			国 $\rightarrow$的	1	1	$\rightarrow$ 间	1	$\frac{1}{11}$			
中是 $\rightarrow$中	1	1									
			的 $\rightarrow$中	1	1						
是中 $\rightarrow$华	1	1									
			是 $\rightarrow$中	1	1						
中华 $\rightarrow$中	1	1									
			华 $\rightarrow$中	1	1						
华中 $\rightarrow$间	1	1									
			间 $\rightarrow$的	1	1						
中间 $\rightarrow$的	1	1									
间的 $\rightarrow$中	1	1									

Tab. 5.2: PPMO model for symbols stream after 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	$\frac{1}{\vert A\vert}$	$\frac{1}{\vert A\vert}$	$\frac{1}{\vert A\vert}$	$\frac{1}{3}$	$\frac{1}{\vert A\vert}$	$\frac{2}{5}$	$\frac{1}{\vert A\vert}$	$\frac{3}{7}$	$\frac{1}{\vert A\vert}$	$\frac{1}{9}$	1

Tab. 5.3: Order & Symbol probabilities for PPMO model when processing the string “中国的中是中华中间的中”.

downgrades from order 2 to the order 1 model and the $d = 1$ context "中" is used. Then a 1 is output in the orders stream because the desired prediction "中" →"国" is selected with probability $\frac{1}{4}$.

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 orders stream and all characters would be encoded as a probability of $\frac{1}{A}$.

As the Table 5.2 shows, all the coding probabilities in the order 2 context model and mostly in the order 1 model are one, $P_s(\varphi) = 1$. This compares with our previous probability distributions for PPMC model in Table 3.2 (see Section 3.7.3), where all probabilities must be smaller than one ($p_d(\varphi) < 1$) by the nature of the traditional PPM encoding scheme.

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

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

Figure 5.1 shows a sample file of the orders stream for encoding the Chinese Bible in SBMCC (see section 2.4.4).

```
1,2,0,1,0,1,2,2,0,0,1,0,1,2,1,0,0,0,0,0,0,0,1,2,0,0,0,1,1,0,
1,2,2,-1,0,1,0,0,0,1,2,2,0,1,-1,-1,0,1,0,0,1,0,-1,0,-1,0,1,
2,0,1,2,2,0,1,2,1,0,-1,0,-1,0,1,0,0,-1,0,0,1,0,0,1,0,1,2,2,
```

Fig. 5.1: Sample output of the orders stream for encoding the Chinese Bible.

It is clear that the orders stream is highly predictable because of its small alphabet size and repetitive nature. For example, if the maximum coding order of the symbols stream is 2 for a model, there are only 4 possibilities in the orders stream, which are $\{-1, 0, 1, 2\}$. We can regard the order d as separate entities to be encoded and we can 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 orders stream and $d^* \leq D$ denotes the current coding order. s_i^* denotes the current orders 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 orders stream, to find out which order the symbol has been encoded with, and then decode the symbols stream based on the order of the model.

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 compared to 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.

5.4 Experimental Results

For comparison, we use the same compression tools as in section 3.5. We use an order 2 model as the maximum encoding order for both the symbols stream and orders stream as this produces the best result. We also use PPM-ch Order 2 model that was discussed in the previous chapter.

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 [BCK04]. The configuration of our test machine is the same one as we described in Chapter 4 Section 4.5.

Table 5.4 shows the results for the Small Bangor Mandarin Chinese Corpus 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% – 4% better than the second place compressor PPM-ch and about 5–6% better than the third place compressor PPMII.

file	**size**	*gzip*	*bzip2*	*WinRAR*	*ABC*	*PPMC*	*PPMD*	*PPMII*	*PPM-Ch*	*PPMO* 2+2
	(bytes)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(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**

Tab. 5.4: Chinese text compression for Small Bangor Mandarin Chinese Corpus using different compressors.

file	**size**	*gzip*	*bzip2*	*WinRAR*	*ABC*	*PPMC*	*PPMD*	*PPMII*	*PPM-Ch*	*PPMO* 2+2
	(bytes)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(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**

Tab. 5.5: Chinese text compression for Large Bangor Mandarin Chinese Corpus using different compressors.

Table 5.5 shows the compression results for the Big Bangor Mandarin Chinese Corpus. 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 orders stream, we achieve further improvements with the average compression rate reducing to 5.860 bpCc.

This is surprising because the standard PPM algorithm has been heavily optimized following decades of research, whereas PPMO has yet to be optimized and is based on the simple idea of separating out the encoding of the orders stream, and using pure counts to encode the symbols

File	**Size**	*PPMO*	*PPMO*	*PPMO*	*PPMO*	*PPMO*
		2+1	2+2	2+3	2+4	2+5
	(bytes)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(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

Tab. 5.6: Chinese text compression for selected files from the Bangor Mandarin Chinese Corpus using different orders for the orders stream.

stream.

Table 5.6 shows the compression results by using different orders for the orders stream in the PPMO model. For example, PPMO 2+4 shows the compression result by using an order 2 model for the symbols stream and order 4 model for the orders stream. We select one small file and one big file to illustrate; results are similar for other files in the corpus. It is clear that the compression results are comparable across orders 1–5 for the orders stream.

File	**Size**	PPMO	PPMO	PPMD	PPMII
		with exclusion	no exclusion		
	(bytes)	(second)	(second)	(second)	(second)
LCMC-A	271,053	27.5	4.1	17.5	0.22
Chinese Bible	2,032,066	185	25.9	70	0.73

Tab. 5.7: Execution times for PPMD, PPMII, and PPMO encode by using exclusion and no exclusion.

Table 5.7 shows the encode and decode execution times for PPMD, PP-MII and PPMO by using exclusions and no exclusions. It is clear that PPMII is much faster than any other PPM-based compressor. However, this well-crafted implementation has been heavily optimized for speed, whereas the focus for our implementation has been on compression performance. Nevertheless, the coding speed of PPMO with no exclusions is approximately 7 times faster than that of PPMO with exclusions, and approximately 3-4 times faster than the traditional PPMD model. Moreover, experimental results of PPMO with no exclusions indicate a 2%–3% drop in compression rate, but this is still much better than most of the other compressors with better execution speed as well.

Figure 5.2 shows how the size of the compressed text increases as the input text grows for PPMO, PPMII and PPMD on the Chinese Bible. At 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 (Table 5.4, 5.5) unlike for the Bible text file which we have chosen for illustration purposes here.

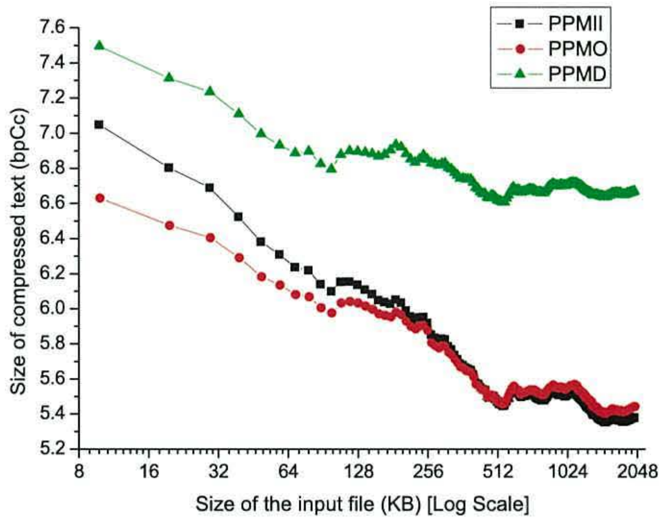

Fig. 5.2: Size of the output compressed file for PPMO, PPMII and PPMD as the text grows.

Table 5.8 shows the separate percentage costs of encoding the symbol and orders stream for the Bible file. It can be seen that the cost of encoding the symbols stream is approximately $\frac{4}{5}$ of the total cost, and the orders stream $\frac{1}{5}$.

Figure 5.3 illustrates the size of the compressed symbol and orders streams as the text grows. Again this is for the Chinese Bible text as an illustration (and similar results are evident for the other files). As we have discussed in section 5.3, the output size of the orders stream is increasing much slower than that of the symbols stream because of its small

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

Tab. 5.8: Percentage of the symbol and orders stream in the PPMO output compressed file.

alphabet size and repetitive nature. The graph shows a noticeable dip in the orders stream compressed output with larger input text, whereas the symbols stream output and combined overall output has reached a plateau.

5.5 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. It has not been reported by the other study, which successfully used a secondary order encoding process to process the text.

Some other language models such as the *Interpolated language model* [JBM75] are widely used in the speech recognition area. These models

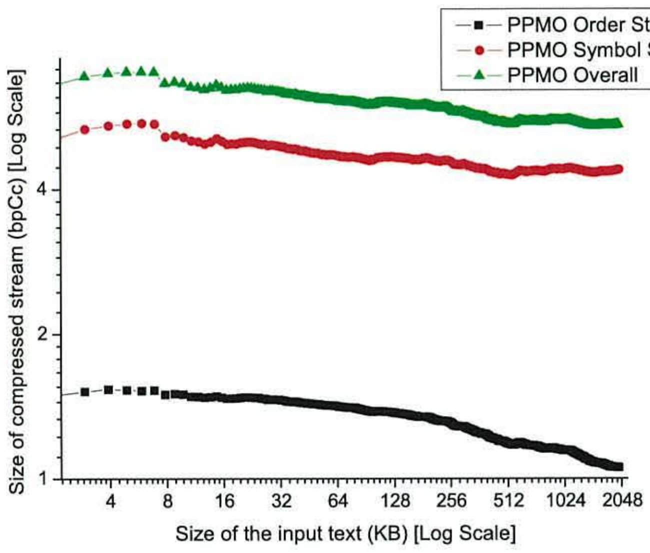

Fig. 5.3: The size of the symbol and orders stream in the PPMO output compressed file as the text grows.

assign a weight to each order model, and then calculate the weighted sum of the probabilities. Different interpolation algorithms such as held-out interpolation [JBM75] and word clustering interpolation [BK05], have been devised based on 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 community, it is also of interest to the wider language modelling community where the adaptive mechanism for encoding the orders may also 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.

This algorithm should also achieve similar improvements in other large alphabet size languages such as Japanese, Korean and Thai.

Further work in this area is required to optimize the coding scheme. It is not clear how the improvements we have detailed here would affect other PPM-based implementations.

6 Word and Tag Based Models for Chinese Text Compression

Contents

6.1 Introduction

In the previous chapters, we have looked at the problem of encoding Chinese text using characters as the unit of encoding. Different problems occur when we consider different units of encoding such as words and part of speech tags. Many traditional language models actually use either a word or part of speech based approach for their applications such as speech recognition and machine translation [BPM+92, Jel90].

Word-based models use the previous few words in the context to predict the upcoming ones. It provides faster compression than the character based approach since fewer symbols are being processed. A series of word-based text compression schemes have been proposed [HC92, Mof89, Tea98].

Tag-based models use part of speech tags to help prediction. Teahan [Tea98] in his thesis states that :

> "The advantage of using the tag is that it may have occurred many times previously and so 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."

In this chapter, we extend the word and tag based approach by Teahan

[Tea98] especially for Chinese language. The structure of this chapter is as follows: we first examine previous work in Section 6.2, then adapt our PPMO model for words and tags in Section 6.3. Section 6.4 details our experimental results. We provide conclusions and a discussion in Section 6.5.

6.2 Previous Work

Several word and tag based models have been described that exploit the blending mechanism of the PPM compression scheme [Tea98, Mof89]. Higher order contexts are tried first, but if the next word has not been seen before in this context then an escape symbol is encoded and a lower order context is used instead. Various escape algorithms were described in [Tea98] and in [WB91]. Experiments with word and tag based models show that method X1 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.

Experiments with English text [Tea98] indicate that the best-performing models are those shown in Table 6.1. It is not clear, however, whether the same models will be best for Chinese. Indeed, the results described in

C|C^5 model	W|W model	W|TW model	T|TWT model
$p(c_i\|\, c_{i-1}c_{i-2}c_{i-3}c_{i-4}c_{i-5})$	$p(w_i\|\, w_{i-1})$	$p(w_i\|\, t_i w_{i-1})$	$p(t_i\|\, t_{i-1}w_{i-1}t_{i-2})$
$\hookrightarrow p(c_i\|\, c_{i-1}c_{i-2}c_{i-3}c_{i-4})$	$\hookrightarrow p(w_i\|\,)$	$\hookrightarrow p(w_i\|\, t_i)$	$\hookrightarrow p(t_i\|\, t_{i-1}w_{i-1})$
$\hookrightarrow p(c_i\|\, c_{i-1}c_{i-2}c_{i-3})$	$\hookrightarrow$ character model	$\hookrightarrow$ character model	$\hookrightarrow p(t_i\|\, t_{i-1})$
$\hookrightarrow p(c_i\|\, c_{i-1}c_{i-2})$			$\hookrightarrow p(t_i\|\,)$
$\hookrightarrow p(c_i\|\, c_{i-1})$			$\hookrightarrow p_{eq}(t_i\|\,)$
$\hookrightarrow p(c_i\|\,)$			
$\hookrightarrow p_{eq}(c_i\|\,)$			

Tab. 6.1: Some models for predicting character, tag and word streams [Tea98].

this chapter show that other variants perform better (see Section 6.4).

We use the same approach adopted by Teahan (1998) to name the algorithms, with some minor modifications. The first model, labelled C|C^5, is an order 5 PPM model that bases the predictions on the stream of character symbols. Conceptually, for a sequence S of length m words, it is represented by the following formula:

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

where p' gives the probabilities returned by the C|C^5 model shown in Table 6.1 and is represented, for example, by the formula $P_{PPMC}(\varphi)$—this will vary depending on the escape method chosen. (Note that in the table, the symbol $\hookrightarrow$ represents an escape).

The second model[1], labelled W|W, is an order 1 model for predicting the stream of word symbols. It first predicts the word using just the previous word. If the word is not predicted, then the model escapes down to an order 0 model. It 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 shown in Table 6.1.

The third model, labelled W|TW, encodes the word using two streams—the stream of tags (or symbols representing parts of speeach such as noun, determiner and verb) and the stream of words. This model first predicts the tag using the model labelled T|TWT in the last column, then predicts the word using its tag and the previous word as the context. If this is unsuccessful, it tries to proceed based on the current tag only, otherwise it escapes down to the character based model.

T|TWT was found to be the best model for predicting the tags (Teahan, 1998). The current tag is first predicted using the prior tag, the prior word and the tag preceding that. If necessary, the model escapes and uses just

[1] This model is similar to Moffat's (1989) WORD model, which uses escape method C instead of X1 and encodes the vocabulary by using an order 0 model to encode the word lengths, followed by an order 0 model to encode the letters in each word.

the previous tag, then escapes again and predicts without any context. The first time a new tag is seen it will not have been recorded even in this model, so there is a final escape to a model which predicts each tag with equal probability.

For word prediction, the W|TW model is combined with either an order 2 tag model (T|TT) or the T|TWT model. The combined W|TW & T|TWT model, for example, 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 T|TWT model shown in Table 6.1 and p'' gives the probabilities returned by the W|TW model.

If the word for the W|W and W|TW models is not predicted at all (i.e. the word has not previously been seen), then a further escape is encoded down to a character based model, where each character in the word (including the space character to mark the end of the word) is encoded separately. All words encoded at this level are either unique (for the W|W model) or have been tagged uniquely (for the W|TW model), so in a sense they can be regarded as the "vocabulary" of the text.

Two methods present themselves for encoding this vocabulary. The first method builds a PPM character model (C|C^4) using all the previous characters in the text, then uses this to predict the characters. The second method exploits the concept of update exclusions using a process dubbed *character exclusion*—the model is built using only the characters contained within the vocabulary itself. Experiments in Teahan's thesis [Tea98] have shown that this method consistently outperforms the first and that an order 4 PPMD model performs best at encoding the vocabulary.

6.3 PPMO Word and Tag Based Models

The main motivation behind the PPMO algorithm is to explore an alternative multi-stream method which obviates the need for performing escapes. The coding process is split into an orders stream and symbols stream. The orders stream outputs a coding order for each coding symbol, whereas the symbols stream outputs the coding symbol itself based on the context for the particular order.

To illustrate this algorithm, the same definitions which were mentioned in the previous section are used. For instance, the combined model, using an order 2 model to encode the orders stream, and an order 2 model to encode the symbols stream, is represented by the following formula:

$$p(S) = \prod_{i=1}^{m} p'(o_i|o_{i-1}, o_{i-2})\ p''(s_i|s_{i-1}s_{i-2})$$

where p' gives the probabilities returned by the O|O^2 model shown in Table 6.2 and p'' gives the probabilities returned by the S|S^2 model (which are estimated using the formula for $P_{PPMO}(\varphi)$ in Section 5.3).

O|O^2 model	**O|S model**	**S|S^2 model**	
$p(o_i\| o_{i-1}o_{i-2})$	$p(o_i\| s_{i-1})$ if $c(o_{i-1}o_{i-2}) > t_{OS}$,	$p(s_i\| s_{i-1}s_{i-2})$	if $o_i = 2$,
$\hookrightarrow p(o_i\| o_{i-1})$	O|O^2 model otherwise	$p(s_i\| s_{i-1})$	if $o_i = 1$,
$\hookrightarrow p(o_i\|)$		$p(s_i\|)$	if $o_i = 0$,
$\hookrightarrow p_{eq}(o_i\|)$		$p_{eq}(s_i\|)$	if $o_i = -1$

Tab. 6.2: Some models for predicting the orders and symbols streams for PPMO.

Table 6.2 list some models for predicting the orders and symbols streams for PPMO. The model labelled O|O^2 on the left is the traditional order 2 PPM-based approach used in Chapter 4. We introduce in the second column an alternative model–labelled O|S for encoding the orders stream. It bases the prediction for the order (in the orders stream) on the previous symbol (from the symbols stream) if some threshold total count t_{OS} for the context $o_{i-1}o_{i-2}$ is exceeded, otherwise it reverts to the O|O^2 model. The S|S^2 is the model used for encoding the symbols stream.

When decoding, the orders stream is first decoded, to find out which order the symbol has been encoded with, and then the symbols stream is decoded based on the order of the model.

It is important to note that PPMO can be used as an alternative coding method wherever PPM is used. For example, it can replace the PPM models for all of the models listed in Table 6.1. Tables 6.3 and 6.4 list several PPMO-based variants for encoding characters and words. The left columns of both tables list the traditional PPM models for the respective character and word streams. The right columns, on the other hand, list the PPMO variants. For naming conventions, to distinguish the different models, the PPMO variants of the C|C^n and W|W models are now labelled as C$_o$|C$_o^n$ and W$_o$|W$_o$ respectively. To distinguish the different orders streams, the orders have also been sub/super-scripted —O$_c$ and o^c for character orders, and O$_w$ and o^w for word orders.

C|C^2 model (single stream)	**C$_o$|C$_o^2$ model (with character orders stream models O$_c$|O$_c^n$ or O$_c$|C)**	
$p(c_i\| c_{i-1}c_{i-2})$	$p(c_i\| c_{i-1}c_{i-2})$	if $o_i^c = 2$,
$\hookrightarrow p(c_i\|c_{i-1})$	$p(c_i\| c_{i-1})$	if $o_i^c = 1$,
$\hookrightarrow p(c_i\|)$	$p(c_i\|)$	if $o_i^c = 0$,
$\hookrightarrow p_{eq}(c_i\|)$	$p_{eq}(c_i\|)$	if $o_i^c = -1$

Tab. 6.3: Two C|C^2 variants for predicting characters.

W|W model (single stream)	**$\mathbf{W}_o\|\mathbf{W}_o$ model (with word orders stream models $\mathbf{O}_w\|\mathbf{O}_w^n$ or $\mathbf{O}_w\|\mathbf{W}$)**	
$p(w_i\| w_{i-1})$	$p(w_i\| w_{i-1})$	if $o_i^w = 1$
$\hookrightarrow p(w_i\|)$	$p(w_i\|)$	if $o_i^w = 0$
$\hookrightarrow$ character model	character model	if $o_i^w = -1$

Tab. 6.4: Two W|W variants for predicting words.

For word-based modeling, multiple options can now be considered. Table 6.5 in the first column lists as a sequence of symbols (words, word orders, characters and character orders) the four possible streams that can be used for word encoding; this is followed by a short description in the second column. The right column lists the models by which the stream can be encoded. It is clearly evident that now there are many more possibilities to use. Whereas previously there was just a single variant (W|W & C|C^n), *nine* variations are now possible—as there are three ways of modeling the seen word stream (W|W; W$_o$|W$_o$ & O$_w$|O$_w^n$; W$_o$|W$_o$ & O$_w$|W), and this is multiplied by the three ways of modeling the unseen word characters stream (C|C^n; C$_o$|C$_o^n$ & O$_c$|O$_c^n$; C$_o$|C$_o^n$ & O$_c$|C).

Stream	Description	Models
$\ldots w_{i-2}\, w_{i-1}\, w_i \ldots$	stream of seen word symbols	$\mathrm{W}\|\mathrm{W},\ \mathrm{W}_o\|\mathrm{W}_o$
$\ldots o^w_{i-2}\, o^w_{i-1}\, o^w_i \ldots$	stream of seen word orders	$\mathrm{O}_w\|\mathrm{O}^n_w,\ \mathrm{O}_w\|\mathrm{W}$
$\ldots c^{nw}_{j-2}\, c^{nw}_{j-1}\, c^{nw}_j \ldots$	stream of unseen word character symbols	$\mathrm{C}\|\mathrm{C}^n,\ \mathrm{C}_o\|\mathrm{C}^n_o$
$\ldots o^{nw}_{j-2}\, o^{nw}_{j-1}\, o^{nw}_j \ldots$	stream of unseen word character orders	$\mathrm{O}_c\|\mathrm{O}^n_c,\ \mathrm{O}_c\|\mathrm{C}$

Tab. 6.5: Multiple streams for word-based encoding.

6.4 Experimental results

The purpose of this section is to report on some experimental results that show that for Chinese text many of the PPMO variants indicated in the previous section outperform the standard PPM in certain configurations and applications. Just as interestingly, the results detailed below show that the results are very different when comparing a large alphabet language (Chinese) to a small alphabet language (English).

Table 6.6 lists results for various PPM and PPMO variants using byte, character, word and tag models for the files in the LCMC. The third column lists results for a standard order 4 PPMD model on the byte stream. The column labelled PPM-ch lists results for PPM-ch on the Chinese character stream (these are mostly two bytes each, but sometimes are a single byte). PPMO in the last column also works with the Chinese character stream. It comprises the following two models to encode the character or-

file	**size**	**PPMD**	**PPM-ch**	$\mathbf{W}_o\|\mathbf{W}_o$	$\mathbf{W}_o\|\mathbf{W}_o$	**W|TW**	**PPMO**
		Order 4	Order 2	(Variant I)	(Variant II)	(Variant III)	
	(bytes)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)	(bpCc)
LCMC-A	271035	8.390	7.926	7.932	7.925	8.028	**7.778**
LCMC-B	170469	7.826	7.420	7.273	**7.266**	7.417	7.268
LCMC-C	112681	7.176	6.808	6.668	**6.661**	6.783	6.696
LCMC-D	103031	8.122	7.624	7.689	7.686	7.904	**7.500**
LCMC-E	218473	8.253	7.778	7.796	7.787	7.901	**7.628**
LCMC-F	266226	8.330	7.878	7.866	7.859	7.965	**7.740**
LCMC-G	452511	8.316	7.896	7.913	7.910	7.963	**7.776**
LCMC-H	208035	5.983	5.756	5.700	5.691	5.722	**5.666**
LCMC-J	509095	7.038	6.802	6.623	6.618	6.646	**6.702**
LCMC-K	159118	8.697	8.094	8.345	8.343	8.506	**8.030**
LCMC-L	137743	8.624	8.006	8.138	8.134	8.312	**7.934**
LCMC-M	35159	8.818	8.086	8.320	8.319	8.703	**8.020**
LCMC-N	155897	8.572	8.016	8.163	8.165	8.347	**7.940**
LCMC-P	273109	6.429	5.856	**4.777**	**4.777**	4.848	5.482
LCMC-R	49289	8.514	7.848	8.091	8.090	8.361	**7.802**

Variant I:	$\mathrm{W}_o\|\mathrm{W}_o$ & $\mathrm{O}_w\|\mathrm{O}_w^2$ & $\mathrm{C}\|\mathrm{C}^2$ & $\mathrm{O}_c\|\mathrm{O}_c^2$.
Variant II:	$\mathrm{W}_o\|\mathrm{W}_o$ & $\mathrm{O}_w\|\mathrm{O}_w^2$ & $\mathrm{C}\|\mathrm{C}^2$ & $\mathrm{O}_c\|\mathrm{C}$ ($t_{OC} = 8$).
Variant III:	W|TW & T|TWT & $\mathrm{C}\|\mathrm{C}^2$ & $\mathrm{O}_c\|\mathrm{O}_c^2$.
PPMO:	$\mathrm{C}\|\mathrm{C}^2$ & $\mathrm{O}_c\|\mathrm{O}_c^2$

Tab. 6.6: Results for PPM and PPMO variants on the LCMC.

der and symbols streams: $\mathrm{C}|\mathrm{C}^2$ & $\mathrm{O}_c|\mathrm{O}_c^2$. The order 1 word models (labelled Variant I and II) are the two best performing of the word model variants that use a PPMO-based method to encode both the seen and unseen word character streams. Variant II is a slight variation to Variant I that conditions the unseen word character order on the previous character if a threshold total count $t_{OC} = 8$ for the context $o_{i-1}o_{i-2}$ is exceeded, otherwise it reverts to the the $O|O^2$ model. The tag model (Variant III) also uses a PPMO-based method to encode the unseen word character stream.

6.5 Conclusions and Discussion

In this chapter, we compared word and tag based variants of the PPMO algorithm as discussed in the previous chapter. The results show that the best performing model is PPMO, followed by PPM-ch, both of which use the character stream, and then the order 1 word models (Variants I and II). The tag model (Variant III) does not perform as well as these models, and perhaps this may be due to the tagset adopted by the LCMC. All of these models clearly perform better than the standard byte-stream PPMD model.

These results are interesting in that the performance order of the models (character models are best, followed closely by word models, then by tag models) is exactly *opposite* to that observed for English text (where tag models are best, followed closely by word models, then by character models). To some extent, this may be a reflection of the nature of the Chinese alphabet, where characters have greater semantic content and are closer to the English notion of a word.

These results highlight that there are alternatives to the traditional

PPM coding algorithm that may be more appropriate for text (such as Chinese text) which has different properties to the standard text (ie. English) usually reserved for experimental evaluation.

7 PPM-ch and PPMO Based Chinese Word Segmentation

Contents

7.1 Introduction

Chinese word segmentation is an important initial step in processing Chinese language. This is because the word is the smallest meaningful unit in the language context. A good word segmentation model is fundamental for many NLP applications such as information retrieval and machine translation.

Unlike English, which uses a space as its delimiter between the words, Chinese text is written without using any spaces or delimiters between each word. For a Chinese native speaker, segmenting Chinese is relatively straightforward based on his own intuition. However, for NLP, segmenting Chinese text into words is still problematical.

The purpose of this chapter is to apply the new models which were developed in Chapter 4 & 5 to a specific NLP application — word segmentation. From this we can gauge whether these models also have potential for natural language processing beyond the main application that has been investigated so far i.e. text compression.

The structure of this chapter is organized as follows: Section 7.2 show previous methods used for Chinese word segmentation. We illustrate our PPMO model for Chinese word segmentation in section 7.3. Section 7.4

shows the experimental results of using our PPMO model to segment Chinese text. Section 7.5 summarizes our study and discusses some future research issues.

7.2 Previous Work

Chinese word segmentation faces many problems. Wu and Tseng [WT93] in their paper made a detailed survey about these problems. In general, the main problems that need to be overcome for word segmentation is the ambiguities of segmentation strings and identification of the new unknown words [WT93, SSGC96, GLH03].

A Sentence in Chinese	乒乓球拍卖完了
Segment Method 1	乒乓•球拍•卖•完•了
Segment Method 2	乒乓球•拍卖•完•了

Tab. 7.1: A Chinese sentence with ambiguity of phrasing.

Table 7.1 shows an example of a Chinese sentence that has an ambiguity in its phrasing. The sentence can be interpreted into two different meanings by using different segmentations. The first segmentation method interprets this sentence as a five-words sentence and its English meaning is *The ping pong rackets have been sold out.* However, the second method segments the same sentence into four words, and the meaning is *The auction for ping pong has finished.*

A Sentence in Chinese	Correct Segmentation
王军虎去广州了	王军虎•去•广州•了
王军虎头虎脑的	王军•虎头虎脑•的

Tab. 7.2: An example of Chinese New unknown word identification.

In Chinese, most new unknown words are a name of a person, an organization, a place, a product or an abbreviated name. Table 7.2 shows examples of identifying a person's name. The first three characters "王军虎" occur in both of the two example sentences. Native Chinese speakers can easily segment the first sentence into "王军虎•去•广州•了" where "王军虎" is actually a person's name (The English translation is *Junhu Wang has been to Guangzhou*). In the second sentence, "王军" is a person's name and "虎头虎脑" is a Chinese idiom which normally states that a child looks naive and fat. It is possible that the character "虎" (*Hǔ* Tiger) can be a component of either a person's name or be a part of a Chinese idiom. However, for the computer, because of the zero frequency problem and sparse data problems, there will be many unseen or infrequent word errors e.g. "王军虎" could be erroneously segmented as "王•军•虎", a three separate character substring.

To solve these problems, many different algorithms have been proposed. They are roughly classified into dictionary-based and statistical-based approaches, although in practice many algorithms use hybrid approaches.

Cheng *et al.* introduced a dictionary-based method [CYW99]. Given a Chinese word frequency dictionary, an input character string searches the dictionary by using the *maximum forward match* heuristic, to find the one that matches the greatest number of characters of the input. However, for dictionary-based approaches, the performance of word segmentation greatly depends on the coverage of the underlying dictionary, but since new words appear constantly, the dictionary will never be complete. That is why in many dictionary-based approaches, there are some special components for new word identification [CYW99].

Statistical-based methods can to some extent resolve these problems. Sproat implemented special adaptive recognizers not only for Chinese names and transliterated foreign names, but for components of morphologically obtained words as well [SSGC96]. Gao [GGLL02, GLH03] proposed a standard statistical approach by using a trigram model to train a 1.6 billion characters corpus. The initial lexicon was built after training. Several optimization techniques were then used to optimize this lexicon to achieve better segmentation results. Xue [Xue03] used a maximum entropy tagger trained on manually annotated data to automatically assign to Chinese characters tags that indicate the position of a Chinese character within a word. The tagged output is then converted into segmented text for evaluation.

7.3 Using PPM for Chinese Word Segmentation

Teahan et. al. [TWMW00] introduced a compression based algorithm for Chinese word segmentation. The idea of this approach is to view a word segmentation problem as a problem of inserting a space into a hidden Markov chain by using a PPM model.

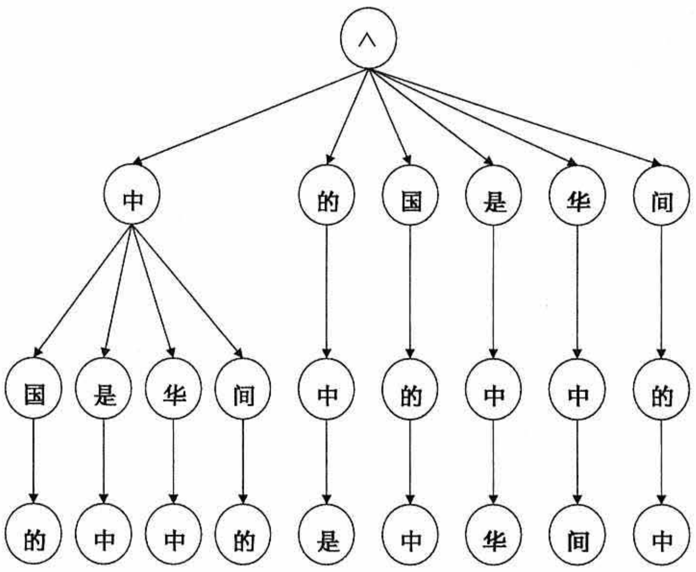

Fig. 7.1: Context trie for the string "中国的中是中华中间的中".

Normally PPM models use a trie structure to store the context relationship. Figure 7.1 shows an example of an order 2 context trie for the string "中国的中是中华中间的中". In this structure, an input string is represented as a path from the root node to a trie node. If the context needs to be extended, it is only necessary to move the input pointer forward by one position.

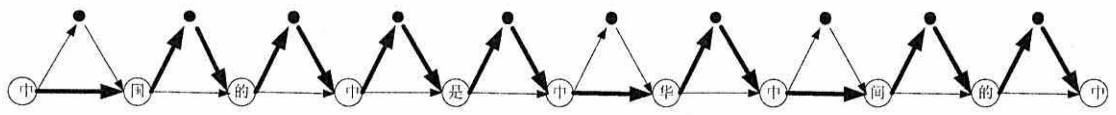

Fig. 7.2: Hidden Markov model for space insertion using an order zero model in the string "中国的中是中华中间的中".

To insert spaces into the text, a hidden Markov modelling procedure can be used. Figure 7.2 shows how to use a hidden Markov model for space insertion using an order zero model in the string "中国的中是中华中间的中". Each character is represented as a node and each possible space insertion is an inter-character node (illustrated as dots in the Figure where the dot represents the space character). We seek to find the path which gives the best PPM text compression performance, from the beginning to end in all possible paths. In this example, the correct path is "中国 • 的 • 中 • 是 • 中华 • 中间 • 的 • 中", shown in bold arrows in Figure

7.2. Similarly, we can readily extend this approach by using higher orders of PPM for this hidden Markov modelling procedure.

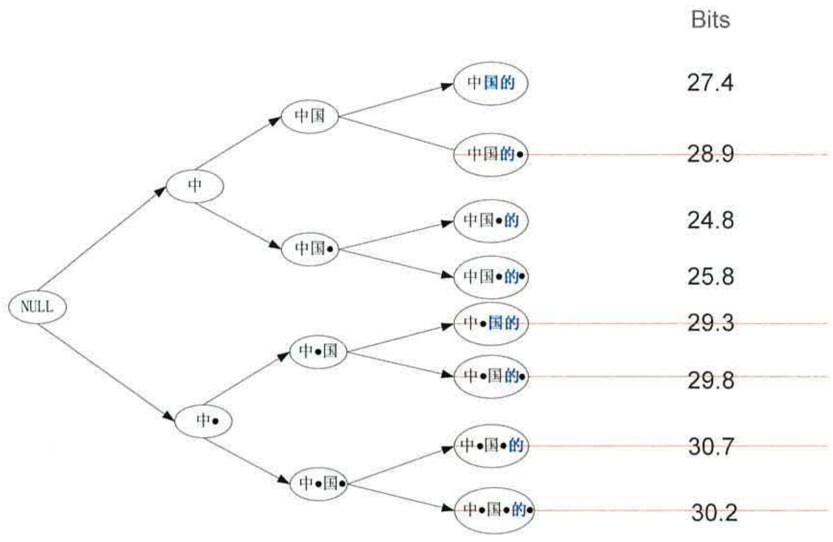

Fig. 7.3: The space insertion procedure for the Chinese word segmentation using an order 2 model.

To find out the path with best compression performance in Figure 7.2, a Viterbi-style algorithm [Vit67] can be used to solve the word segmentation problem. The initial state starts from NULL, and at each possible state the highest probability of reaching that state from the beginning has been recorded. The problem of segmenting the string "中国的中是中华中间的中" can be regarded as the process to find the maximum probability path according to a trellis-based search tree from several space insertion alternatives such as "中•国" and "中国•". There are 8 possible combinations of space insertions for the substring "中国的" and these are shown in Figure 7.3. The tree has a branching factor of 2 as there are two possibilities in the way each character is processed — either left unchanged (the higher branch) or with a space inserted after it (the lower branch). Because of the Markov assumption, and if we only use an order 2 model (as in Figure 7.1) we can discard poorer performing paths which have the same order 2 context as no matter what happens in the future, these paths can not perform better than the other paths. If we encode these 8 possible strings using an order 2 PPM-ch models, the highest probability paths are 27.4 bits, 25.8 bits and 24.8 bits respectively, and the rest can be discarded.

So we just keep these three optimal state paths and extend the calculation to the next state. A path with maximum probability is therefore calculated once the final node is reached and this path is deemed the best word segmentation path for the whole context.

The variants of PPM models, PPM-ch and PPMO, can be easily adapted for this algorithm. In the next section, we show our experiment results by using these models. Although our PPM models have not been carefully crafted for segmentation usage, we still achieve significant results.

7.4 Experimental Results

To compare our experimental results with other approaches, we used the Second International Chinese Word Segmentation Bakeoff corpora as our experimental corpora [SIG05], which can be downloaded at *http://sighan.cs.uchicago.edu/bakeoff2005/* .

Table 7.3 shows four corpora that are used in this evaluation. Each corpus comprises 3 different sub-corpora: the training corpus for building up an initial segmentation model, the testing corpus for testing the segmentation model and the hand-segmentation gold standard corpus.

Corpus	**Encoding**	**Word Types**	**Words**	**Character Types**	**Characters**
Academia Sinica	Big5	141,340	5,449,698	6,117	8,368,050
HongKong City University	Big5	69,085	1,455,629	4,923	2,403,355
Peking University	GB2312-80	55,303	1,109,947	4,698	1,826,448
Microsoft Research	GB2312-80	88,119	2,368,391	5,167	4,050,469

Tab. 7.3: The Second International Chinese Word Segmentation Bakeoff Corpora

To evaluate the accuracy of our segmentation results, we used three measures: *recall rate*, *precision rate* and *error rate*.

Let N be number of words occurring in the hand-segmentation. Let e

be number of words incorrectly identified by the automatic method and c be number of words correctly identified by the automatic method, $n = c+e$ be number of words identified by the automatic method. Then we have:

$$\begin{aligned}\text{Recall Rate (\%)} &= \frac{c}{N} \times 100,\\ \text{Precision Rate (\%)} &= \frac{c}{n} \times 100,\\ \text{Error Rate (\%)} &= \frac{e}{N} \times 100.\end{aligned}$$

Recall and precision are standard information retrieval measures used to assess the quality of a retrieval system in terms of how many of the relevant documents are retrieved (recall) and how many of the retrieved documents are relevant (precision) [TWMW00].

We also used the F-measure to compare our results with others, where:

$$F\text{-measure} = \frac{2 \times \text{Precision Rate} \times \text{Recall Rate}}{\text{Precision Rate} + \text{Recall Rate}}.$$

If the automatic method produces the same number of words as the hand-segmentation, recall and precision both become equal to one minus the error rate. A perfect segmenter will have an error rate of zero and recall and precision of 100% [TWMW00].

Table 7.4 shows our experimental results when using different PPM-Based models. The PPM-ch model uses our pre-processing techniques

Test Corpus	File Size	Segment model	Recall Rate	Precision Rate	F-Measure
	(Bytes)		%	%	%
HongKong City University	136,040	PPMD	**92.0**	93.6	**92.8**
		PPM-ch	89.8	**93.7**	91.7
		PPMO	88.9	92.1	90.5
Academia Sinica	422,268	PPMD	93.4	93.6	93.5
		PPM-ch	**93.8**	**94.7**	**94.3**
		PPMO	93.1	93.4	93.2
Microsoft Research	376,280	PPMD	94.8	95.8	95.3
		PPM-ch	**95.2**	**96.9**	**96.0**
		PPMO	94.3	95.5	94.9
Peking University	343,120	PPMD	**90.4**	93.2	**91.8**
		PPM-ch	89.2	**94.0**	91.5
		PPMO	88.2	92.7	90.4

Tab. 7.4: Chinese Word Segmentations by using different PPM-Based Models.

that were discussed in Section 4.3, but the PPMD model does not use any pre-processing techniques and is based on processing the ASCII byte stream in the same way as the method used by Teahan et al. [TWMW00]. The PPMO model is our normal PPMO model that was discussed in Chapter 5. All three of these PPM variants use an order 2 model as their maximum order.

From the table we can see that the PPM-ch model outperforms the PPMO model in all cases. This is interesting in that the improved PPMO compression results detailed in Chapter 5 are not mirrored in improved segmentation results. Normally, better compression will result in better

Training Corpus	Segment model	Model Size (Bytes)
HongKong City University	PPMD	15,578,593
	PPM-ch	25,089,553
	PPMO	3,125,144
Academia Sinica	PPMD	29,625,054
	PPM-ch	64,923,877
	PPMO	6,255,521
Microsoft Research	PPMD	17,957,111
	PPM-ch	34,094,699
	PPMO	3,841,597
Peking University	PPMD	11,184,531
	PPM-ch	18,073,623
	PPMO	2,395,369

Tab. 7.5: Size of different PPM-based Chinese word segmentation models.

application performance, but this is not the case for PPMO.

However, when we look at the size of the different PPM-based models, as shown in Table 7.5, an important notable difference is that models that were built by PPMO are very compact, significantly smaller than that of PPMD. The size of the PPMO model is only 2 or 3 MB, but both the PPMD and the PPM-ch models are all above 15 MB.

Table 7.6 shows the comparison between PPM-ch, PPMO and the winning result of the Second International Chinese Word Segmentation Bakeoff. Although our PPM-ch and PPMO models have not been heavily optimized for word segmentation, they still achieve excellent results, only 1%

Test Corpus	Segment model	Recall Rate	Precision Rate	F-Measure
		%	%	%
HongKong City University	Winning Bakeoff Result	94.6	94.1	94.3
	PPMO	88.9	92.1	90.5
	PPM-ch	89.8	93.7	91.7
Academia Sinica	Winning Bakeoff Result	95.1	95.2	95.2
	PPMO	93.1	93.4	93.2
	PPM-ch	93.8	94.7	94.3
Microsoft Research	Winning Bakeoff Result	96.6	96.2	96.4
	PPMO	94.3	95.5	94.9
	PPM-ch	95.2	96.9	96
Peking University	Winning Bakeoff Result	94.6	95.3	95.0
	PPMO	88.2	92.7	90.4
	PPM-ch	89.2	94.0	91.5

Tab. 7.6: PPM-ch and PPMO based and The winning bakeoff Chinese Word Segmentations result.

– 5% worse than the best results.

7.5 Conclusion and Discussion

In this chapter, we have applied the PPMO and PPM-ch models to Chinese word segmentation. Both PPM-ch and PPMO show a competitive result for the application of Chinese word segmentation, although they have not been heavily optimized.

Compared with PPMO, PPM-ch achieves the better word segmentation result. This is interesting because normally better text compression modules yield better performance in other applications. One possible reason for this result is that the PPMO model separates the coding process into an orders and symbols stream, the symbol's prediction is only according to the symbols stream, and each symbol is encoded by a fixed order. Traditional PPM models encode in different orders by assigning the escape probability. It is possible that the escape symbol helps the word segmentation result, as it penalizes the model for sequences it has not seen before. Further experiments should be tested to prove this assumption. However, results for PPMO are promising despite not being heavily optimized.

Significantly, however, PPM-ch is competitive with the standard PPM approach, and outperforms it on a number of cases, although also not being optimized for the word segmentation problem. The PPM approach [TWMW00] was used as the benchmark for the Second International Chinese Word Segmentation Bakeoff evaluation, so improvements over this benchmark is significant and indicates promise in the approach taken.

These results indicate that the new PPM algorithms have potential to be used for NLP. Possible further applications such as text categorization [TH01b] and text mining using compression models [TH01a] should be investigated.

8

Summary and Further Work

Contents

8.1 Summary

This thesis has been concerned with building adaptive models of Chinese text. We proposed several improvements for traditional PPM models and developed a novel variant of PPM, which is named PPMO. This new model achieved excellent experimental results. The significance of the results obtained from the investigations are described as follows.

8.1.1 PPMO: Coding Chinese Characters Without Using Escapes

Tradition PPM models use escape as their blending scheme, to avoid the zero frequency problem and to combine different probability predictions that were assigned by different context models into one overall probability. Different PPM variants are normally named according to their probability escape algorithms.

However, the PPMO model is the first PPM variant that does not use the escape mechanism. This algorithm divides the coding process into two separate streams, named the orders stream and the symbols stream. The order of the model is encoded first, then we encode the symbol itself. This algorithm achieves the best compression results for Chinese text.

This algorithm is not just applicable for the Chinese language. It can be applied to other large alphabet languages, but may also be applicable to other (non PPM) types of language models.

8.1.2 PPM-ch: Adaptive PPM models for Chinese Text.

We made several changes in the PPM model to adapt it specifically to the Chinese language. Changing the symbol encoding unit to 16 bits captures the structure of the language precisely. Sorting all the characters in the context by frequency order improves the program speed significantly and using no exclusions also leads to faster execution speed.

In our experiments, we show that PPM-ch outperforms all traditional PPM models. We also found that for Chinese text, the proportion of escape probabilities that need to be encoded is much higher than for English text, which leads to wasted code space. This insight motivated us to develop our new PPMO models.

8.1.3 Chinese Word and Tag Based Models

Both PPMO and PPM-ch models encode the character stream, but they can be adapted for word and tag based symbol streams as well. Our experiments show some interesting results as normally for English text, the tag and word based models achieve the best compression results, followed by the character based models. However, for Chinese text, an opposite result has been found, where the character-based PPMO achieved the best performance, followed by the character-based PPM-ch and then the word-based models (see 6.4). The tag based model do not perform as well as these models.

8.1.4 Chinese Word Segmentation

Word segmentation is an important NLP application, especially for the Chinese language. This is because in the Chinese text, there is no any delimiters to sperate the sentence into words.

We have shown that our PPMO and PPM-ch models perform well when applied to Chinese word segmentation. Both PPM-ch and PPMO achieved competitive results, although they have not been heavily optimized.

We also found an interesting result that PPM-ch performs better than PPMO model in Chinese word segmentation, although PPMO achieves better results at compression.

8.2 Conclusions and Discussion

As we have discussed in chapter 1, the primary aim of this thesis is the development of effective Chinese language models for text compression. More specifically, we have answered the following questions that were listed in section 1.2:

What is the best computer model for compressing Chinese text?

Of the models we investigated, PPMO is the best computer model for compressing Chinese text. PPM-ch was slightly worse than PPMO but still outperformed the traditional PPM models.

What are the disadvantages of the current models when they are applied to Chinese text?

The main drawback of the traditional PPM models originates from the large alphabet characteristics of Chinese language. This leads to many disadvantages, not only in the compression performance aspect, but also in the execution speed aspect.

We have shown that the traditional PPM models encode characters as 8 bits symbols although in reality each Chinese character is 16 bits. From the experiments, we have found that when encoding Chinese text by using the traditional PPM models, the encoder outputs a large proportion of escape symbols, which actually lowers the prediction accuracy.

Furthermore, we found execution speeds for processing Chinese text are much lower than that of English.

What is the best way of dealing with the large alphabet size problem in Chinese?

To solve the large alphabet problem, we have proposed a preprocessing technique. This technique has been used in both the PPM-ch and PPMO models. Our experiments have shown that the pre-processing work can improve the compression rate significantly.

To solve the large proportion of escapes problem in the traditional PPM models, we also proposed PPMO, a novel PPM variant. The significance of the PPMO model is that it provides an alternative way for estimating probabilities in PPM rather than using the escape mechanism. To avoid the escape mechanism that was used in all previous traditional PPM models, PPMO proposes a new blending scheme that separates the PPM coding process into the orders stream and the symbols stream. The trade-off between the PPMO algorithm and the traditional PPM algorithms is between encoding the orders stream separately versus additional code space from encoding escapes and backing off to lower orders. Our experiments show that at least for Chinese text, this new multi-stream blending scheme achieves the best performances not only in character-based models, but also in word and tag based models.

How well do the best models perform in several natural language processing applications?

Our PPM-ch and PPMO is designed especially for Chinese text compression. Additionally, we have shown that they can also achieve competitive performance for Chinese word segmentation despite not being heavily optimized.

This provides encouragement that other applications such as text categorization and text mining using our models might also yield competitive results. One would expect, PPMO and PPM-ch could achieve better NLP performance in these areas.

8.3 Further Work

The idea of the PPMO model originates from the insights of the experimental results from PPM-ch and specifically the higher proportions of escapes that occur for the traditional PPM algorithm, especially for Chinese text. That is the reason why we multi-stream the coding stream to avoid using the escape symbols for PPMO. The optimization of PPMO models can therefore be focused on both the orders stream and the symbols stream. One possible improvement is to change the method of determining the coding order, by using more complex standards or thresholds, rather than simply adapting the maximum coding coder as an orders stream in the current version. On the other hand, the symbols stream can also be optimized by adopting a more powerful probability estimator, rather than calculating the probability according to its actual frequency count.

The PPMO model can also be used in many NLP applications such as text categorization, and text mining. It should also be applicable to other large alphabet language such as Japanese, Koreans and Thai.

Our experiments also show that the Chinese language has some characteristics which are fundamentally different to that of the English. Future work is required to explain these differences.

Bibliography

[Abo07] About.com. Chinese characters for tattoos. *http://chinese culture.about.com/library/extra/character/bls_characters3.htm. Last visited 6th Feb,* 2007.

[AT06] J. Abel and W. J. Teahan. Text preprocessing for data compression. *Submitted to IEEE Transaction Computer*, 2006.

[Bau72] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[BCK04] E. Bodden, M. Clasen, and J. Kneis. Arithmetic coding revealed. *RWTH Aachen University*, 2004.

[BCP$^+$90] P. F. Brown, J. Cocke, S. D. Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[BCW90] T.C. Bell, J.G. Cleary, and I.H. Witten. *Text Compression*. Prentice Hall, New Jersey, 1990.

[BJM83] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2:179–190, 1983.

[BK05] N. Bassiou and C. Kotropoulos. Interpolated distanced bigram language models for robust word clustering. *International Workshop on Nonlinear Signal and Image Processing (NSIP 2005)*, 2005.

[BM89] T. Bell and A. Moffat. A note on the DMC data compression scheme. *Computer Journal*, 32(1):16–20, 1989.

[BPM$^+$92] P. F. Brown, V. J. Della Pietra, R. L. Mercer, S. A. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language, December 1992.

[BW94] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. 124, 1994.

[BWC89] Timothy Bell, Ian H. Witten, and John G. Cleary. Modeling for text compression. *ACM Comput. Surv.*, 21(4):557–591, 1989.

[BYY04] R. Begleiter, R. E. Yaniv, and G. Yona. On prediction using variable order Markov models. Technical Report CS-2004-07, Technion - Israel Institute of Technology, May 2004.

[CG91] K. W. Church and W. A. Gale. A comparison of the enhanced good.turing and deleted estimation methods for estimating probability of English bigrams. *Computer Speech and Language*, 5(5)(19-54), 1991.

[Che96] S. Chen. Building probabilistic models for natural language. *Ph.D. Thesis, Harvard University*, 1996.

[CHJP93] E. Charniak, C. Hendrickson, N. Jacobson, and M. Perkowitz. Equations for part-of-speech tagging. In *National Conference on Artificial Intelligence*, pages 784–789, 1993.

[Cho57] N. Chomsky. *Syntactic Structures*. The Hague:Mouton, U.S.A, 1957.

[CT97] J. G. Cleary and W. J. Teahan. Unbounded length contexts for PPM. *The Computer Journal*, 40(2/3), 1997.

[CW84] J.G. Cleary and I.H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, COM-32(4):396–402, April 1984.

[CYW99] K. S. Cheng, G. H. Young, and K. F. Wong. A study of word-based and integral-bit chinese text compression algorithms. *American Society for Information Science*, 50(3)(218-228), 1999.

[DeF84] J. DeFrancis. *The Chinese Language: Fact and Fantasy.* University of Hawaii Press, Hawaii, 1984.

[FK82] W. N. Francis and H. Kucera. The Brown corpus of standard american english. *Brown University, Providence, RI*, 1982.

[Fol06] C. Folkard. *Guinness World Records 2006.* Bantam USA, 2006.

[GGLL02] J. Gao, J. Goodman, M. Li, and K. F. Lee. Toward a unified approach to statistical language modeling for Chinese.

ACM Transactions on Asian Language Information Processing (TALIP), 1(1):3–33, 2002.

[GLH03] J. Gao, M. Li, and C. N. Huang. Improved source-channel models for Chinese word segmentation. *Microsoft Research, Asia Beijing 100080*, 2003.

[Goo53] I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264, 1953.

[Gu95] H.Y. Gu. Large-alphabet Chinese text compression using adaptive Markov model and arithmetic coder. *International Conference on Chinese Computing*, 9(2):111–124, 1995.

[Gu05] H.Y. Gu. A large-alphabet-oriented scheme for Chinese and english text compression. *Software- Practice and Experience*, 35:1027–1039, 2005.

[HB98] J. Hockenmaier and C. Brew. *Error-driven learning of Chinese word segmentation.* The 12th Pacific Conference on Language and Information, Singapore,Chinese and Oriental Languages Processing Society, 1998.

[HC86] R. N. Horspool and G. V. Cormack. Dynamic Markov modelling – a prediction technique. In *Proceedings International Conference on the System Services*, Honolulu, Hawaii, 1986. IEEE Computer Society.

[HC92] R. N. Horspool and G. V. Cormack. Constructing word-based text compression algorithms. In *Data Compression Conference*, pages 62–71, 1992.

[How93] P.G. Howard. *The design and analysis of efficeient lossless data compression systems*. Ph.D thesis, Brown University, Providence, Rhode Island, United States, 1993.

[JBM75] F. Jelinek, L. R. Bahl, and R. L. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, 21(3):250–256, 1975.

[Jel90] F. Jelinek. Self-organized language modeling for speech recognition. *Readings in speech recognition, edited by A.Waibel and K.Lee. Morgan Kaufmann Publishers*, pages 450–5–6, 1990.

[Jin93] G. Jin. PH: a Chinese corpus. *Communications of COLIPS*, 3(1):45–48, 1993.

[Kat87] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics Speech and Signal Processing*, 35(3):400–401, 1987.

[KCG90] M. Kemighan, K. W. Church, and W. A. Gale. A spelling correction program based on a noisy channel model, 1990.

[Lap14] P.S. Laplace. Essai philosophique sur les probabilites. *Paris:Mme. Ve. Courcier*, 1814.

[Len94] Y. L. Leng. *Chinese Character Dictionary*. Zhong Hua Press, P. R. China, 1994.

[Lua95] K.T. Lua. A minimum entropy approach for Chinese text compression. *Computer Processing of Chinese & Oriental Languages*, 9(2):155–161, 1995.

[Mar13] A. Markov. An example of statistical investigation in the text of 'eugene onyegin' illustrating coupling of 'tests' in chains. In *the Academy of Sciences, St. Petersburg*, volume 7(6), pages 153–162, 1913.

[MNW98] A. Moffat, R. Neal, and I. Witten. Arithmetic coding revisited. *ACM Trans. on Information Systems*, 16(3):256–294, 1998.

[Mof89] A. Moffat. Word-based text compression. *Software Practice & Experience*, 19(2):185–198, 1989.

[Mof90] A. Moffat. Implementing the PPM data compression scheme. *IEEE Transactions on Communications*, 38(11):1917–1921, 1990.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[MSWB94] A. Moffat, N. Sharman, I. H. Witten, and T. C. Bell. An empirical evaluation of coding methods for multi-symbol alphabets. *Information Processing and Management: an International Journal*, 30(6):791–804, 1994.

[MX04] T. McEnery and R. Xiao. The Lancaster corpus of Mandarin

Chinese. *European Language Resources Association*, February 2004.

[ON05] G. H. Ong and J. P. Ng. Dynamic Markov compression using a crossbar-like tree initial structure for Chinese texts. *ICITA '05: Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)*, 2(407–410), 2005.

[PC98] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Research and Development in Information Retrieval*, pages 275–281, 1998.

[Rab89] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[RJ86] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, January 1986.

[RL81] J. Rissanen and G.G. Langdon. Universal modeling and coding. *IEEE Transaction on Information Theory*, IT-27(1):12–23, January 1981.

[Rog00] W. Rogers. *TREC Mandarin Chinese Corpus.* The Linguistic Data Consortium, Philadelphia, U.S.A, 2000.

[RST96] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[Shk02] D. Shkarin. PPM : one step to practicality. *Data Compression Conference 2002*, pages 202–211, 2002.

[SIG05] SIGHAN. The second international Chinese word segmentation bakeoff. *The Second International Joint Conference on Natural Language Processing (IJCNLP)*, 2005.

[Soy05] Soyouwanna.com. The most widely spoken languages in the world. *http://www.soyouwanna.com/site/toptens/languages/languages4.html*, 2005.

[SSGC96] R. Sproat, C. Shih, W. Gale, and N. Chang. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404, 1996.

[Tea98] W. J. Teahan. *Modelling English text.* University of Waikato, New Zealand, 1998.

[TH01a] W. J. Teahan and D. J. Harper. Combining PPM models using a text mining approach. *Data Compression Conference 2001*, pages 153–162, 2001.

[TH01b] W. J. Teahan and D. J. Harper. Using compression-based language models for text categorization. *Workshop on Language Modeling and Information Retrieval*, 2001.

[TVW97] T. J. Tjalkens, P. A. J. Volf, and F. M. J. Willems. Context-tree weighting method for text generating sources. In *DCC '97: Proceedings of the Conference on Data Compression*, page 472, Washington, DC, USA, 1997. IEEE Computer Society.

[TWMW00] W. J. Teahan, Y. Wen, R.J. McNab, and I.H. Witten. A compression-based algorithm for Chinese word segmentation. *Computational Linguistics*, 26(3):375–393, 2000.

[Vit67] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[VZ98] P. Vines and J. Zobel. Compression techniques for Chinese text. *Software Practice and Experience*, 28(12):1299–1314, 1998.

[WB90] I. H. Witten and T. C. Bell. Source models for natural language text. *International Journal of Man-Machine Studies*, 32(5):545–579, 1990.

[WB91] I. H. Witten and T. C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991.

[WBMT99] I. H. Witten, Z. Bray, M. Mahoui, and W. J. Teahan. Text mining: A new frontier for lossless compression. *Data Compression Conference*, pages 198–207, 1999.

[Wel84] T. A. Welch. A technique for high-performance data compression. *IEEE Computer*, 17(6):8–19, 1984.

[Wil86] G.B. Willam. Early Chinese writing, world archaeology. *Early Writing Systems*, 17(3):420–436, 1986.

[WNC87] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[WST95] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context tree weighting method: basic. properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

[WST96] F. M. J. Willems, Yu.M. Shtarkov, and T. J. Tjalkens. Context weighting for general finite context sources. *IEEE Transactions on Information Theory*, 42(5):1514–1520, 1996.

[WT93] Z. Wu and G. Tseng. Chinese text segmentation for text retrieval: achievements and problems. *Journal of the American Society for Information Science*, 44(9):532–542, 1993.

[WT05] P. Wu and W. J. Teahan. Modelling Chinese for text compression. *Data Compression Conference, 2005*, page 488, 2005.

[WT07] P. Wu and W. J. Teahan. A new PPM variant for Chinese text compression. accepted. *Journal of Natural Language Engineering*, 2007.

[Xue03] N. Xue. Chinese word segmentation as character tagging.

Computational Linguistics and Chinese Language Processing, 8(1):29–48, 2003.

[Zip49] G. K. Zipf. *Human Behavior and the principle of least effort*. MA: Addison-Wesley Press, Cambridge, 1949.

[ZL77] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transaction on Information Theory*, 23(3):337–343, 1977.

[ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transaction on Information Theory*, 24(5):530–536, 1978.