Adaptive Compression-based Approach for Chinese Pinyin Input

Jin Hu Huang and David Powers
School of Informatics and Engineering
Flinders University of South Australia
GPO Box 2100, SA 5001
Australia
{jin.huang,powers}@infoeng.flinders.edu.au

Abstract

This article presents a compression-based adaptive algorithm for Chinese Pinyin input. There are many different input methods for Chinese character text and the phonetic Pinyin input method is the one most commonly used. Compression by Partial Match (PPM) is an adaptive statistical modelling technique that is widely used in the field of text compression. Compression-based approaches are able to build models very efficiently and incrementally. Experiments show that adaptive compression-based approach for Pinyin input outperforms modified Kneser-Ney smoothing method implemented by SRILM language tools (Stolcke, 2002).

1 Introduction

Chinese words comprise ideographic and pictographic characters. Unlike English, these characters can't be entered by keyboard directly. They have to be transliterated from keyboard input based on different input methods. There are two main approaches: phonetic-based input methods such as Pinyin input and structure-based input methods such as WBZX. Pinyin input is the easiest to learn and most widely used. WBZX is more difficult as the user has to remember all the radical parts of each character, but it is faster.

Early products using Pinyin input methods are very slow because of the large number of homonyms in the Chinese language. The user has to choose the correct character after each Pinyin has been entered. The situation in current products such as Microsoft IME for Chinese and Chinese Star has been improved with the progress in language modelling (Goodman, 2001) but users are still not satisfied.

2 Statistical Language Modelling

Statistical language modelling has been successfully applied to Chinese Pinyin input (Gao et al., 2002). The task of statistical language modelling is to determine the probability of a sequence of words.

$$P(w_1 \dots w_i) = P(w_1) * P(w_2|w_1) * \dots * P(w_i|w_1 \dots w_{i-1}) \quad (1)$$

Given the previous i-1 words, it is difficult to compute the conditional probability if i is very large. An n-gram Markov model approximates this probability by assuming that only words relevant to predict are previous n-1 words. The most commonly used is trigram.

$$P(w_i|w_1 \dots w_{i-1}) \approx P(w_i|w_{i-2}w_{i-1}) \quad (2)$$

The key difficulty with using n-gram language models is that of data sparsity. One can never have enough training data to cover all the n-grams. Therefore some mechanism for assigning non-zero probability to novel n-grams is a key issue in statistical language modelling. Smoothing is used to adjust the probabilities and make distributions more uniform. Chen and Goodman (Chen and Goodman, 1999) made a complete comparison of most smoothing techniques and found that the modified Kneser-Ney smoothing(equation 3) outperformed others.

$$p_{KN}(w_i|w^{i-1}_{i-n+1}) = \frac{c(w^{i-1}_{i-n+1}) - D(c(w^{i-1}_{i-n+1}))}{\sum_{w_i} c(w^{i-1}_{i-n+1})} + \gamma(w^{i-1}_{i-n+1}) p_{KN}(w_i|w^{i-1}_{i-n+2}) \quad (3)$$

where

$$D(c) = \begin{cases} 0 & if\ c = 0 \\ D_1 & if\ c = 1 \\ D_2 & if\ c = 2 \\ D_3+ & if\ c \geq 3 \end{cases}$$

$$\gamma(w^{i-1}_{i-n+1}) = \frac{D_1 N_1(w^{i-1}_{i-n+1}\cdot) + D_2 N_2(w^{i-1}_{i-n+1}\cdot) + D_{3+} N_{3+}(w^{i-1}_{i-n+1}\cdot)}{\sum_{w_i} c(w^{i-1}_{i-n+1})} \quad (4)$$

$$
\begin{aligned}
N_1(w_{i-n+1}^{i-1}\cdot) &= |\{w_i : c(w_{i-n+1}^{i-1}w_i) = 1\}| \quad (5)\\
Y &= \frac{n_1}{n_1 + 2n_2}\\
D_1 &= 1 - 2Y\frac{n_2}{n_1}\\
D_2 &= 1 - 3Y\frac{n_3}{n_2}\\
D_{3+} &= 1 - 4Y\frac{n_4}{n_3}
\end{aligned}
\tag{6}
$$

The process of Pinyin input can be formulated as follows.

$$W = \arg\max_W \Pr(W|A) \tag{7}$$

$$W = \arg\max_W \Pr(A|W)\Pr(W) \tag{8}$$

We assume each Chinese character has only one pronunciation in our experiments.

Thus we can use the Viterbi algorithm to find the word sequences to maximize the language model according to Pinyin input.

3 Prediction by Partial Matching

Prediction by Partial Matching (PPM)(Cleary and Witten, 1984; Bell et al., 1990) is a symbolwise compression scheme for adaptive text compression. PPM generates a prediction for each input character based on its preceding characters. The prediction is encoded in form of conditional probability, conditioned on previous context. PPM maintains predictions, computed from the training data, for larger context as well as all shorter con-texts. If PPM cannot predict the character from current context, it uses an escape probability to "escape" another context model, usually of length one shorter than the current context. For novel characters that have never seen before in any length model, the algorithm escapes down to a default "order-1" context model where all possible characters are present.

PPM escape method can be considered as an instance of Jelinek-Mercer smoothing. It is defined recursively as a linear interpolation between the nth-order maximum likelihood and the (n-1)th-order smoothed model. Various methods have been proposed for estimating the escape probability. In the following description of each method, e is the escape probability and $p(\phi)$ is the conditional probability for symbol ϕ , given a context. $c(\phi)$ is the number of times the context was followed by the symbol ϕ . n is the number of tokens that have followed. t is the number of types.

Method A works by allocating a count of one to the escape symbol.

$$e = \frac{1}{n+1} \tag{9}$$

$$p(\phi) = \frac{c(\phi)}{n+1} \tag{10}$$

Method B makes assumption that the first occurrence of a particular symbol in a particular context may be taken as evidence of a novel symbol appearing in the context, and therefore does not contribute towards the estimate of the probability of the symbol which it occurred.

$$e = \frac{t}{n} \tag{11}$$

$$p(\phi) = \frac{c(\phi) - 1}{n} \tag{12}$$

Method C (Moffat, 1990) is similar to Method B, with the distinction that the first observation of a particular symbol in a particular symbol in a particular context also contributes to the probability estimate of the symbol itself. Escape method C is called Witten-Bell smoothing in statistical language modelling. Chen and Goodman (Chen and Goodman, 1999) reported it is competitive on very large training data sets comparing with other smoothing techniques.

$$e = \frac{t}{n+t} \tag{13}$$

$$p(\phi) = \frac{c(\phi)}{n+t} \tag{14}$$

Method D (Howard, 1993) is minor modification to method B. Whenever a novel event occurs, rather than adding one to the symbol, half is added instead.

$$e = \frac{t}{2n} \tag{15}$$

$$p(\phi) = \frac{2c(\phi) - 1}{2n} \tag{16}$$

To illustrate the PPM compression modelling technique, Table 1 shows the model after string *dealornodeal* has been processed. In this illustration the maximum order is 2 and each prediction has a count c and a prediction probability p. The probability is determined from

Order 2				
Prediction			c	p
al	→	o	1	1/2
	→	Esc	1	1/2
de	→	a	2	3/4
	→	Esc	1	1/4
ea	→	l	2	3/4
	→	Esc	1	1/2
lo	→	r	1	1/2
	→	Esc	1	1/2
no	→	d	1	1/2
	→	Esc	1	1/2
od	→	e	1	1/2
	→	Esc	1	1/2
or	→	n	1	1/2
	→	Esc	1	1/2
rn	→	o	1	1/2
	→	Esc	1	1/2

Order 1				
Prediction			c	p
a	→	l	2	3/4
	→	Esc	1	1/4
d	→	e	2	3/4
	→	Esc	1	1/4
e	→	a	2	3/4
	→	Esc	1	1/4
l	→	o	1	1/2
	→	Esc	1	1/2
n	→	o	1	1/2
	→	Esc	1	1/2
o	→	d	1	1/4
	→	r	1	1/4
	→	Esc	2	1/2
r	→	n	1	1/2
	→	Esc	1	1/2

Order 0			
Prediction		c	p
→	a	2	3/24
→	d	2	3/24
→	e	2	3/24
→	l	2	3/24
→	n	1	1/24
→	o	2	3/24
→	r	1	1/24
→	Esc	7	7/24

Order –1			
Prediction		c	p
→	A	1	$1/\|A\|$

Table 1: PPM model after processing the string *dealornodeal*

counts associated with the prediction using escape method D(equation 16). $|A|$ is the size the alphabet which determines the probability for each unseen character.

Suppose the character following *dealornodeal* is *o*. Since the order-2 context is *al* and the upcoming symbol *o* has already seen in this context, the order-2 model is used to encode the symbol. The encoding probability is 1/2. If the next character were *i* instead of *o*, it has not been seen in the current order-2 context (*al*). Then an order-2 escape event is emitted with a probability of 1/2 and the context truncated to *l*. Checking the order-1 model, the upcoming character *i* has not been seen in this context, so an order-1 escape event is emitted with a probability of 1/2 and the context is truncated to the null context, corresponding to the order-0 model. As *i* has not appeared in the string *dealornodeal*, a final level of escape is emitted with a probability of 7/24 and the *i* will be predicted with a probability of 1/256 in the order-–1, assuming that the alphabet size is 256 for ASCII. Thus *i* is encoded with a total probability of $\frac{1}{2} * \frac{1}{2} * \frac{7}{24} * \frac{1}{256}$.

In reality, the alphabet size in the order- –1 model may be reduced by the number of characters in the order-0 model as these characters will never be predicted in the order- –1 context. Thus it can be reduced to 249 in this case. Similarly a character that occurs in the higher-order model will never be encoded in the lower-order models. So it is not necessary to reserve the probability space for the character in the lower-order models. This is called "exclusion", which can greatly improve compression.

Compression Method	Size	Compression Rate
Escape A(order 2)	434228	54.8%
Escape B(order 2)	332278	41.9%
Escape C(order 2)	333791	42.1%
Escape D(order 2)	332829	42.0%
Escape D(order 1)	345841	43.6%
Escape D(order 3)	332932	42.0%
gzip	434220	54.8%
compress	514045	64.8%

Table 2: Compression results for different compression methods

Table 2 shows the compression result for file People Daily (9101) with 792964 Bytes using different compression methods. PPM compression methods are significantly better than practical compression utilities like Unix *gzip* and *compress* except escape method A but they are slower during compression. The compression rates for escape method B and D are both higher than escape method C. Order-2 model (trigram) is slightly better that order-1 and order-3 models for escape method D.

In our experiment we use escape method D to calculate the escape probability as escape method D is slightly better than other escape methods in compressing text although Method B is the best here. Teahan (Teahan et al., 2000) has successfully applied escape method D to segment Chinese text.

4 Experiment and Result

We use 220MB People Daily (91-95) as the training corpus and 58M People Daily (96) and stories download from Internet (400K) as the test corpus.

We used SRILM language tools (Stolcke, 2002) to collect trigram counts and applied modified Kneser-Ney smoothing method to build the language model. Then we used *disambig* to translate Pinyin to Chinese characters.

In PPM model we used the same count data collected by SRILM tools. We chose a trie structure to store the symbol and count. Adaptive PPM model updates the counts during Pinyin input. It is similar to a cache model (Kuhn and De Mori, 1990). We tested both static and adaptive PPM models on test corpus. PPM models run twice faster than SRILM tool *disambig*. It took 20 hours to translate Pinyin (People Daily 96) to character on a Sparc with two CPUs(900Mhz) using SRILM tools. The following Table 3 shows the results in terms of character error rate. People Daily(96) is the same domain as the training corpus. Results obtained testing on People Daily are consistently much better than Stories. Static PPM is a little worse than modified Kneser-Ney smoothing method. Adaptive PPM model testing on large corpus is better than small corpus as it takes time to adapt to the new model.

	People Daily(96)	Stories
modified Kneser-Ney	5.82%	14.48%
Static PPM	6.00%	16.55%
Adaptive PPM	4.98%	14.24%

Table 3: Character Error Rates for Kneser-Ney, Static and Adaptive PPM

5 Conclusion

We have introduced a method for Pinyin input based on an adaptive PPM model. Adaptive PPM model outperforms both static PPM and modified Kneser-Ney smoothing.

References

T.C. Bell, J.G. Cleary, and I.H. Witten. 1990. *Text Compression*. Prentice Hall.

Stanly Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13, 10.

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

Jianfeng Gao, Juashua Goodman, Mingjing Li, and Kai Fu Lee. 2002. Toward a unified approach to statistical language modeling for chinese. *ACM transaction on Asian Language information processing*, 1(1), March.

Joshua Goodman. 2001. A bit of progress in language modeling. Technical Report MSR-TR-2001-72, Microsoft Research.

P.G. Howard. 1993. *The design and analysis of efficient lossless data compression systems*. Ph.D. thesis, Brown University, Providence, Rhode Island.

R. Kuhn and R. De Mori. 1990. A cache-based natural language model for speech reproduction. *IEEE Transac-tion on Pattern Analysis and Machine Intelligence*, 6.

Alistair Moffat. 1990. Implement the ppm data com-pression scheme. *IEEE Transaction on Communications*, 38(11):1917–1921.

A. Stolcke. 2002. Srilm – an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Lan-guage Processing*, volume 2, pages 901–904, Denver.

W.J. Teahan, Yingying Wen, and I.H. Witten R. McNab. 2000. A compression-based algorithm for chinese word segmentation. *Computational Linguistics*, 26(3):375–394, September.