An Empirical Study of Statistical Language Models for Contextual Post-processing of Chinese Script Recognition

Yuan-Xiang Li, Chew Lim Tan
School of Computing, National University of Singapore, Singapore 117543
{lxy, tancl}@comp.nus.edu.sg

Abstract

It is crucial to use statistical language models (LMs) to improve the accuracy of Chinese offline script recognition. In this paper, we investigate the influence of several LMs on the contextual post-processing performance of Chinese script recognition. We first introduce seven LMs, i.e., three conventional LMs (character-based bigram, character-based trigram, word-based bigram), two class-based bigram LMs and two hybrid bigram LMs combining word-based bigrams and class-based bigrams. We then investigate how the LMs’ perplexities are affected by training corpus size, smoothing methods and count cutoffs. Next, we demonstrate the above LMs’ influence on the post-processing performance in terms of recognition accuracy, memory requirement and processing speed. Finally, we give a proposal to select a suitable LM in real recognition tasks.

1. Introduction

Recognizing offline handwritten Chinese characters is still a challenging pattern recognition problem [1-2]. Mainly because of the large character set, complex character shapes, many confusable subsets of characters with only slightly different shapes, and great variations of writing style, it’s difficult to significantly improve the accuracy of Chinese script recognition in an offline handwritten isolated Chinese character recognition system. Statistical language models (LMs) have been successfully used for the contextual post-processing to increase accuracy in the recognition of Chinese scripts [3-6].

In some earlier works, owing to the limitation of the corpus size and the required memory, class-based LMs were widely used in the contextual post-processing of Chinese script recognition. Tung [3] used POS (parts-of-speech) bigram LMs, Chang [4] used bigram LMs based on words clustered by simulated annealing method, Lee [5] used semantically clustered word-based bigram LMs, Wong [6] also used word-class bigram LMs. Class-based LMs have proved effective for training on small corpora and for fast LM adaptation. For large training corpora, word-based LMs are still superior in capturing collocational relations between words [7]. With the rapid advancement of computer technology, it is now feasible to obtain large-scale corpora and to execute a large LM with many parameters.

In the Chinese language, a word consisting of one or more characters is a basic syntax-meaningful unit, but each character in the word also has a definite meaning in itself. Thus, conventional n-gram LMs can be based on either words or characters. In this paper, besides traditional class-based LMs, three conventional n-gram Chinese LMs are used, i.e., character-based bigram, character-based trigram, word-based bigram. On the other hand, in speech recognition systems, class-based LMs have frequently proved to improve the performance when combined with word-based LMs even when a large amount of training corpora is available [8]. So, we will also use a hybrid bigram LM in the post-processing, which combines word-based bigrams and class-based bigrams.

For Chinese script recognition, high accuracy is certainly the most important to be pursued. However, other two aspects, namely memory requirement and computational complexity are also important in real recognition tasks. In this paper, we will investigate the influence of various LMs on the contextual post-processing in terms of recognition accuracy, memory requirement and processing speed. It is our hope that this investigation can facilitate the practitioners to make the intelligent use of LMs in Chinese script recognition tasks.

The rest of this paper is organized as follows: Section 2 first introduces the framework of Chinese script recognition. In Section 3, we first present several Chinese LMs, and then discuss the factors affecting their perplexities. Section 4 demonstrates the influence of different LMs on the post-processing in detail. Finally, the conclusion is given in Section 5.

2. Description of Contextual Post-processing

Let $X=x_1,x_2,...,x_T$ be a sequence of Chinese character images, where $x_i$ is the $i^{th}$ character image, and $T$ is the length of $X$. Let $S=s_1,s_2,...,s_T$ be a sequence of Chinese characters recognized by an isolated Chinese character recognizer (ICCR), in which each output $s_i$ may include top $K$ candidates. By applying the rule of maximal posterior probability, the optimal sentence $O=o_1,o_2,...,o_T$ from $K^T$ possible sentences can be represented as [9]:

$$O = \arg \max_{o_1,o_2,...,o_T} P(o_1,o_2,...,o_T | X).$$

The above formulation is also called Viterbi decoding, which is an efficient way to find the optimal path through a lattice.

Proceedings of the 9th Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)
0-7695-2187-8/04 $20.00 © 2004 IEEE
where \( p(S) \) stands for a statistical LM; \( p(s_i|x_i) \) stands for the confidence of a candidate, which can be estimated by the Logistical Regression Model [9]. The optimal sentence \( O \) can be searched by the well-known Viterbi algorithm.

### 3. Statistical Language Model

#### 3.1. Description of Chinese LMs

In the Chinese language, conventional n-gram LMs can be based on either characters or words. Based on Chinese characters, for \( n=2, 3 \), we have the character-based bigram model \((charBi)\) and the character-based trigram model \((charTri)\), which can be expressed as follows:

\[
p(S) = p(s_1) \prod_{i=2}^{T} p(s_i | s_{i-1})
\]

\[
p(S) = p(s_1) p(s_2 | s_1) \prod_{i=3}^{T} p(s_i | s_{i-1}, s_{i-2})
\]

Considering Chinese words, we use \( S=w_1w_2...w_T \) \((S \text{ contains } T' \text{ words})\) instead of \( S=s_1s_2...s_T \). Based on words, for \( n=2 \), we have the word-based bigram model \((wordBi)\) expressed as follows:

\[
p(S) = p(w_1) \prod_{i=2}^{T} p(w_i | w_{i-1})
\]

For obtaining word classes, the exchange algorithm using the criterion of perplexity improvement was employed [8]. In this paper, we test 500 and 2000 word classes, from which we obtain the class-based bigram models called \( \text{class500} \) and \( \text{class2k} \) respectively.

While class-based LMs generalize better to unseen word sequences, word-based LMs in general have better performance, when enough training corpora is available. It is desirable to retain the advantages of each of these models by combining their word predictions [10]. So, we can construct a hybrid bigram model \((hybridBi)\) that combines \( wordBi \) with \( classBi \) by linear interpolation expressed as follows:

\[
p_c(w_i | w_{i-1}) = \lambda p(w_i | w_{i-1}) + (1-\lambda) p_c(w_i | w_{i-1})
\]

The optimal value of \( \lambda \) can be estimated by optimizing over the held-out data. Interpolating \( wordBi \) with \( class500 \) and \( class2k \), we obtain \( \text{hybrid500} \) and \( \text{hybrid2k} \) respectively.

### 3.2. Perplexity

The most common metric for evaluating the performance of a given LM is the value of its perplexity (PP) [11], which can be computed on a test corpus. PP is defined as follows:

\[
PP = p(M)^{-1/|M|}
\]

where \( M \) is a sequence of the considered language, \( p(M) \) denotes a statistical LM. \( L \) is the length of a test corpus measured in characters for character-based LMs or the length of a test corpus measured in words for word-based LMs. Intuitively, PP can be interpreted as the average number of possible successors of a Chinese character or word. Clearly, the lower the perplexity, the better is the LM in use.

#### 3.3. Factors Affecting Perplexity

For a test corpus, the perplexity of a given LM is affected by the size of training corpus, the smoothing method for unseen n-grams, and count cutoffs.

In our experiment, there are 3763 character types and 78988 word types in the Chinese lexicon. We use four training corpora from the People's Daily, named as Set1 to Set4. Set1, Set2 and Set3 consist of 1993 newspaper, 1993-1994 newspapers and 1993-1995 newspapers respectively. Set4 consists of Set3 and 1996 newspaper excluding November and December, which contains 83.8 million characters (54.4 million words). The texts of November 1996 are used as held-out data. The test corpus is made of the texts containing 2.2 million characters (1.4 million words) from December 1996. People's Daily corpora are very comprehensive and LMs trained by them can be widely applied to different domains.

#### 3.3.1. The Size of Training Corpus

Using the Jelinek-Mercer smoothing method [11], we test PPs with different corpus size for \( charBi \), \( charTri \), \( wordBi \), two class-based bigram models and two hybrid bigram models, as shown in Fig.1.
From Fig.1, we can see that:

- Obviously, \textit{charBi} has the highest PP while \textit{charTri} has the lowest PP. \textit{wordBi} has a little higher PP than \textit{charTri}.
- The PPs of two class-based bigram LMs are lower than that of \textit{charBi}, but higher than that of \textit{wordBi}. Obviously, \textit{class500} has a higher PP than \textit{class2k}.
- Both \textit{hybrid500} and \textit{hybrid2k} have little lower PPs than \textit{wordBi}. It is worth noting that \textit{hybrid500} almost has the same PP as \textit{hybrid2k}, which indicates that more word classes are hardly beneficial for constructing \textit{hybridBi}. Small classes may be enough to construct \textit{hybridBi}.
- With increasing size of training corpus, the PPs of all LMs decrease. Note that the PPs of both \textit{classBi} and \textit{hybridBi} decrease slowly while the PPs of conventional n-gram LMs decrease fast. For small training corpora, \textit{hybridBi} is beneficial to decrease PP. For example, with set1, its PP nearly equals the PP of \textit{charTri}.

3.3.2. Smoothing Method. For n-gram LMs, smoothing technology for sparse data is a central issue. Chen and Goodman [11] investigated the most widely used smoothing methods for addressing English sparse data issues. We test the PPs of the above seven LMs using the following four smoothing methods: Jelinek-Mercer (J-M) smoothing, Witten-Bell (W-B) smoothing, Katz smoothing and Kneser-Ney (K-N) smoothing, as shown in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>J-M</th>
<th>W-B</th>
<th>Katz</th>
<th>K-N</th>
</tr>
</thead>
<tbody>
<tr>
<td>charBi</td>
<td>75.7</td>
<td>75.2</td>
<td>74.9</td>
<td>74.9</td>
</tr>
<tr>
<td>charTri</td>
<td>36.2</td>
<td>34.9</td>
<td>34.6</td>
<td>35.6</td>
</tr>
<tr>
<td>wordBi</td>
<td>39.2</td>
<td>38.4</td>
<td>37.9</td>
<td>37.5</td>
</tr>
<tr>
<td>class500</td>
<td>60.3</td>
<td>58.8</td>
<td>58.8</td>
<td>58.8</td>
</tr>
<tr>
<td>class2k</td>
<td>47.5</td>
<td>46.4</td>
<td>46.3</td>
<td>46.4</td>
</tr>
<tr>
<td>hybrid500</td>
<td>38.7</td>
<td>37.9</td>
<td>37.5</td>
<td>37.2</td>
</tr>
<tr>
<td>hybrid2k</td>
<td>38.5</td>
<td>37.7</td>
<td>37.3</td>
<td>37.0</td>
</tr>
</tbody>
</table>

From Table 1, we can see that different smoothing methods could impact PP to some extent. But the change for a given LM is trivial. For simplicity, J-M smoothing method is adopted in the following statements.

3.3.3. Pruning LM. For large training corpora, count cutoffs (pruning) are often used to restrict the size of the n-gram model constructed (see Section 4.1). With model pruning, all n-grams with fewer than a given number of occurrences in the training corpus are ignored. Using J-M smoothing method, we display the effect of count cutoffs on PP for the above seven LMs in Fig.2.

![Fig.2. Perplexity affected by model pruning](image)

As can be seen in Fig.2, with increasing pruning threshold (PT), PP rises for all other LMs except for \textit{class500}. In comparison with \textit{hybridBi} and \textit{class2k}, the conventional n-gram LMs are rather sensitive to PT. Especially \textit{class500} is fairly robust to PT. Note that the PP of \textit{class2k} is lower than that of \textit{wordBi} when PT>7. For \textit{hybridBi}, its PP is lower than that of \textit{charTri} when PT>5.

4. Comparative Experiments and Results

We conduct our post-processing experiments on a DELL PC (Pentium-IV, CPU 2.4Ghz, 256MB RAM). “THOCR’97 Synthetic and Integrated Chinese Character Recognition System” [12] is used as ICCR. The objects of post-processing are three scripts, i.e. ScriptA, ScriptB and ScriptC, whose recognition accuracies (RAs) without post-processing are 92.32%, 81.58% and 70.84% respectively. Each script consists of about 22,000 characters, covering news, politics, and computer selected from the Internet (the contents are not in set4).

Although RA is naturally very important, memory requirement and computational complexity are also important in real recognition tasks. In this section, for the seven LMs mentioned in Section 3, we first compare their memory requirements and processing speed, then compare their recognition accuracies in detail.

4.1. Comparison of Memory Requirement

Fig.3 demonstrates that the memory requirement varies with PT for \textit{charBi}, \textit{charTri}, \textit{wordBi}, \textit{class500} and \textit{class2k}. Without count cutoffs, the sizes of these five
LMs are 12MB, 53MB, 49MB, 2MB and 16MB respectively. Since hybridBi consists of wordBi and classBi, its size is certainly larger than wordBi. hybrid500 and hybrid2k need 51MB and 65MB respectively.

With increasing PT, except for class500, the other LMs’ sizes decrease exponentially. Especially, pruning the n-grams with one occurrence can greatly decrease the size of a model. For example, the memory space is only 28MB for charTri and 20MB for wordBi in the case.

4.2. Comparison of Processing Speed

The contextual post-processing speed mainly depends on three factors: the complexity of looking up LM parameters, the complexity of searching optimal sentence, and the complexity of constructing a word graph. Apparently, the parameters of charBi are far fewer than those of charTri and wordBi, and the searching space of charBi post-processing is also far smaller than that of charTri and wordBi post-processing. On the other hand, charBi and charTri post-processing do not require the construction of word graph. For classBi, although its parameters are extremely few (see Section 4.1), its post-processing needs the construction of word graph like wordBi post-processing. Intuitively, hybridBi post-processing is more complex than both wordBi post-processing and classBi post-processing.

We adopt the above seven LMs without pruning to obtain the relationship curve between the post-processing time and the number of candidates K for ScriptB, as shown in Fig.4. Noting that the complexity of constructing a word graph rapidly rises with increasing K [13], we have, in practice, only processed the candidate set in which the first candidate’s confidence is less than 0.99.

As can be seen from Fig.4, charBi post-processing is extremely fast and its processing time is almost negligible compared to other six LMs. Since charTri post-processing does not require the construction of word graph, its post-processing is faster and its processing time rises linearly with increasing K, while wordBi post-processing appears very slow and its processing time rises exponentially with increasing K. For classBi, its processing time also rises exponentially with increasing K, although its post-processing is rather fast with small K. Obviously, hybridBi post-processing is a little slower than wordBi post-processing.

Note that both wordBi post-processing and hybridBi post-processing are very slow when K is large. In Fig.4, for K=50, wordBi, hybrid500 and hybrid2k take 38 minutes, 43 minutes and 48 minutes respectively; while charTri, class500 and class2k take 10 minutes, 14 minutes and 17 minutes respectively. However, for K=50, charBi post-processing only needs 23 seconds.

4.3. Comparison of Recognition Accuracy

In this sub-section, using the seven LMs, we show their influence on RA in contextual post-processing.

4.3.1. Post-processing with 10 candidates. Using these LMs without pruning, Table 2 shows the experimental results of seven post-processing methods coded as M1 to M7 with 10 original candidates. The average processing time for M1-M7 is also shown in Table 2. From Table 2, experimental results are characterized by the following:

- Among M1-M5, charBi has the lowest RA, while charTri has the highest RA. wordBi, class500 and class2k have the RAs between charTri and charBi. Certainly, wordBi has a higher RA than classBi. The above results are in accordance with the analysis in Section 3, and confirm that lower PP correlates with a higher accuracy.
• Obviously, hybridBi has a higher RA than wordBi. For ScriptC, hybridBi even outperforms charTri.

| Table 2. Comparing RAs for various processing methods with 10 candidates (%) |
|-------------------|-----------------|-----------------|-----------------|-----------------|-----------------|
|                   | Script A (%) | Script B (%) | Script C (%) | Average (%) | Average time (s) |
| Before post-       | 92.32        | 81.58         | 70.84         | 81.58         | -               |
| processing         |               |               |               |               |                 |
| M1: charBi        | 98.51        | 93.51         | 84.44         | 92.15         | 4s              |
| M2: charTri       | 98.79        | 94.34         | 85.04         | 92.72         | 145s            |
| M3: wordBi        | 98.76        | 93.92         | 84.96         | 92.55         | 82s             |
| M4: class500      | 98.70        | 93.74         | 84.48         | 92.31         | 11s             |
| M5: class2k       | 98.73        | 93.82         | 84.86         | 92.47         | 15s             |
| M6: hybrid500     | 98.77        | 94.08         | 85.04         | 92.63         | 94s             |
| M7: hybrid2k      | 98.78        | 94.11         | 85.10         | 92.66         | 99s             |

Table 3. Comparing RAs for various processing methods with suitable \(K\) candidates (%)

<table>
<thead>
<tr>
<th></th>
<th>Script A (%)</th>
<th>Script B (%)</th>
<th>Script C (%)</th>
<th>Average (%)</th>
<th>Average time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M8: charBi</td>
<td>98.83</td>
<td>95.49</td>
<td>90.08</td>
<td>94.80</td>
<td>31s</td>
</tr>
<tr>
<td>M9: charTri</td>
<td>99.26</td>
<td>96.69</td>
<td>93.35</td>
<td>96.43</td>
<td>708s</td>
</tr>
<tr>
<td>M10: wordBi</td>
<td>99.22</td>
<td>96.54</td>
<td>93.94</td>
<td>96.57</td>
<td>13217s</td>
</tr>
<tr>
<td>M11: class500</td>
<td>99.15</td>
<td>96.13</td>
<td>92.77</td>
<td>96.02</td>
<td>10028s</td>
</tr>
<tr>
<td>M12: class2k</td>
<td>99.16</td>
<td>96.44</td>
<td>93.60</td>
<td>96.40</td>
<td>10429s</td>
</tr>
<tr>
<td>M13: hybrid500</td>
<td>99.24</td>
<td>96.69</td>
<td>93.95</td>
<td>96.63</td>
<td>14009s</td>
</tr>
<tr>
<td>M14: hybrid2k</td>
<td>99.25</td>
<td>96.71</td>
<td>94.05</td>
<td>96.67</td>
<td>14475s</td>
</tr>
</tbody>
</table>

4.3.3. Model pruning. In Section 3.3.3, we discussed the influence of count cutoffs on the perplexity. Here, taking example for ScriptB, we show its RA affected by count cutoffs for M1-M7 in Fig.5(a) and M8-M14 in Fig.5(b).

Fig. 5. Accuracy affected by model pruning

From Fig.5, we can see that:
• With increasing PT, RA decreases for charTri and wordBi; for charBi, its RA decreases very slowly compared to charTri and wordBi.
• With increasing PT, both classBi and hybridBi show their stable performance, although their model sizes reduce greatly.
• With increasing PT, charTri still has a higher RA than all other LMs when K=10; however, for K=50, not only hybridBi outperforms charTri when PT>1, but class2k outperforms charTri a little when PT>2.

4.4. Discussion

According to the above experimental results, we can make an appropriate decision in choosing a suitable LM for constructing a practical contextual post-processing system when a script is recognized. Which LM to be employed really depends on the available memory and computational resources as well as the requirement of response time in real recognition tasks.

It is quite clear that if an application has to be run on a platform with only very limited memory and computational resources, then class500 is the choice to build a practical post-processor. If high recognition accuracy is the main concern of the application, then hybrid500 or charTri can be used. If processing speed is strictly required in some applications, charBi is a practicable LM. If enough memory space is available, charTri is such a good LM that it can obtain high recognition accuracy while being efficient in terms of processing speed, especially when processing a poorly recognized script.

It is noticeable that model pruning can greatly reduce the size of a LM, while the model’s capability of improving accuracy only decreases a little.

Since increasing the number of candidates often reduces the processing speed, we should try to improve the effectiveness of candidate set when a script is poorly recognized, that is to allow the correct character to be captured in a limited number of candidates.

5. Conclusion

In this paper, several statistical language models have been investigated in the contextual post-processing of Chinese script recognition. We first show three factors affecting their perplexities, and then demonstrate their influence on the contextual post-processing performance in terms of recognition accuracy, memory requirement and processing speed. We give the proposal in choosing a suitable language model according to the requirement of a practical recognition system.

Acknowledgement This research is partly supported by the Agency for Science, Technology and Research (Grant No. R252-000-123-305) in Singapore.

References