Improving Language Modeling for (Off-line) Chinese Character Recognition

by

Hung Kei-yuen

A Thesis Submitted to

the Hong Kong Polytechnic University

in Fulfillment of the Requirements

for the Degree of

Master of Philosophy

Department of Computing

The Hong Kong Polytechnic University

March 2002

Abstract

We analyze the error characteristics of a Chinese character recognizer and developed two approaches to improve Chinese character recognition system.

We first develop a non-contiguous context dependent language model as a post processing module. The model makes use of far away context to predict the interested character. The model is only as good as the traditional bigram model in terms of accuracy.

Secondly, we developed a method to detect errors in language model. The method employs pattern recognition technique. It combines both dictionary and statistical features to predict whether a block of character is correct or contains error. This detection scheme as demonstrated in our experiment is effective. The performance is 80%, 91% and 75% of precision, recall and skip ratio respectively.

Acknowledgement

I would like to thank my supervisor Dr. Robert Luk, for his continuous help during my study. Dr. Luk devoted much time to discuss with me, point me to the right direction when I am confused, give me advice when I have problems. Without his help, I would not be able to complete this thesis.

In addition, I would like to thank Dr. Qin Lu, Dr. K.F. Wong, Dr. D.Y. Yeung and Dr. Korris F.L. Chung for their valuable and constructive comments on my thesis.

Last, but not least, I would like to thank my family for their endless love and unlimited support. This thesis is dedicated to them.

Table of Contents

Table of Figures

Table of Tables

Chapter 1

Introduction

1.1 The need for Chinese character recognition

Computers are invented in the western countries. The internal code and input devices are designed to facilitate the representation and input of English text. In order to enable processing of Chinese text in the existing computer systems, several coding standards have been designed such as Big5, Guobiao (GB) and Unicode to encode Chinese characters. For the input of Chinese characters, numerous input methods such as changjei have been designed for users to input Chinese characters with the standard input device i.e. the keyboard. However these keystroke based input methods are too complicated for the average computer users to master. As a result, new input devices are invented to make the input of Chinese characters more convenient and natural. The most successful product is the pen-based input device, which allows users to input characters as if they are writing on a paper. Unfortunately, this on-line means of input method cannot cope with all the Chinese text input problems. For instance, it is time consuming to "write" pages of Chinese text into the computer using pen-based devices. Therefore, an off-line means to enter bulks of text from printed material or handwritten text is required. This approach works by scanning the text into the computer as an image and the image is analyzed to extract the characters. There are many potential applications with this input approach. For example, it can be applied in a post office to automatically recognize addresses on the mail

envelopes and sort mails accordingly. Another important application is to automatically extract the information, which is manually filled in, from paper forms.

1.2 Difficulties of offline Chinese character recognition

Both of the above mentioned input strategies require sophisticated pattern recognition technology. Offline handwritten Chinese character recognition is one of the most challenging fields in pattern recognition because of the visual complexity and the large number of characters in the alphabet. (over 50,000, from which about 5,000 are commonly used). The problem is generally easier for on-line recognition than the off-line counterpart because stroke sequence is available and timing information of each stroke is provided. Without the timing information, spatially overlapping characters may cause segmentation problems. In addition, on-line recognition constrains users to write inside a predefined area which automatically acts as the boundary box of a character, whereas in the case of off-line recognition, this boundary information needs to be extracted by the recognizer. The extraction process is easier for printed character text because the characters have approximately the same size and the separation between characters is well defined. Handwritten characters, on the other hand, may have different size and style; the boundary between characters is not that obvious. In addition, Chinese characters can be components to other characters. Thus finding boundaries of characters are non-trivial.

1.3 Language model

Due to the above-mentioned difficulties, it is almost impossible to obtain high

recognition accuracy by analyzing the individual character bitmap alone. The contemporary offline Chinese character recognition systems have accuracy around 75% [1][2][3]. This level of accuracy is still not good enough for real world applications such as mail address sorting and form processing. To increase the recognition rate, the nearby context can be utilized to correct any mis-recognized characters. In fact, the same strategy is often applied by human beings when reading a blurred text or listening to a piece of speech in a noisy environment.

The contextual information is described by a language model, which is either incorporated into the recognizer or used as a standalone post-processing module. The application of language model is not limited to offline handwritten character recognition; many other domains such as speech recognition, machine translation and spelling correction can be benefited by incorporating a language model.

1.4 Objective

The aim of this research is to improve the performance, in term of accuracy, of language modeling for (offline) Chinese character recognition.

1.5 Scope of study

Figure 1.1 shows the architecture of a character recognition system. Our works will focus on the post-processing part of the system. We tried two approaches to improve the post-processing of a character recognition system. The first one is to use an alternative non-contiguous language model to replace the traditional model (bigram model) and the second one is to add an additional error detection module to detect the error from the output of the traditional language model

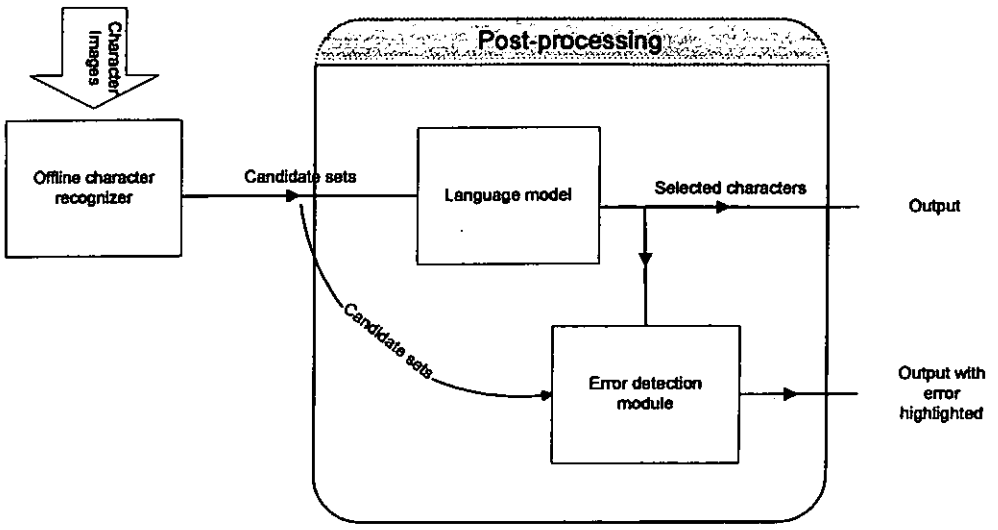

Figure 1.1 The architecture of the proposed system

1.6 Outline of the thesis

The thesis is organized into seven chapters. We give a brief introduction to Chinese character recognition in Chapter 1. This is followed by a literature review of language modeling in Chapter 2. We studied different approaches for language modeling and contrast their strengths and weaknesses. We claim that the knowledge of error characteristic of a character recognizer can help us design a better language model. Therefore, in Chapter 3, we run experiments to determine and analyze the error occurrence characteristics of a Chinese character recognizer. In Chapter 4, we determine the context window size for Chinese language. Based on the results in chapters 3 and 4, we proposed a non-contiguous context dependent language model in Chapter 5. We run experiments based on the proposed model, but the obtained performance is comparable to our baseline bigram model only. In Chapter 6, we develop a method to detect language model errors and in Chapter 7, we give a conclusion of this thesis and suggest some future works.

Chapter 2

Literature Review

The language models that utilize contextual information to improve the accuracy of character recognizers can mainly be classified either as dictionary based, statistical-based or hybrid approach.

2.1 The dictionary approach

This approach uses a dictionary to correct the mistakes made by the recognizer. The sequence of characters O recognized are compared with entries in the dictionary and the entry E, which is most similar to O, is selected as the intended word. The distance (i.e. a measure of dissimilarity) between O and an entry E is the number of edit operations to transform O to E, and this distance can be computed by dynamic programming [4].

Dictionary based approach has the advantage that existing linguistic knowledge can be incorporated into the system with small parameter space and low computational cost. Also, some of the unrecognized characters can be recovered by contextual approximate word matching.

The major problem faced by the dictionary approach is unknown words. It is not possible to obtain a complete Chinese dictionary because, (1) domain dependency, some strings may be words in some domain while not in others, (2) proper names, there are countless names for people, countries, mountains etc, (3) authors have the flexibility to create new words by some combination of characters or words

(compounds and acronyms).

Another problem is the abundance of single-character word in Chinese. About 30% of Chinese words consist of only one character. This makes the use of approximate string matching ineffective. Dictionary approach also faces the word segmentation problem which will be described in section 2.3.1.

Pak-Kwong Wong and Chorkin Chan [5] used dictionary approach as the baseline language model to improve an off-line hand-written Chinese character recognizer [1]. Basically, it employs the maximum matching algorithm to match words with the entries in the dictionary. An improvement of 6.8% in recognition rate is achieved.

2.2 The statistical approach

This approach is based on statistics gathered from a training corpus. The statistics may include the individual character occurrence frequencies, the character co-occurrence frequencies, the dependency probabilities etc. The language model uses these statistics to select the most plausible candidate sequence among those proposed by the recognizer.

The most widely used statistical language model is the n-gram model. It works under the Markovian assumption that the occurrence probability of a symbol is only strongly dependent on its previous n-1 symbols. In this model, each character isolated by the recognizer produces several candidates at a particular position.

Let $I = i_1, i_2, ..., i_n$ represents the character images of a sentence with n characters where i_1 is the first character and i_n is the last character in the sentence. Each character image is recognized as m candidate characters. For instance, the first character image i_1 has a set of candidates $c_{11}, c_{12}, ..., c_{1m}$. A sentence hypothesis S

$= c_1, c_2, \ldots, c_n$ is formed by selecting one candidate at each position. As a result, there are m^n sentence hypotheses. The task of the language model is to determine a sentence hypothesis S_{max} that has the maximum likelihood among all sentence hypotheses.

$$S_{\max} = \arg\max_{S} P(S \mid I)$$

The probability that i_j will be recognized as c_j is

$$P(c_j \mid i_j)$$

and the probability that the character at position j is c_j provided that its preceding sequence of character is $c_1, c_2, \ldots c_{j-1}$ is given by

$$P(c_j \mid c_{j-1}c_{j-2}\ldots c_1)$$

The probability that a sentence hypothesis S will be obtained given an image I is then the multiplation of the above conditional probabilities from c_1 to c_n. i.e.

$$P(S \mid I) = P(c_1 \mid i_1) \cdot P(c_2 \mid c_1)P(c_2 \mid i_2) \cdot P(c_3 \mid c_2c_1)P(c_3 \mid i_3) \cdots P(c_n \mid c_{n-1}c_{n-2}\ldots c_1)P(c_n \mid i_n)$$

or ,

$$P(S \mid I) = \prod_{j=1}^{n} P(c_j \mid c_{j-1}c_{j-2}\ldots c_1) \prod_{k=1}^{n} P(c_k \mid i_k)$$

It is infeasible to directly estimate and store the contextual probability for an arbitrary n. Many researches restrict the character sequence to two or three. The resulting model is a bigram and trigram language model respectively. For a trigram model, the representation can be simplified as,

$$P(S \mid I) = \prod_{j=1}^{n} P(c_j \mid c_{j-1}c_{j-2}) \prod_{k=1}^{n} P(c_k \mid i_k)$$

For Indo-European languages, the word-bigram language model is used in speech recognition [6] and handwriting recognition [7]. Various ways to improve language models were reported. First, the model has been extended with longer dependencies (e.g. trigram) [8] and using non-contiguous dependencies, like

trigger pairs [9] or long distance *n*-gram language models [10]. For better probability estimation, the model was extended to work with (hidden) word classes [10][11]. A more error-driven approach is the use of hybrid language models, in which some detection mechanism (e.g. perplexity measures [12] or topic detection [13]) selects or combines with a more appropriate language model.

For Asian languages (e.g. Chinese, Japanese and Korean) represented by ideographic characters, language models are widely used in computer entry because these Asian languages have a large set of characters (in thousands) that the conventional keyboard is not designed for. Apart from using speech and handwriting recognition for computer entry, language models for Asian languages can be used for sentence-based keyboard input [14], as well as detecting improper writing (e.g. dialect-specific words or expressions).

Unlike Indo-European languages, words in these Asian languages are not delimited by space and conventional approximate string matching techniques [4][15] in handwriting recognition are seldom used in Asian language models. Instead, a widely used and reported Asian language model is the character-bigram language model [16][17] because it (1) achieves high recognition accuracy (around 90-96%) (2) is easy to estimate model parameters (3) can be processed quickly and (4) is relatively easy to implement.

Improvement of these language models for Indo-European languages can be applied to Asian languages but words need to be identified. For Asian languages, the model was integrated with syntactic rules [18]. Class based language model [19] was also examined but the classes are based on semantically related words. A new approach [20] is reported using segments expressed by prefix and suffix trees but the comparison is based on perplexity measures, which may not correlate well with recognition improvement [21].

2.3 The hybrid approach

The advantage of the dictionary approach is the recent development in approximate string matching that allows correction of mis-identifications and segmentation errors, and the adaptation of edit operations and distances to a particular recognizer for better performance. Furthermore, the dictionary approach incorporates the language aspect. For words that exist in the dictionary, correction can be made with high confidence. On the other hand, the *n*-gram approach can select the appropriate single-character words (which are difficult for the dictionary approach) and detect new words that do not exist in the dictionary based on finding the most likely sequence of characters. This provides a global optimization criterion which the dictionary approach lacks.

Since the two approaches complement each other, a hybrid approach combining both the dictionary and *n*-gram approaches is promising. The idea is to use word instead of character as the basic unit to build an *n*-gram model and we call this a word-based *n*-gram approach.

2.3.1 Word segmentation

The hybrid approach is not without its deficiency. In particular, to use words as the basic units introduce the word segmentation problem. Unlike English and other Western languages, which has explicit word boundary, there exists ambiguities to convert a sequence of character candidates to a sequence of words. In general, there are two types of ambiguities; grouping ambiguities and overlapping ambiguities.

The former occurs when there are two or more different segmentations of a phrase

that are all meaningful but some words identified by one segmentation enclose the other word in a different segmentation, for example the sentence 香港人口多 can be segmented as

香港人 | 口多 or 香港 | 人 | 口多,

where 香港人 in the first segmentation enclose 香港 and 人 in the second segmentation.

The latter occurs when there exits two or more different segmentations of a phrase that are all meaningful but some words identified by different segmentations overlap. For example, the above sentence can be segmented as

香港 | 人口 | 多 or 香港人 | 口多.

In this case, the consequence is more serious as wrong segmentation will produce a different meaning.

Word segmentation has long been a research topic in the area of Chinese language processing. Many recent papers [5][20][22][23][24] addressed the issue and proposed some solutions. All of the solutions were to incorporate some statistical information to correct the mis-segmented sequence generated by the dictionary alone.

2.3.2 Word class

Another problem of the hybrid approach is that, the number of parameters is too large. There are over 80,000 words in a typical Chinese lexicon. The parameters for the model are the conditional probabilities between words. The number of probabilities is astronomical even for a bigram or trigram model. Therefore, most word-based *n*-gram model will group the words into classes first using their syntactic or semantic information. There are a great many grouping strategies. A common method is to group the words according to their part of speech [19][2].

Another approach is to use both semantic information and statistical data [5][19] as the basis for grouping. Currently, a word-class based bigram or trigram model can boost the recognition rate by about 10%.

2.4 Summary

The table in Table 2.1 shows the summarized recognition rates of current Chinese character recognition system and the improvement that a particular language model can make to the recognition rate. The figure is only an approximation, which may be varied if the testing conditions have changed.

	Printed character	Handwritten character
Recognition rate		
Without language model	85%	75%[1][2][3]
Improvement in recognition rate		
Dictionary model		7% [5]
Character-based bigram	5%	
Word-based bigram POS	3%	
Word-based trigram POS	3%	10%[5]

Table 2.1 A table summarizing the recognition rate of recognizer and the improvement that language model can achieve

Chapter 3

Error Characteristics

In this section, we attempt to observe and analyze the error occurrence characteristics of a Chinese character recognizer. We believe that, with the knowledge of error pattern, we can develop a better language model. The recognizer we used can only support printed character recognition. If handwritten characters are presented to the recognizer, the resulting text will contain too many errors to be analyzed meaningfully. For that reason, we only analyze the error characteristics of printed character recognition.

3.1 The experiment

The first step is to compile a Chinese text for the experiment. Ten news articles are randomly selected from a local online newspaper. From each of these articles, we randomly extract one paragraph to constitute the text. The Chinese text is then printed with seven different fonts. Some of these fonts are supported by the recognizer while some are not. To simulate the effect of handwritten characters, we have included xingshu font type, which is a cursive Chinese font. Next, the printed text is scanned into images and presented to the recognizer. Finally, the string distance between the recognized text and the original text is computed using dynamic programming [25] to determine any errors occurred. The following shows an example of the algorithm.

		香	港	特	別	行	正	又	區
	0	1	2	3	4	5	6	7	8
香	1	0	1	2	3	4	5	6	7
港	2	1	2	3	4	5	6	7	8
特	3	2	3	2	3	4	5	6	7
別	4	3	4	3	2	3	4	5	6
行	5	4	5	4	3	2	3	4	5
政	6	5	6	5	4	3	4	5	6
區	7	6	7	6	5	4	5	6	5
。	8	7	8	7	6	5	6	7	6

Figure 3.1 An example of computing string distance using dynamic programming.

Figure 3.1 is a matrix of costs computed by the algorithm. The first column shows the original string x while the first row shows the recognized string y. The distance between x and y is defined in terms of elementary edit operations (see below) which are required in order to transform x into y. Different edit operations are used to handle the errors detected in the string recognition process.. There are three different types of edit operations:

1. Substitution of a symbol in x by a symbol in y. This is represented by a diagonal step in the matrix. A special case of this is the substitution of a symbol by itself, which corresponds to a correctly recognized symbol.
2. Insertion of a symbol in y. This is represented by a horizontal transition in the matrix.
3. Deletion of a symbol in x. This is represented by a vertical transition in the matrix.

We assigned a cost of 1 for a substitution, insertion or deletion operation and 0 for a correct. Then the distance between x and y is obtained by summing up the costs of all the elementary operations of the sequence with minimum total costs among all sequences which transform x into y. This is represented by the path of line segments shown in Figure 3.1.

3.1.1 Edit operation and error

We randomly examine some sample outputs from dynamic programming. We observed that the edit operations identified to transform the original string to the recognized string correspond well with the errors that occur in the recognized string. That is, a substitution, deletion or insertion operation identified indicates that there is respectively a substitution, deletion or insertion error in the recognized string.

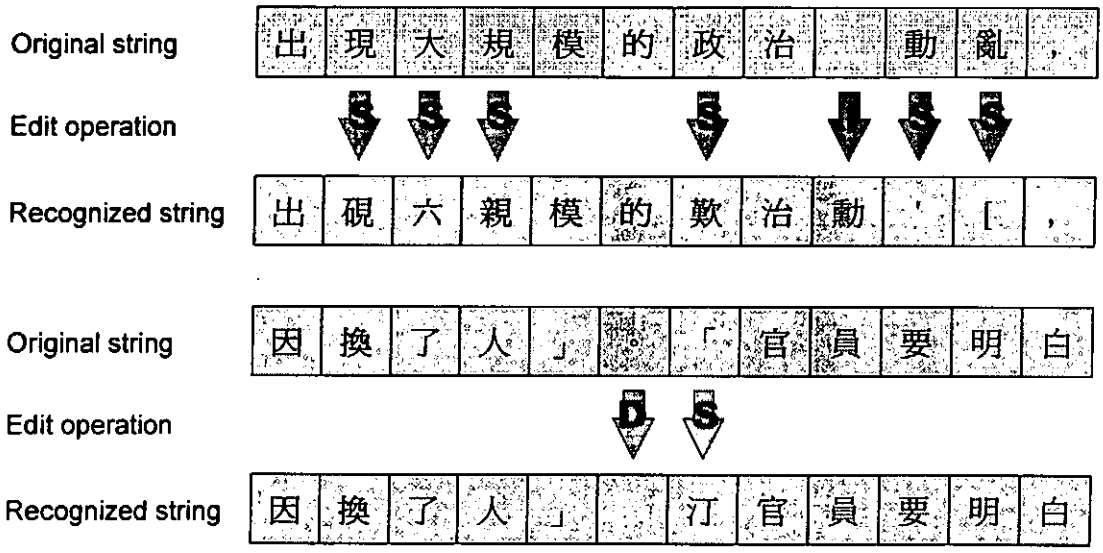

Figure 3.2 Sample results from dynamic programming

Figure 3.2 shows two sample results from dynamic programming. Arrows with a character S, I and D indicate respectively that a substitution, insertion and deletion is needed to transform the original string to the recognized string. We observed that all substitution and deletion errors can be identified correctly. Although the insertion error identified may be wrong in position, the error is still within the contiguous error block.

As we can interpret the edit operations as errors, we can analyze the error characteristics or patterns of a recognizer.

3.2 Result and observation

3.2.1 The aggregate statistics

The number of correctly recognized symbols (characters or punctuation marks) and the number of errors with their corresponding types (insertion, substitution and deletion) determined for the seven fonts are tabulated in Table 3.1:

	Hei 黑	Kai 楷	Li 隸	Ming 明	Song 宋	Weibei 魏碑	Xingshu 行
Correct	1075	1403	965	1400	1132	776	717
Insertion	1	0	4	0	5	12	169
Substitution	422	95	533	97	366	722	781
Deletion	1	0	0	1	0	0	0
Accuracy (%)	71.76	93.66	64.25	93.46	75.32	51.39	43.01

Table 3.1 A table summarizing the recognition results for different fonts

The total number of symbols (including characters and punctuation marks) in the experiment text is 1498 and the accuracy percentages are calculated by dividing the number of correct symbols over this number times one hundred. The results reflect that most of the recognition errors are substitution errors. Deletion errors seldom occur in printed character recognition. By examining the accuracy percentages and the occurrence of insertion errors, we can group the fonts into three categories as follow:

Accuracy	Insertion errors	Fonts
Accurate (> 90%)	None	Kai and Ming
Medium (between 60% & 90%)	Few (< 10)	Hei, Li and Song
Inaccurate (< 60%)	More (>10)	Weibei and Xing

Table 3.2 Fonts classified into three categories according to recognition rate

3.2.2 The distribution of errors

The error pattern whether the errors occurred in burst or randomly is determined by compiling the occurrence frequency over a statistics on the burst error length. The sequences of edit operations, i.e. errors, are investigated to obtain the statistics of burst error length. The result is tabulated below:

Burst length	Hei 黑	Kai 楷	Li 隸	Ming 明	Song 宋	Weibei 魏碑	Xingshu 行
0	1075	1403	965	1400	1132	776	717
1	280	94	281	95	266	296	233
2	75	1	125	3	69	150	156
3	20	0	35	0	14	68	110
4	7	0	9	0	4	32	56
5	5	0	6	0	0	18	31
6	0	0	3	0	0	10	14
7	0	0	3	0	0	1	3
8	0	0	1	0	0	0	5
9	0	0	0	0	0	0	6
10	0	0	0	0	0	0	4
11	0	0	0	0	0	0	2

Table 3.3 A table showing the occurrence frequency of different burst error length in various fonts

The frequency versus burst length curve is plotted for each font type and is shown in Figure 3.3.

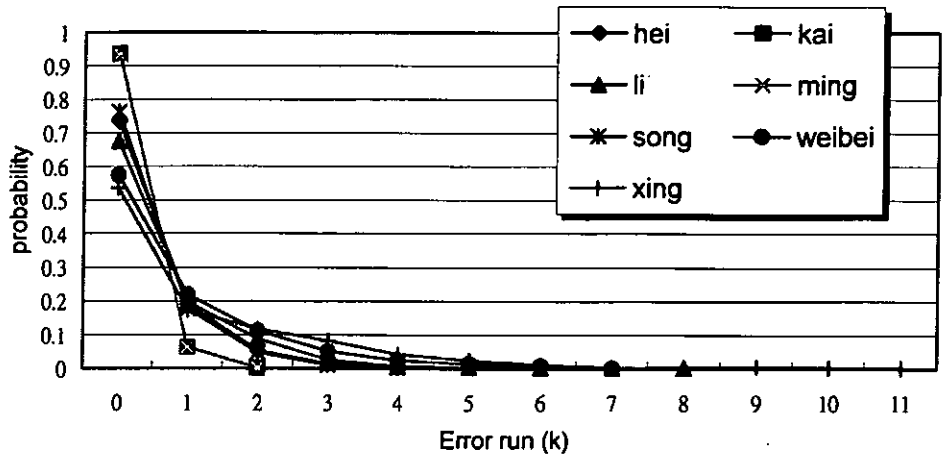

Figure 3.3 The histogram of burst error length.

The shape of the curves resemble that of an exponential curve, to make it more clear, we plot the curves again with the y-axis in log scale as in Figure 3.4.

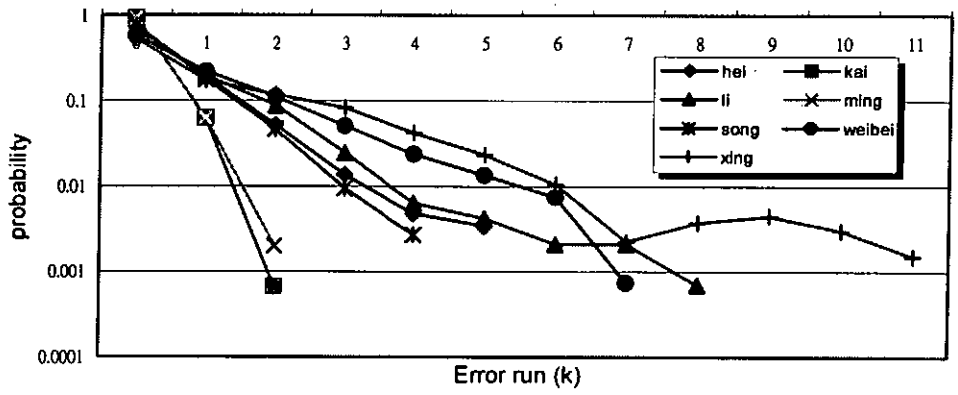

Figure 3.4 The statistics of burst error in log scale

The curves are approximate straight lines except for some variations in high value of *k*. The fluctuations are due to inadequate occurrence counts at these points (refer to Table 3.3). We need to set a threshold to cut the low occurrence cases. This value is arbitrary and from Table 3.3, the occurrence count for burst length longer than six drops to a low value, we decided that a threshold of six is appropriate. Figure 3.5 shows the curves after eliminating low occurrence cases.

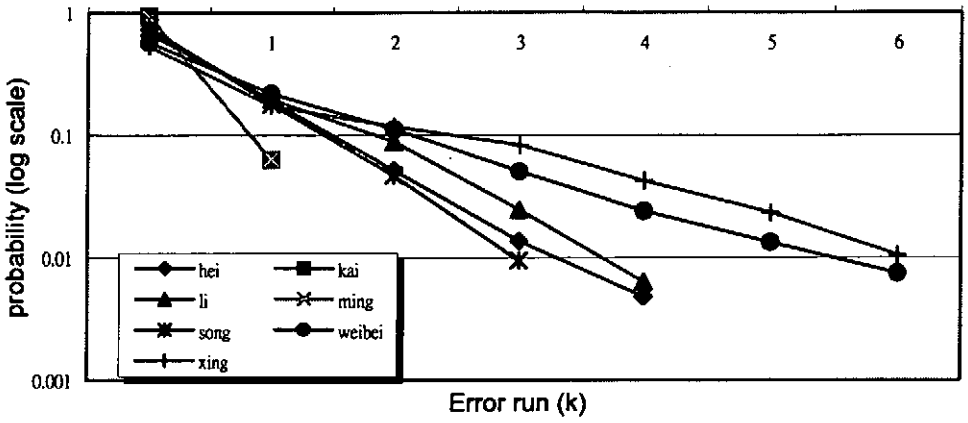

Figure 3.5 The burst error curves after cutting low occurrence cases

3.3 The error model

If we consider the errors occur like a sequence of binomial trials in which the probability of an error is β and a correct is α, where $\beta = 1 - \alpha$. The recognizer can then be viewed as a process that generate a character, either correct or wrong, at a time. To model the burst errors, we let the state of the process at trial k be the number of uninterrupted errors that have been generated at this point (the error run). Figure 3.6 and Figure 3.7 shows the state transition diagram and the transition probability of the Markov chain respectively.

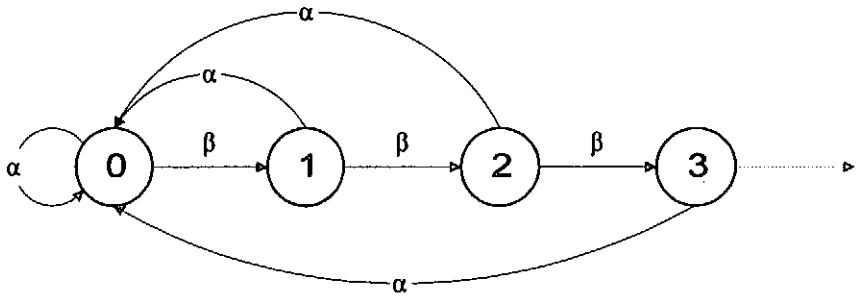

Figure 3.6 The state transition diagram

$$\begin{vmatrix} \alpha & \beta & 0 & 0 & \cdots \\ \alpha & 0 & \beta & 0 & \\ \alpha & 0 & 0 & \beta & \\ \alpha & 0 & 0 & 0 & \\ \vdots & & & & \end{vmatrix}$$

Figure 3.7 The transition probability matrix

The state 0 can be reached in one transition from any state while the state k+1 can only be reached from state k in one transition. It can be proved that the probability of the model being at a state k is $P(S_k) = \alpha\beta^k$.

3.4 Test the fitness of the error model

In this section, we tested the fitness of the observed state occupancy distribution obtained in section 3.2.2 with our proposed error model in section 3.3. We used the mean of the observed distribution to determine the expected line and applied chi-square test (with 0.01 significant level) to verify the goodness of fit. Since there is only one parameter α ($\beta = 1-\alpha$), we can use the mean only to completely specify the distribution.

Let m be the mean of the observed distribution and n be the number of state, then

$$m = \sum_{n=0}^{\infty} n(1-\beta)\beta^n$$

It can be proved that when n tends to infinity,

$$\beta = \frac{m}{1+m}$$

The results are tabulated in Table 3.4.

Font type	m	β	χ^2	v	$\chi^2_{.99}$	
Hei 黑	0.35431	0.73838	0.82628143	4	13.3	Accepted
Kai 楷	0.06275	0.94095	1.42485	1	6.63	Accepted
Li 隸	0.47059	0.68	12.218	5	15.1	Accepted
Ming 明	0.06342	0.94036	1.49182	1	6.63	Accepted
Song 宋	0.30034	0.76903	1.17805	3	11.3	Accepted
Weibei 魏碑	0.79793	0.5562	10.544	6	16.8	Accepted
Xing 行	1.00075	0.49981	54.1444	6	16.8	Rejected

Table 3.4 Chi-square test results (v is the degree of freedom)

Figure 3.8 shows the graph of the curve fitting result for Hei type font. From the figure, it can be observed that the expected line fit very well with the observed distribution. From the result, it can be concluded that the recognition errors for printed character text can be modeled using a Markov chain with state occupancy probability at a state k equal to $\alpha\beta^k$.

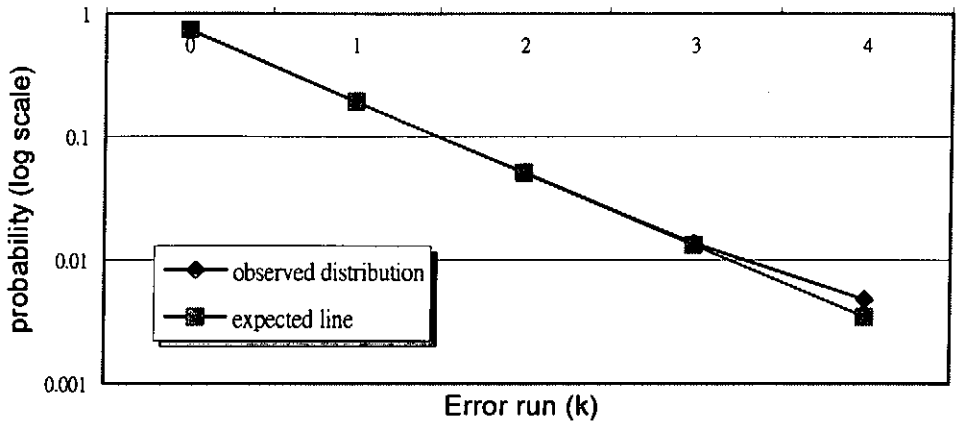

Figure 3.8 Curve fitting using distribution mean for Hei type font

3.5 Summary

From the experiment, we observed that for texts with medium and inaccurate recognized results (76% or below), significant amount of errors occur in burst.

Therefore, it is not justify using only the nearby context to correct recognition errors because the characters within this context may itself be incorrectly recognized. In view of this, we claim that the traditional language model approaches, dictionary and *n*-gram, which utilized only the nearby contextual information, will not be effective for these cases.

We propose to use a non-contiguous and context-dependent language model to solve the problem. To be non-contiguous means that the associations may not necessarily be formed by adjacent characters and the language model considers the characters in the context as independent units. This avoids the propagation of errors, which result in incorrect prediction.

Chapter 4

Context Window Size

The language model we proposed is context-dependent. An important parameter for this model is the context window size, which is measured by the number of linguistic units (characters in our case) that the context includes before and after the interested unit. For example, the following concordance,

香	港	理	工	大	學	電	子	計	算	學

has 5 characters before and 5 characters after the interested character 學. The size of the context is usually defined either *a prior* or derived after a few experimentation. If the size is too small, important information would be missed and the performance of the task at hand would suffer. On the other hand, if the size is too large, unnecessary computational costs would be incurred and unwanted information (which may be regarded as noise) might interfere and degrade performance. Therefore, a tradeoff exists between performance and context size.

4.1 Previous work

Lucassen and Mercer [26] discovered a method to determine the size of the context for English pronunciation. The main idea is to measure the degree of association between the interested character (phoneme in this case) and the following or preceding character (English character or orthographic symbol) at a

fixed relative distance. As the distance increases, the degree of association decreases (Figure 4.1). Up to some distance, the degree of association would be too small or decreasing too slowly to justify the size of the context.

The measure of association used in [26] is the mutual information:

$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$$

which is the mean of the information content of the two random variables X and Y. The measure is symmetric so that $MI(X,Y) = MI(Y,X)$. Lucassen and Mercer extended the mutual information to variables X and Y separated by a fixed distance d so that:

$$MI(X,Y,d) = \sum_{x \in X} \sum_{y \in Y} p(x,y,d) \log_2 \frac{p(x,y,d)}{p(x)p(y)}$$

where $p(x,y,d)$ is the probability that x occurs before y at a fixed distance d.

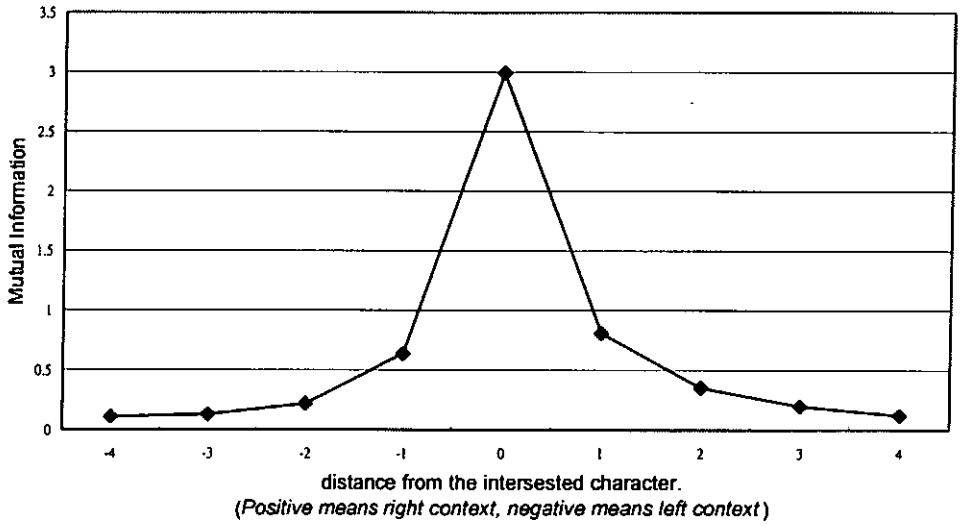

Figure 4.1 Mutual information measured for determining the context window size for English spelling to phoneme conversion (after Lucassen and Mercer [26]).

In computational linguistics, measurements of the degree of association are used to find collocations [27] and word compounds [28]. Instead of mutual information, Church [29] proposed an association score, which is very similar to mutual information. The association score $A(x, y)$ tests whether the joint occurrence of x

and y is much larger than if x and y occurred by chance. The comparison of joint and by chance occurrence is expressed as a ratio, which is made symmetric over the origin, when the joint occurrence is the same as the by-chance occurrence, by taking the logarithm of the ratio.

The association score differs from mutual information in two noticeable ways. First, the association score $A(x, y)$ effectively measures the information content between two values instead of taking the average of the information content between two random variables. Second, $A(x, y)$ is asymmetric, i.e. $A(x, y) \neq A(y, x)$. This suggested that the association score has a direction component.

Since the association score is a widely used measure in computational linguistic and natural language processing, it is not known whether context window sizes should be determined by mutual information or association score or other measures. Here, we address this problem by developing the basic model to determine the context window size. Based on the model, we show that the association score is the preferred measure and we demonstrated how to obtain the context window size in practice.

4.2 Association score approach

A context window, $w[I-L..I+R]$, is a substring of a text $w[]$, where I is the unit (eg. A character) of interest and L and R are its left and right context. The size of the context window is the number of units in the window, which is $L+R+1$. Figure 4.2 shows an example of a context window of size 11 (i.e. $L+R+1 = 5+5+1$). Within the context window, some units (characters) in the context have high association scores with the interested unit (i.e. I). These are useful to many applications as the presence of those units could predict the presence (co-occurring) of the interested

unit (i.e. *l*). Therefore, the context window size should be large enough to include all the significant associations' scores.

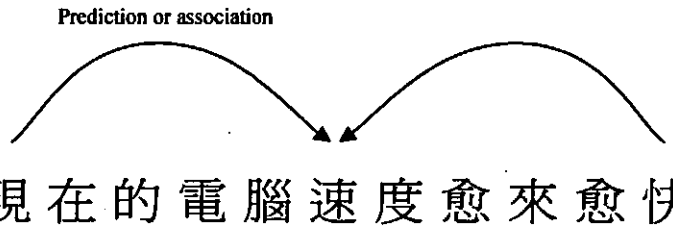

Figure 4.2 An example of a context window of size 11. The window should contain most of the prediction or association so that the application could use the information in the context to make appropriate predictions.

4.2.1 Prediction versus association

In many applications, it is the prediction ability rather than the degree of association that is important. The conditional probability $p(x|y, d)$ can be used as a statistical prediction measure because it expresses the likelihood that x occurs given that y is present at relative distance d. Similar to the association score, the prediction of x by y can be compared with the case if y has no prediction ability (i.e. where x and y occur independently). Thus, a measure of the prediction ability is

$$\frac{p(x \mid y,d)}{p(x)}.$$

To make the ratio symmetric over the origin, we define the prediction score as:

$$P(x \mid y,d) = \log_2 \frac{p(x \mid y,d)}{p(x)}$$

It turns out that the prediction score is equals to the association score:

$$A(x,y,d) = \log_2 \frac{p(x,y,d)}{p(x)p(y)}$$
$$= \log_2 \frac{p(x,y,d)}{p(x)p(y,d)}$$
$$= \log_2 \frac{p(x \mid y,d)}{p(x)} = P(x \mid y,d)$$

if the probability that y occurs is the same for all the different relative distance d. We regard prediction and association as synonymous unless it is not clear in the context.

Obviously, the prediction score is asymmetric similar to the association score but both scores are symmetric over the relative distance d (i.e. $P(x,y,d) = P(y,x,-d)$). For example,

$$P(香 \mid 港,d) = \log_2 \frac{p(香,港,d)}{p(香)p(港)}$$
$$= \log_2 \frac{p(港,香,-d)}{p(香)p(港)}$$
$$= P(港 \mid 香,-d)$$

Therefore, the prediction score can be computed for only one of the two contexts.

4.2.2 Mutual Information versus Association Score

Mutual information simply takes the average of the association scores over the different units that can occur in the context. This may result in discarding many significant associations. For instance, if the association score is symmetrically distributed with a mutual information value of zero (i.e. on average the association are spurious), then half of the significant association scores (i.e. larger than 0) would be discarded. Thus, mutual information is not an appropriate measure to decide the context window size because the mutual information value would drop steeper than desired. Figure 4.3 shows the mutual information at various distance

d for a Chinese corpus [30]. The mutual information values do not decrease significantly when $d > 6$.

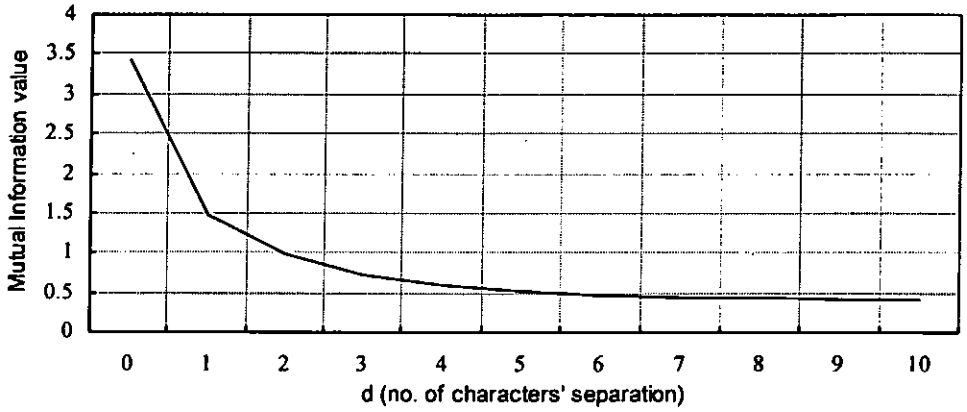

Figure 4.3 Mutual information over different distance values.

The alternative to mutual information is to use the association score $A(x,y)$. Unlike mutual information in which the degree of association is already summarized to a numerical value for each distance d, there is a set of association scores $\{A(x,y,d)\}$ for every distance d and these association scores must be summarized to a numerical value to represent the amount of significant associations at distance d. Since the context window size includes all the significant associations, the maximum association score (i.e. an extremum) value for a particular distance d can be considered as a representative (or upper bound) of the set of associations, i.e.:

$$\max_{x \in X, y \in Y} \{A(x,y,d)\}$$

Figure 4.4 shows the maximum association score for different values of d. The maximum association score is much larger than mutual information and it is unclear whether the maximum association score would decrease any further for larger values of d.

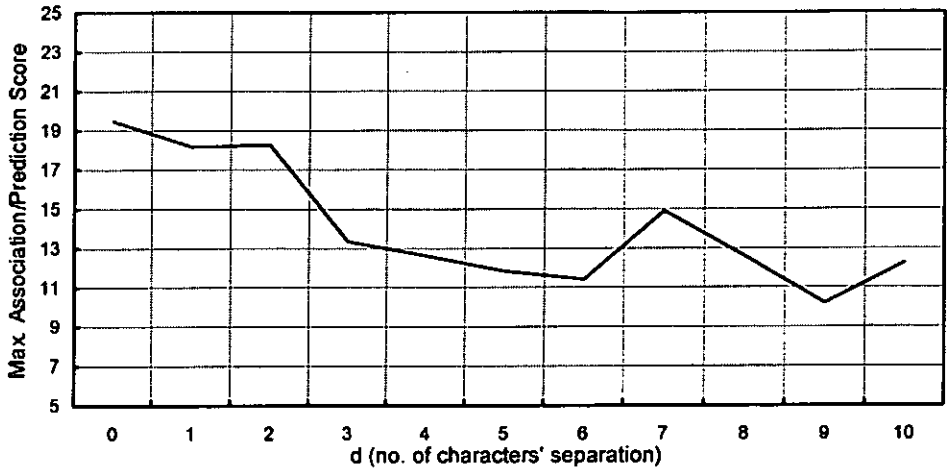

Figure 4.4 Variation of the maximum association score with distance d.

Unfortunately, the maximum association score usually does not vary smoothly with the distance *d* because any individual spurious but strong association would distort the curve. To visualize whether there are spurious but strong associations, the distribution of the ranked association scores is plotted as in Figure 4.5. The largest association score has the top rank (i.e. 1) and second largest has the second etc. According to Figure 4.5, great changes of the association score values and the ranks occur in the top 25 ranked association scores. For the rest of the ranks, the association score values decrease gradually. Instead of using the maximum association score (i.e. the top rank), we use the N% quartile such that the spurious but strong associations in the top ranks are neglected.

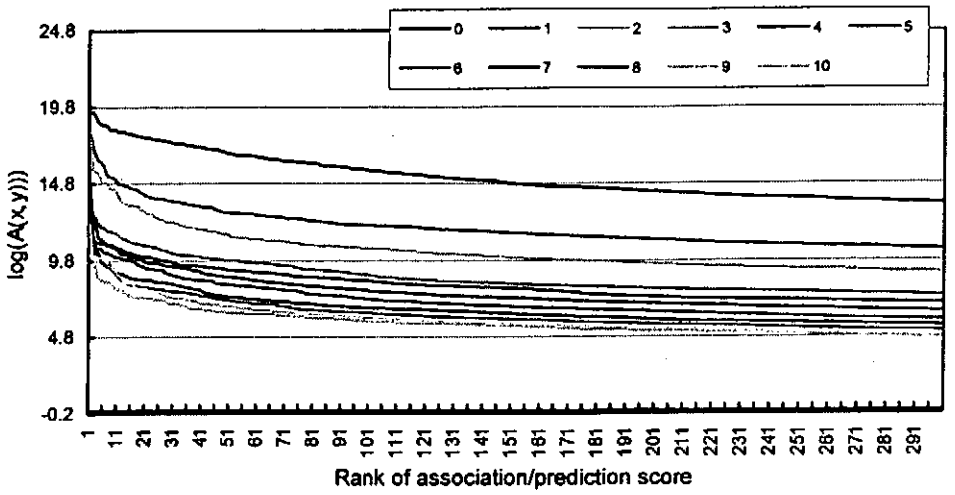

Figure 4.5 The distribution of the ranked association score values.

Figure 4.6 shows the variation of the association score at the top 99.99% quartile. As expected, the association score varies more smoothly with the distance *d*. In addition, the position (i.e. 9) where the amount of association score decreases insignificantly occurred further than that (i.e. 6) for the mutual information measure (Figure 4.3).

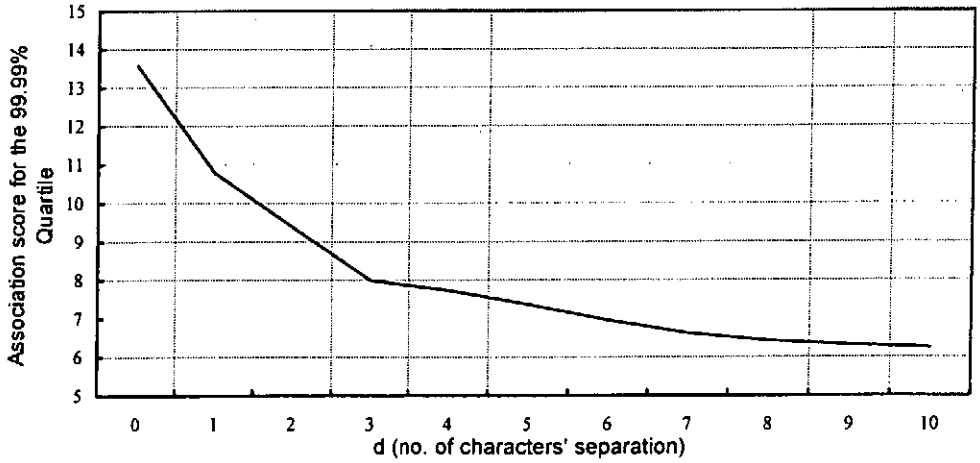

Figure 4.6 Variation of the association score at the 99.99% quartile with the distance d.

4.2.3 Spurious associations

It is well known that spurious strong associations occur when $f(x,y,d)$ is low,

where $f(x,y,d)$ is the frequency of x co-occurring with y at a distance d. Previous work has used a threshold of 5 [29] or 8 [28] to limit the amount of spurious associations. To examine the effect of this threshold, we have plotted the association score values for the 99.99% quartile at various distances d in Figure 4.7. Clearly, as the threshold is lowered, the association scores at large distance (i.e. between 5 and 10) increases. For threshold values larger than 1, the position where the decrease in association score values are more or less the same (at approximately 9). This confirms setting the threshold value to 5 or 10 are acceptable.

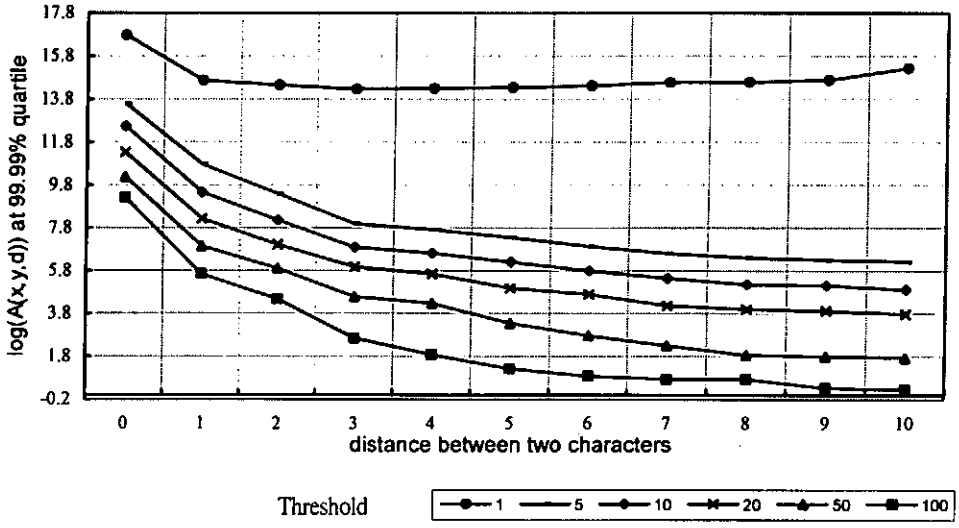

Figure 4.7 Effect of varying the threshold to limit spurious associations on the association scores at the 99.99% quartile.

4.2.4 Text-type dependence

The results in the previous figures were obtained using the PH corpus [30], which is a collection of newswire stories published by Xinhua news agency. However, it is not known whether the variation of the association score with the distance d is similar for text written in different geographical regions, since there are regional variations in writing, influenced by the local language communities.

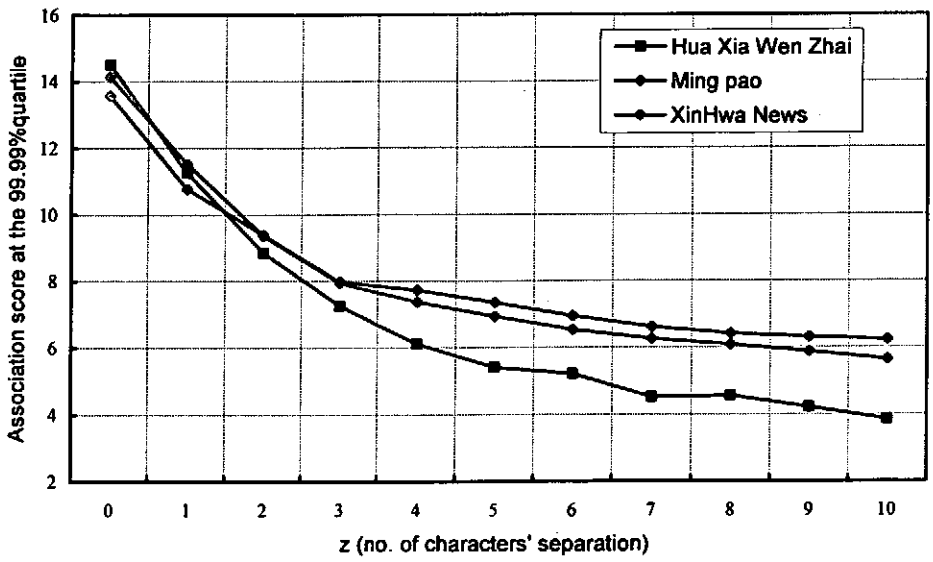

Figure 4.8 Association score variation with distance *d* for three different text collections from Hua Xia Wen Zhai, Ming pao and Xinhua news agency.

Figure 4.8 shows the variation of the association score at the 99.99% quartile for text in Ming Pao (a Hong Kong newspaper), Xinhua news agency (an official newspaper) and Hua Xia Wen Zhai (an on-line magazine). The association score variations for the three collections of text were very similar. Surprisingly, Ming Pao text was more similar to text from Xinhua news agency than Hua xia wen zhai. Above the value of 9, the association scores for all the text collections do not make much significant drop and we conclude that a context window of 9 is adequate.

4.2.5 Different units

The figures obtained before were character-based (i.e. the interested unit is a single character and the length of the contexts is also in terms of characters). However, many applications require a word-based context, where a word is usually a string. For example, The target in word sense determination is a word.

For word-based context, there is a problem to define the relative distance *d*.

Usually, there is a consensus that counting the distance begins from the interested character and ends by the word in the context. For example, the counting starts from the character on the right of the interested word 香港 and ends at the word 區 in the context.

<香港>特別行政 區

However, the distance *d* could be in terms of the number of characters or words. In this example, *d* could be 4 characters or 2 words. Here, we examine *d* in terms of characters only since (1) the number of words depends on the available word list and (2) we can compare the result with the character-based context.

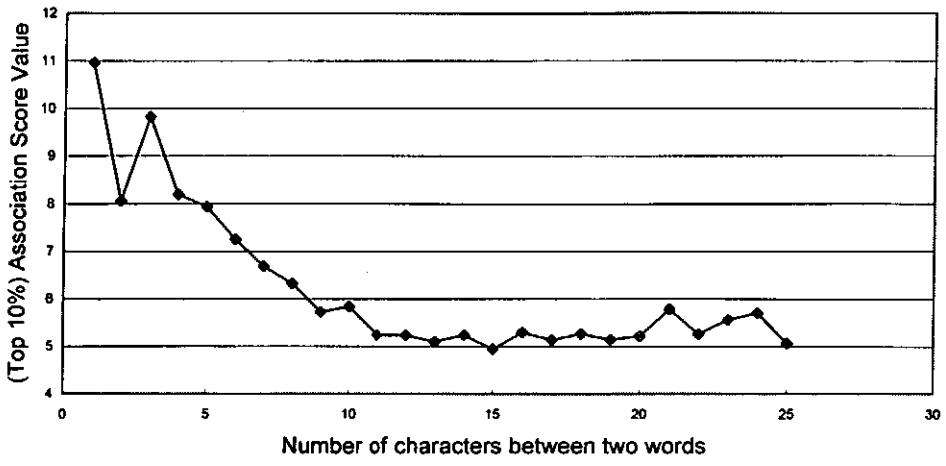

Figure 4.9 The top 10% association score value for the different distances *d* between two words, measured in terms of the number of characters. When $d > 9$, there is no long-term decline of the association score.

Figure 4.9 shows the variation of the association score value at the top 10% rank for the different distances *d* between two words, measured in terms of number of characters. The association scores were measured from the PH corpus, which was manually segmented. There is a general trend of association score decline as the distance increases. When $d > 9$, the decline of the association scores stopped. This context window size agrees with the context window size defined by the association score using character-based context.

Figure 4.10 shows the mutual information values for different distances between two words, measured by the number of characters. Again, there is a general trend of decline in mutual information value with increasing values of distance. However, the estimation of the mutual information value becomes less reliable as distance d is large because there are fewer samples to estimate.

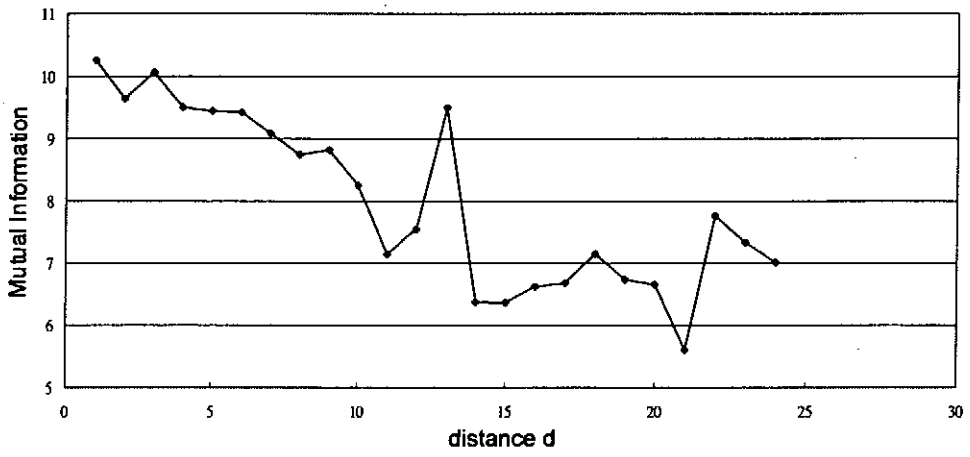

Figure 4.10 Mutual information value for different distances between two words, measured in terms of the number of characters.

4.3 Nonparametric statistic approach

An alternative approach to the association score approach is to use a nonparametric statistics that summarizes the associations between units at a fixed relative distance d. Although mutual information can be considered as a candidate, it was found to be unsuitable for this task because it summarizes the association scores by taking the average.

For categorical data, Lambda statistic L_B developed by Goodman and Kruskal [31] is a suitable nonparametric statistic to measure the associations between two random variables. The statistic $L_B(X|Y)$ is asymmetric because it measures the ability of the random variable Y predicting the other random variable X. The

statistic ranges between 0 and 1 and it can be considered as the percentage of prediction error reduced when Y is known. To be precise, the association measure λ, which estimates L_B, is:

$$\lambda(X \mid Y) = \frac{P(error) - P(error \mid Y)}{P(error)}$$

where *P(error)* is the prediction error probability for X and *P(error|Y)* is the conditional probability of the prediction error for X given Y is known. If *λ(X|Y)* is large, then the prediction error with and without Y differs significantly and therefore Y is associated with X.

The statistic L_B is obtained by computing:

$$L_B = \frac{\sum_{y \in Y} \max_{x \in X} \{n_{x,y}\} - \max_{x \in X} \left\{ \sum_{y \in Y} n_{x,y} \right\}}{\sum_{x \in X} \sum_{y \in Y} n_{x,y} - \max_{x \in X} \left\{ \sum_{y \in Y} n_{x,y} \right\}}$$

where $n_{x,y}$ is the number of times that x and y co-occurred. For a large sample, L_B is normally distributed and its variance can be determined and used as an indicator of the reliability of reducing errors at the specified amount.

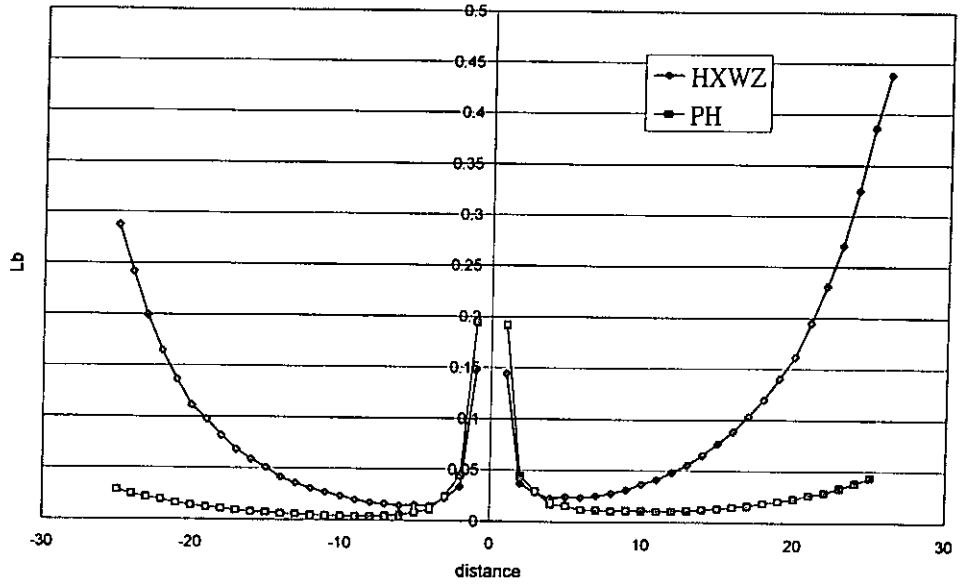

Figure 4.11 Estimated lambda statistics of the PH corpus and the HXWZ on-line magazine.

Figure 4.11 shows the lambda statistics estimated from the PH corpus and the on-line magazine Hua Xia Wen Zhai (HXWZ) for character-based contexts. Since the statistics is asymmetric over the relative distance *d*, statistics for both the left and right contexts are estimated. Negative and positive distances indicate the interested character is on the left and right respectively, of the candidate character. The highest statistics value obtained is just under 0.2, which means that knowing the character at the specified distance can reduce the error of predicting the candidate character by (just below) 20%. Notice that characters in the right context appear to have slightly higher prediction ability than those in the left context.

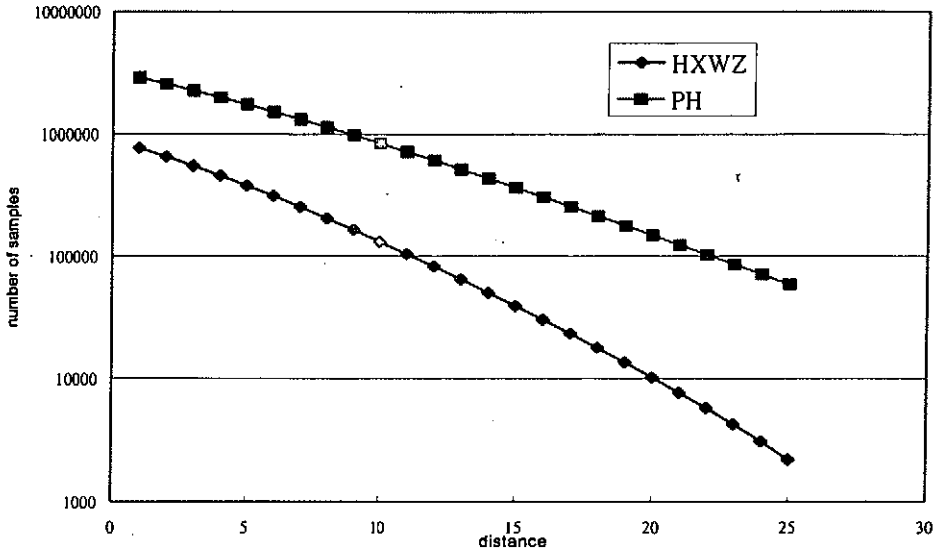

Figure 4.12 The number of samples for different relative distances.

The lambda statistics in Figure 4.11 for both the PH corpus and HXWZ magazine are very similar except that the statistics increase when the magnitude of the relative distance is larger than 10 for the PH corpus and 5 for HXWZ magazine. We plot the number of samples against the relative distance in Figure 4.12. We observe from the figure that the number of samples drops as the relative distance increases. We believe that this account for the abnormal sharp increase at both end

of the chart in Figure 4.11.

As the distance increases, the number of samples decreases and the validity of the lambda statistics estimate decreases. A rough estimate of the number of samples for valid results can be found as follows. There are about 5,000 different characters in the text collection. Therefore, there are 25 million different combinations of prediction of one character by another. Assuming that only 1% of the character combination would be collected in the sample, then the number of samples should be at least 250,000. This implies that valid results are obtained (Figure 4.12) when the magnitude of the relative distances is below 17 for the PH corpus and 7 for the HXWZ magazine.

4.4 Summary

Context windows are important in a variety of natural language processing and analysis. If the size is too small, important information would be missed. On the other hand, if the size is too large, computational cost and interference would be high. Context window size is determined by observing whether there are any strong associations between the interested character/word with another character/word in the nearby context. We argued that mutual information is not a suitable association measure because of its averaging effect. Instead association scores, by Church [29], are used. The association score was found to be related to the prediction ability of the character/word in context over the interested character/word. Since there are spurious strong associations even after discarding associations with low support (frequency < 5), the top N% quartile association score is used as a representative. The result of determining the context window size by the association score was found to be not significantly dependent on the

text types. A window size of 9 (for one direction) characters was found to be large enough for most associations between characters, and between words.

An alternative to the association score is to use the (nonparametric) lambda statistics, which measures the reduction of error in the prediction of the characters/words, which are at some fixed distance from the interested character/word. The lambda statistics aggregate the predictions from individual character/word prediction, not by averaging. It has a more stable and smooth change in reducing prediction error than mutual information. For large distances (i.e. > 10) between the character in context and the interested character, the number of samples is too small to draw valid conclusions. We measured the lambda statistics for both the PH corpus and the HXWZ on-line magazine. We found that the reduction of error became almost insignificant when the window size is about 4. This is substantially different from the window size determined using association score (i.e. 9). The reason is that the association score obtains only one representative instead of aggregating all the significant association score like Lambda statistics. Thus, the association score is suitable to contextual models that do not exhaustively use all the association score for analysis or processing. For example, the discovery of collocations is interested if there exist one or more significant associations in the context. The discovery is not interested in all the associations in the context. The lambda statistics is an aggregate of the associations. Therefore, it is more suitable to contextual models that make use of all associations. For example, hidden Markov models use all the *n*-gram probabilities to define the optimal path. Since all the *n*-gram probabilities are used in order to determine the optimal path, the Lambda statistics give a better measure of the longest history of the *n*-gram probabilities.

Chapter 5

Non-Contiguous Model

In this chapter, we implement a character based non-contiguous context dependent language model. It predicts a character in position *P* from the characters inside its context window. As determined in the last chapter, the window size for each context is 9. Therefore the context window refers to those characters that lie within 9 characters on the left and 9 character on the right of *P*. The predictions are based on data gathered from PH corpus. The associations between two characters with a number of characters apart are stored. It is non-contiguous because the associations may not necessarily be formed by adjacent characters. This contrasts with the traditional dictionary approach and *n*-gram approach, which depend on a continuous block of characters to correct errors.

5.1 Why non-contiguous?

In Chapter 3, we obtained the error characteristics of a recognizer. The results reflected that for accurately recognized texts, errors appear quite randomly; while for the partial-accurately or inaccurately recognized texts, significant amount of errors occur in burst or close to each other. The burst errors deeply decrease the performance of the traditional language models, which use contiguous contextual information. For instance, the *n*-gram language model depends on the previous *n* characters to predict the current one. However, if the current character is an error, then there is a significant chance that its previous characters are also erroneous in

the case of partial-accurately or inaccurately recognized texts. Thus, a wrong prediction would be made from the wrong context leading to further decrease in recognition rate.

5.2 The architecture

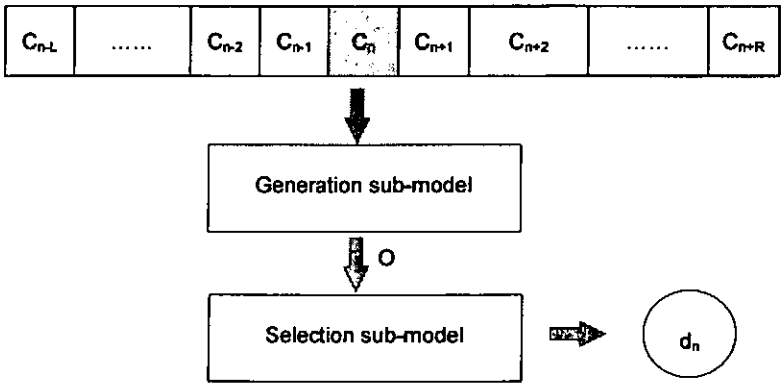

Figure 5.1 The architecture of the proposed model

The model works using a sliding window approach (See Figure 5.1). The symbol $C_i, n-L \le i \le n+R$ is a set of candidate characters at position i. The aim of the model is to pick the right candidate character from the candidate set C_n.

The model consists of two parts. The first part, namely the generation sub-model, retrieves all associations between the characters in the candidate set C_n and the characters in the context candidate sets. Each character in C_n will have a set of associations supporting it; we denote the set as O_j, where j run from 1 to the total number of characters in the candidate set. All O_j are combined to form the superset O_n. The second part, referred to as the selection model, selects the output character d_n from the set O_n.

5.2.1 Generation model

This part of system generates a set of candidate characters for each character

within the context window C_i, where $n-L \le j \le n+R$. In actual recognition system, the candidates are provided by the recognizer. In our case, we do simulation without using an actual recognizer, thus we need to include candidate generation in this part.

Let $f(C, d) = \{X_i\}$ be the set of associations with C, which are d characters from C_n in the context. Positive value of d denotes that C is on the right of C_n and negative value of d means that C is on the left of C_n.

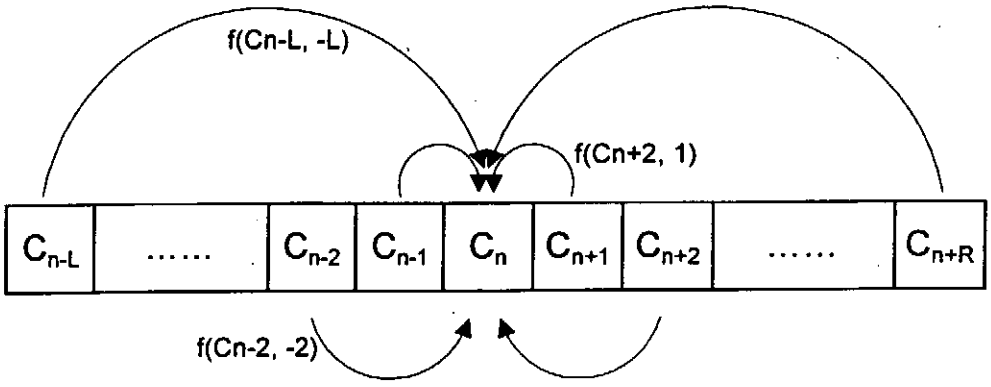

Figure 5.2 Prediction of the current character by the context

Refer to Figure 5.2, entire set of associations are defined as follows:

$$O = \{X \mid X \in f(C_i, i-n), n-L < i < n+R\},$$

where n is position of the unit of interest and L and R are its left and right context size.

5.2.2 Selection Model

Based on the similarity or distance selection model, we can pick one association from the extrema function:

$$x = \arg\max_{X \in O} \{similarity(X, C_i) \mid n-L < i < n+R, i \neq n\}$$

and the output character d_n can be extracted from the association x.

The similarity function could be related to the generation model (e.g. $A(x,y)$) or

has some probabilistic sense (e.g. Naive Baysan model) or has some vector-space interpretation. However, such simplistic model does not take into account of the fact that the output character could be predicted by more than one character in the left and right context. The similarity function must be able to combine the various predictions by different characters within the same left and right contexts. For a probabilistic model, Dempster-Shafer theory [32] of combining evidence could be used.

5.3 Feasibility

A potential problem of the proposed model is the high demand for storage space. The model makes prediction of an interested character using every character inside its context window. That means the model needs to store the association for any two-character combination of every position in the context window. Consider the Big5 character set C, which has about 13,000 characters. There would be $13{,}000^2$ items in the set $P = \{a, b \mid a \in C, b \in C\}$, which contains all the combinations of any two characters. If the context window size is set at 9, 9 association scores would need to be stored for each item in the set P. Assume that a computer use 4 bytes to store one floating point number. The total storage requirement would be $13{,}000^2 * 9 * 4$ bytes, or about 5.7 gigabytes. Although the price of a hard disk is very low nowaday (eg. 10 gigabytes disk costs under one thousand Hong Kong dollars), an application that requires several gigabyte space is still unreasonable.

Our proposed model seems to be infeasible due to heavy storage requirement. In fact, the storage demand is not that high if we consider that not every character has association with all the characters in C within the defined window size. In

practice, the storage requirement depends on the size and the number of distinct characters of the training corpus. Data obtained from the PH corpus [29], which has over three million characters, occupies about 50 megabytes of storage. Storage requirement may increase if we use a larger training corpus to gather the data, But it can be reduced. As stated in Chapter four, for reliable prediction, we can exclude associations which appear fewer than a threshold count T i.e. *count(a, b, dist)* < T. Much storage could be saved if we use a higher value of T. In this way, however, we may risk the reduction in coverage of the prediction data. We performed an experiment to determine an appropriate value of T.

5.3.1 The threshold count

In determining the value of T, there is a tradeoff between coverage and storage. In our model, the candidate, which has the highest association with the context, will be selected as the correct one. If this association pair exists in a particular context window, we say that this window is covered. As we increase the threshold T, more association pairs will be eliminated and eventually, some context windows will have their highest association pair removed and thus become uncovered.

In the experiment, the PH corpus was run through with different value of T and the number of context windows covered was counted. The counting is converted into fraction and plotted against different threshold values T in Figure 5.3. For each run, the storage requirements were also recorded. The fraction of storage saved, as compared with the storage requirement when $T=0$ is plotted on the same graph.

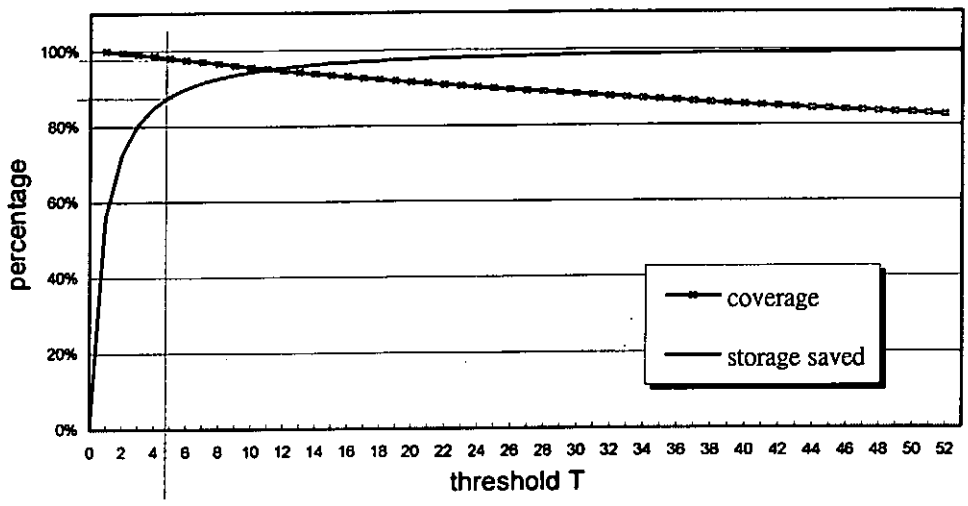

Figure 5.3 The coverage and storage saving versus threshold curve

The curve representing "storage saved" increases sharply as T moves from 0 to 4. That means a dramatic save in storage requirement can be achieved even we use a small value of T. We have arbitrarily selected 5 as the threshold value in determining the context window size in Chapter 4 and claim that this is an appropriate value. From Figure 5.3, when T is set at 5, about 85% of storage can be cut and the coverage remains around 98%.

5.4 Experiment

In this section, we examine the performance of the proposed non-contiguous model and compare it with the baseline. Throughout the experiment, PH corpus is used as the training corpus and Ming Pao news mentioned in section 4.2.4 is used as the testing corpus. Ten candidate characters are randomly generated for each character in the testing corpus and the language model is applied to select the most plausible characters from the candidate sets. We use random candidates because it gives highest entropy, not bias for a particular model and thus serve better for comparison. The accuracy of the model is measured in terms of the number of correctly selected characters over the total number of characters in the

corpus.

5.4.1 Baseline model

We use the bigram model as the baseline for comparison. We chose bigram model because it is simple, widely used and gives good performance. Bi-grams statistics were extracted from the PH corpus. We recorded a correct recognition rate of 78%. This result deviated a lot from that reported in the literature (96%). However, the high accuracy reported in the literature was based on closed tests.

5.4.2 Non-contiguous model

In Section 5.2, we have introduced the architecture of the non-contiguous model. We now go to the implementation detail of the model.

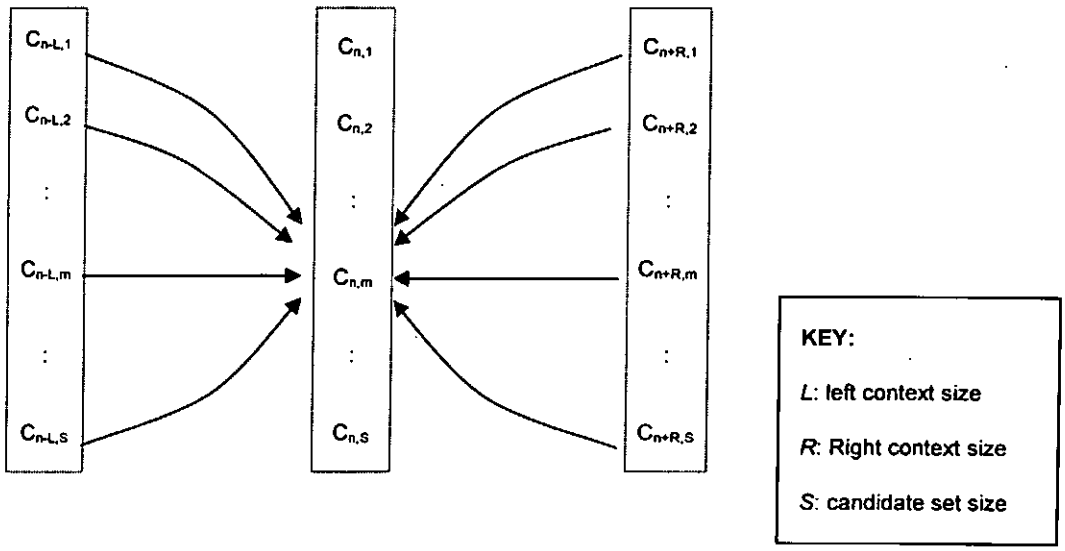

Figure 5.4 The non-contiguous model

Suppose the model is going to determine the nth character C_n from the candidate set $\{C_{n,1} \ldots C_{n,S}\}$. The first step is to generate evidences for each of the S characters in the candidate set. The evidence is measured in terms of the

association probabilities with the characters in the context candidate sets. For left context, the evidence is the probability that a character $C_{n,m}$ will occur if a character $C_{n-i,j}$ occurred i characters before it, where $1 \le i \le L$ and $1 \le j,m \le S$. For right context, the evidence is equal to the probability that a character $C_{n+i,j}$ will occur after i characters from $C_{n,m}$. Each candidate character will have $S \times L$ evidences from the left context and $S \times R$ evidences from the right context. As determined in Chapter 4, the context window size is 9 for each context, i.e. $L = R = 9$ and the candidate size $S = 10$. There will be a maximum of $10 \times 9 + 10 \times 9 = 180$ evidences for each candidate character. We let E_j as a set that contains all the evidence values for candidate $C_{n,j}$.

The second step is to combine all the evidences from E_j that support each $C_{n,j}$ in the candidate set. The character with the highest combined probability will be selected. The main concern in this part is how do we combine the evidences. The simplest method is to select the highest evidence value from E_j. However this method does not utilize all the evidences and can be affected by spurious high association. Two methods that utilize all evidences are experimented. One is to apply Dempster-Shafer (DS) theory, which in its simplest form is just to multiply all the probabilities. Another method is Confidence Factor (CF) which combine the probabilities using the formula $a_{new} = p \times (1 - a_{old}) + a_{old}$.

5.4.3 Result and evaluation

The results are tabulated below:

	Bigram	non-contiguous model		
		Maximum	DS theory	CF
Accuracy	78%	52%	73%	74.6%

Table 5.2 Performance of the non-contiguous model

Table 5.2 shows that the result of our proposed model is not that promising. If we use the first approach to combine evidences i.e. select the highest evidence, the accuracy is only 52%. Even the best result, using confident factor to combine evidences, achieve only 74.6% accuracy, which is still poorer than the baseline. The poor result may due to the noise introduced from far away context. To reduce this effect, we introduce a distance factor before combining the evidence. We multiple a factor of $1/d$, where d is the distance between the two characters, to their association probability. The accuracy improved to about 79% in the confidence factor case.

The performance of the non-contiguous model is just 1% improved over the bigram model. However, we can utilize both models and combine their prediction results if their prediction is not the same. To show whether our non-contiguous model could complement the bigram model, we run the two models in parallel and compare their prediction output. We discover that the two models made the same decision in most of the time (Around 89 out of 100 cases of time). This result implies that it is not worth combining the two models as it only gives extra information only in about 10% of the time.

We run the non-contiguous model (use Confidence factor to combine evidence with distance factor) for context size of 0 to 9. The results are tabulated in Table 5.3.

Context size	0	1	2	3	4
Accuracy	9.1%	80.2%	79.6%	79.4%	79.1%

5	6	7	8	9
78.9%	78.9%	78.7%	78.6%	78.6%

Table 5.3 Performance of the non-contiguous model for context size up to 9

We observed that the accuracy of the non-contiguous model in fact decrease as the

context size increase. This finding disagree with our belief that increase in context size will provide more information for the model to make a better prediction.

The main reason for the poor performance using large context size may be the sparseness of training data. The PH corpus we use to gather association statistic is not large enough. The associations thus obtained may not be statistically valid and they introduce noise instead of information to the model. We believe that a much larger training corpus must be used if we want to obtain meaningful and valid far away associations. Due to limitation in resource, we cannot perform the experiment at this moment.

Chapter 6

Error Detection Approach

While attempts to improve the (bigram) language models were (quite) successful, the high recognition accuracy (about 96% [1]) is still inadequate for professional data entry services, which typically require an error rate lower than 1 in 1,000. For quality control, these services often estimate the error rate by sampling. They then identify and correct the errors manually in order to achieve the required quality. For a large volume of text, automatic error identification is perhaps more important than automatic error correction. This is because (1) manual correction is more reliable than automatic correction, (2) manual error sampling is simple and (3) more manual efforts are required in error identification than correction. For example, if after applying language model, the accuracy of a recognition system is 96%, there remains 4% of error in the output. If the error identification is 97% and there is no error in manual correction. Then 97% of the errors will be corrected which improve the accuracy from 96% to 99.9%.

In typical applications, the accuracy of the bigram language model may not be as high as those reported in the literature because the data may be in a different genre than that of the training data. For evaluation, we tested a bigram language model with text from a novel domain and its accuracy dropped significantly from 96% to 78%, which is similar to English [13]. Improvement in the robustness of the bigram language model across different genre is necessary and several approaches are available, based on detecting errors of the language model.

The adaptive approach is to automatically identify the errors and manually correcting them. The information about the correction of errors is used to improve the bigram language model. For example, the bigram probabilities of the language model may be estimated and updated with the corrected data. In this way, future occurrences of these errors are reduced.

The hybrid approach uses a different language model to correct the identified errors. This additional language model could be computationally more expensive than the bigram language model because it would only be used on the identified errors. Also, topic detection [13] followed by language model selection [12] could be employed to find an appropriate language model for a specific topic. This is because topic-dependent words are main source of errors and depending on the topic, different language model gives different performance.

The integrative approach improves the language model accuracy by using more sophisticated recognizers, instead of a complementary language model. The more sophisticated recognizer would simply give the correctly recognized character or a set of different results. The bigram language model could than be re-applied to this set. This integrates well with the coarse-fine grain recognition architecture proposed by Nagy [33] back in the 1960s. Coarse grain recognition provides the candidates for the language model to select. Fine grain recognition, which is expensive, is carried out only where the language models failed. Finally, it is possible to combine all the different approaches (i.e. adaptive, hybrid and integrative).

Although error detection in language models is significant, there is only very little research work in this area. Perhaps, these errors were considered random and therefore hard to detect. However, users can detect errors quickly. We suspect that some of these errors are systematic due to the properties of the underlying

language model or due to the properties of the language.

We adopt a pattern recognition approach to detect errors in the bigram language model for Chinese. Each output is assigned to either the class of correct output or the class of errors. The assignment of a class to an output is based on a set of features. We explore a number of features to detect errors, which are classified into model-based features and language-specific features.

The proposed approach can work with Indo-European languages at the word-bigram level. However, language-specific features have to be identified. In addition, this approach could be adopted by the *n*-gram language models. In principal, the model-based features can be found or evaluated similar to the bigram language model. For example, if the trigram probability (instead of bigram probability) is low, then the likelihood of a language model error is high.

6.1 Features

We evaluate individual features for error detection. Articles from Yazhou Zhoukan (YZZK) magazine (4+ Mbytes)/PH corpus (Guo and Liu, 1992) (7+ Mbytes) are used for evaluation. We use the recall and precision measurements for evaluation. The recall is the number of errors identified by a particular feature divided by the total number of errors. The precision is the number of errors identified by a particular feature divided by the total number of times the feature indicate that there are errors. In the first subsection, we describe some model-based features. Next, we describe the language-based features. In the last subsection, we discuss the combined use of both types of features.

6.1.1 Model-based features

The bigram language model selects the most likely path P_{max} out of a set S. The probability of a path s in S is simply the product of the conditional probabilities of one character c_i after the other c_{i-1} where $s = c_0.c_1..c_{|s|}$, after making the Markov assumption. Formally,

$$P_{\max} = \arg\max_{s \in S}\{p(s)\}$$
$$= \arg\max_{s \in S}\left\{ p(c_0)\prod_i p(c_i \mid c_{i-1}) \mid c_0 c_1 ... c_{|s|} = s \right\}$$

The set s is generated by the set of candidate characters for each recognition output. The recognizer may supply the set of candidate characters. Alternatively, a coarse grain recogniser may simply identify the best-matched group or class of characters. Then, members of this class are the candidate characters. Formally, we use a function $h(.)$, that maps the recognition position to a set of candidate characters, i.e. $h(i) = \{c_{i,j}\}$. We can also define the set of sentences in terms of $h(.)$, i.e. $S = \{s \mid s = c_0 c_1 ... c_n, \forall i,\ c_i \in h(i)\}$.

6.1.1.1 Features based on zero probabilities ($F_{1,1}$)

One feature to detect errors is to count the number of conditional probabilities $p(c_i|c_{i-1})$ that are zero, between 2 consecutive positions. Zero conditional probabilities may be due to insufficient training data or may be because they represent the language properties. Figure 6.1 shows the likelihood of an error occurring against occurring against the percentage $Z(i)$ of the conditional probabilities over two consecutive positions (i.e. $i - 1$ and i) that are zero, estimated based on the YZZK and PH data. Formally,

$$Z(i) = \frac{card(\{c_{i-1}c_i \in h(i-1) \times h(i) \mid p(c_i \mid c_{i-1}) = 0\})}{card(\{c_{i-1}c_i \in h(i-1) \times h(i)\})} \times 100\%$$

where $card(.)$ is the cardinality of the argument.

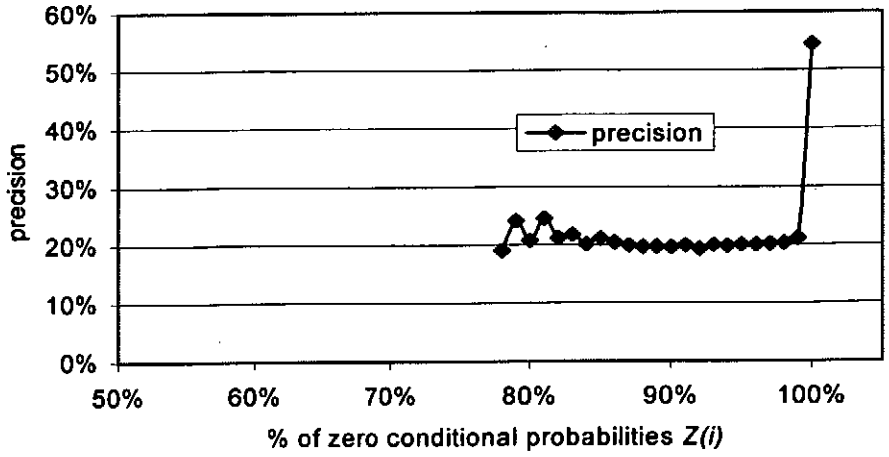

Figure 6.1 The language model output errors against percentages of zero conditional probabilities between the candidate sets.

6.1.1.2 Features based on low probability ($F_{1,2}$)

When there are insufficient data, the conditional probabilities that are small are not reliable. If P_{max} have selected some conditional probabilities that are low, then probably there are no other choices from the candidate sets. Hence, the insufficient data problem may occur in that particular P_{max}. Specifically, these conditional probabilities $p(c_i \mid c_{i-1})$ are along the maximum likelihood path P_{max} and they are defined as $\{p(c_i|c_{i-1}) \mid P_{max} = p(c_0) \prod p(c_i \mid c_{i-1})\}$.

In Figure 6.2, we plot the likelihood of errors identified against the different logarithmic conditional probability values (i.e. $\{log\ p(c_i|c_{i-1})\}$). When the recall increases, unfortunately, the precision drops.

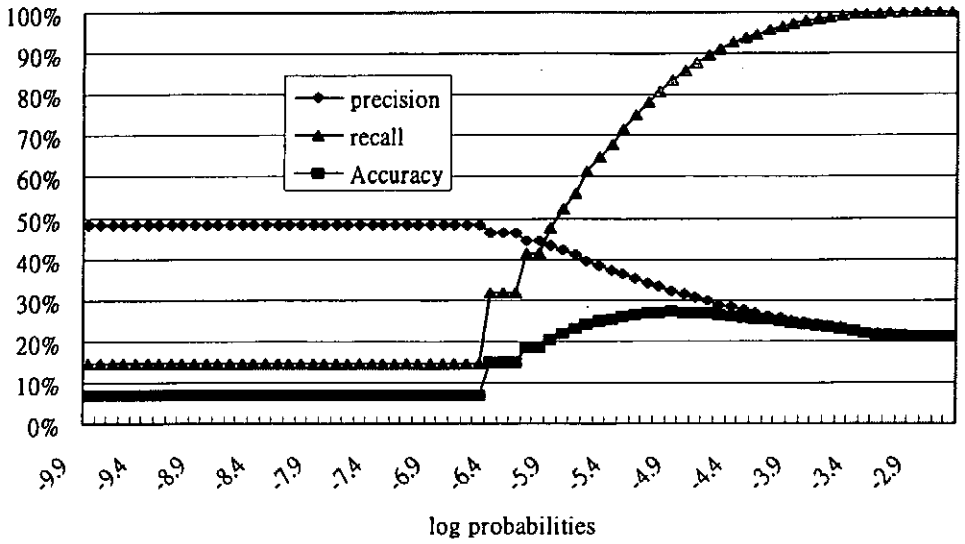

Figure 6.2 The precision, recall and accuracy (i.e. recall × precision) of detecting language model errors by examining the logarithm of conditional probabilities, $log\ p(c_i|c_{i-1})$, on the maximum likelihood path P_{max}.

6.1.2 Language-specific features

The language-specific features are based on applying the word segmentation algorithm [34] to the maximum likelihood path. The ROCLING [35] word list used for segmentation has 78,000+ entries.

6.1.2.1 Features based on word length ($F_{2,1}$)

If the matched word in the maximum likelihood path is long, then we expect the likelihood of an error is low because long words are specific. Figure 6.3 shows the precision of detecting the matched word is correct and the recall of errors in multi-character words. In general, the longer the matched words, the more likely that they are correct and the likelihood of missing undetected long words is small.

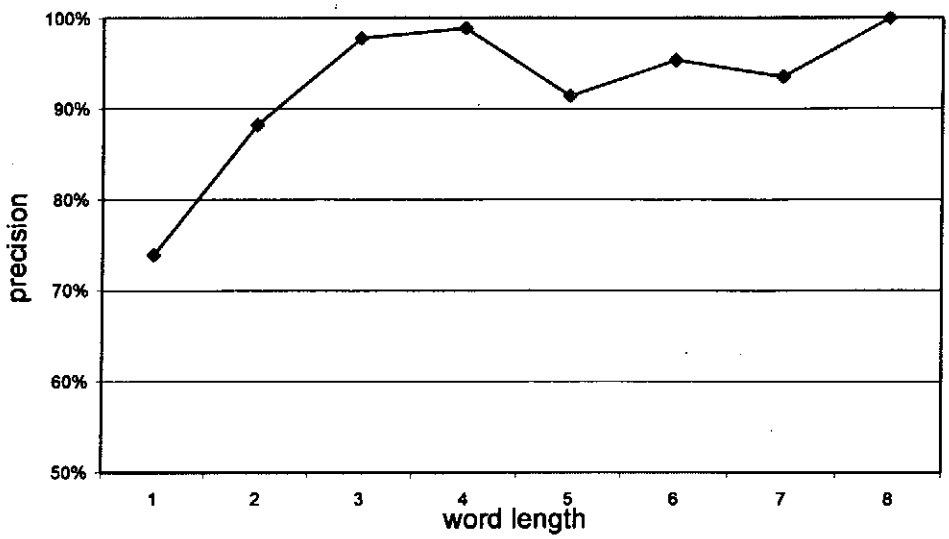

Figure 6.3 The precision of correct matched words against word lengths.

6.1.2.2 Features based on single-character sequences ($F_{2,2}$)

In word segmentation, when there are no entries in the dictionary, the input is segmented into single characters. Thus, Lin *et al* [36] noted that single-character sequences after word segmentation might indicate segmentation problems. Here, we apply the same technique for the detection of errors. If we count on the per character basis, the recall of error is 80% and the precision in error identification is 35%. If we count multi-character words and a sequence of single-characters as blocks, then the recall of errors is 79% and the precision in finding one or more errors in the block is increased to 51%.

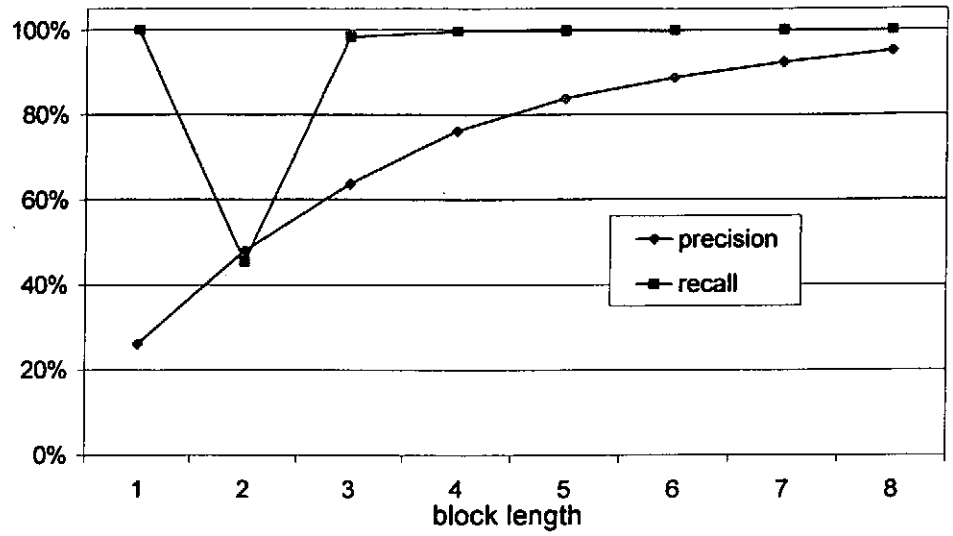

Figure 6.4 The precision and recall of single-character sequences of different lengths.

Similar to matched words in the maximum likelihood path, the error detection performance of single-character sequences may depend on their length. Therefore, we plotted the recall and precision of detecting errors against the length of the single-character sequences. According to Figure 6.4, as the length of the single-character sequence is large, the likelihood of an error is larger. The recall of errors is particularly low for single-character sequences that have 2 characters. The other single-character sequences (i.e. its length is not equals to 2) have almost 100% recall. One possible reason why 2 single-character sequences achieved low precision is that there are many spurious bigrams and therefore false match.

6.1.3 Combined use of features

We carried out a preliminary study using a single classifier to detect errors using features mentioned in subsection 1.1 and 1.2. The error detection follows Algorithm A, which accepts a set of confusion character sets from position 1 to k and returns the error array indicating whether the positions between 1 and k have errors or not. Algorithm A first produces the maximum likelihood path P_{max} using the language model, represented as the function LM(.). The next step segment

P_{max} into a sequence W of words, $w_1.w_2...w_m$, based on forward maximal matching, FMM(.). For each identified word w_i, the classifier decides whether an error is found or not based on the classifier(.) in Step 7. Step 5 resets the default error array values to false. The classifier(.) function derive feature values for w_i since the position *pos* in P_{max} is known and whether the identified word is a single-character or not is also known (i.e. whether $|w_i| = 1$).

```
Input: h(1),⋯,h(k)
Output: boolean array error[1..k]
Step 1:  string Pmax ← LM(h(1),⋯,h(k));
Step 2:  string array W[1..m] = w1⋯wm ← FMM(Pmax);
Step 3:  integer pos ← 0;
Step 4:  for i ← 1 to m do
Step 5:      for j ← pos+1 to pos+|wi| do error[j] ← false;
Step 6:      pos ← pos + |wi|;
Step 7:      classifier(pos, wi);
```

Algorithm A: The error detection scheme using a single classifier.

Algorithm A is the error detection scheme using a single classifier. LM(.) returns the maximum likelihood path P_{max} given a sequence of confusion character sets, *h(1)...h(k)*. FMM(.) returns the segmented word sequence of the argument using the forward maximal matching algorithm and the segmented word sequence is stored in the string array *W*.

Our preliminary study used the Bayesian classifier for Algorithm A, with features discussed in subsection 2 as input. Algorithm A achieved 83% recall but 35% precision, which can be achieved using language specific features only (i.e. $F_{2,2}$ with 80% recall and 35% precision in error identification). These results are comparable with existing *n*-gram detection methods [37] for Japanese optical character recognition error detection, in which the recall is 85% and the corresponding precision is about 30%. However, if a morphological analyzer is

used with the *n*-gram statistics as in [38], the recall is 97% and the precision is about 34%.

To further improve our detection performance, we try to combine the use of these features in a more careful manner. We divided the error detection into 3 general cases: (1) single character (feature $F_{2,2}$); (2) single-character sequence of length 2 (feature $F_{2,2}$) and (3) 2 character words (feature $F_{2,1}$). Each case is assigned a classifier to detect errors and for each case, additional features are considered in the following subsections. Single-character sequences longer than 2 are considered as having errors (Figure 6.4). Words of length longer than 2 are considered correct (Figure 6.3).

```
Input: h(1),···,h(k)
Output: boolean array error[1..k]
Step 1:  string Pmax ← LM(h(1),···,h(k));
Step 2:  string array W[1..m] = w1···wm ← FMM(Pmax);
Step 3:  integer pos ← 0;
Step 4:  string T ← Λ; {reset single character sequence}
Step 5:  for i ← 1 to m do begin
Step 6:      for j ← pos+1 to pos+|wi| do error[j] ← false;
Step 7:      if |wi| = 1 then T ← T • wi;
Step 8:      else begin
Step 9:          if |T| = 1 then classifier1(error, pos, T);
Step 10:         else if |T| = 2 then classifier2(error, pos, T);
Step 11:         else if |T| > 2 then
Step 12:             for j ← pos-|T|+1 to pos do error[j] ← true;
Step 13:         T ← Λ; {reset single character sequence}
Step 14:         if |wi| = 2 then classifier3(error, pos, wi);
Step 15:     end; {if-else}
Step 16:     pos ← pos + |wi|;
Step 17: end; {for-loop}
```

Algorithm B: The error detection scheme using multiple classifiers.

Algorithm B shows the basic detection scheme using separate classifiers for the identified 3 cases discussed above. The variable T stores the single character

sequences and it is initially set to the empty string Λ at step 4. By default, the error flags are false (Step 6) unless the classifiers identified errors. At step 9, classifier1(.) sets the error flag at position *pos* for single characters (i.e. $|T| = 1$). At step 10, classifier2(.) sets the error flags at position *pos* and *pos - 1* for single-character sequences of length 2 (i.e. $|T| = 2$) and at step 14, classifier3(.) sets the error flags at position *pos - 1* and *pos* for two character words (i.e. $|w_i|=2$). For single character sequences of length larger than two, their corresponding error flags are set to true at step 12.

6.1.3.1 Single characters

In word segmentation, a string, which does not math with any dictionary entry, results as single character. Single characters can have different part-of-speech tags.

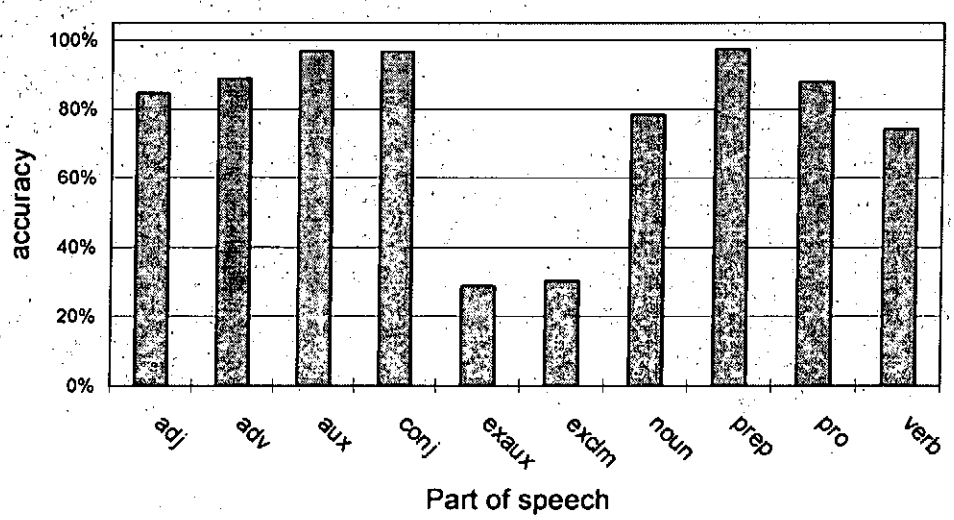

Figure 6.5 Single characters and their corresponding language model output accuracy for different part-of-speech tags.

Figure 6.5 shows that the accuracy of the language model for the single characters with part-of-speech tags related to exclamations are low. For error detection, a feature is assigned to each part-of-speech tag.

The language model accuracy for single characters may depend on the availability of the left and right context to form high probability bigrams. Therefore, we expect that language model accuracies of single character at the beginning (70%) and end (70%) of a sentence are lower than those in the middle (85%) of the sentence. The worst case occurs when the sentence has only one single character, where the measured accuracy is only 8.75% (this is because it has no bigram context).

6.1.3.2 Two-single-characters sequences

Figure 6.6 shows that language model output accuracy increases as the bigram probability of single-character sequences of length 2 increases. Hence, the bigram probabilities can be used as a feature for detection.

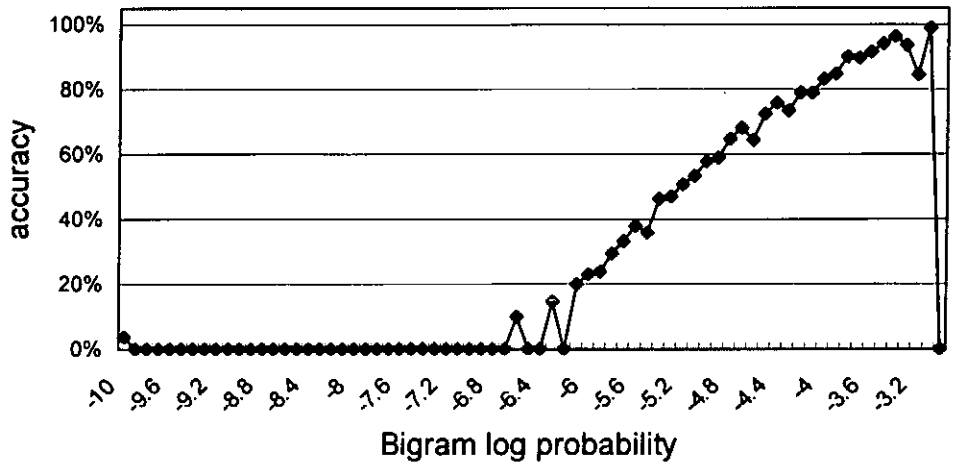

Figure 6.6 The bigram (logarithm) probability of the single-character sequence of length 2.

Similar to single characters, the language model accuracy for 2-single-characters sequences at the start, middle and end of a sentence are 48%, 47% and 30%, respectively. The accuracy is 33% if the sentence is the 2-single-characters sequence.

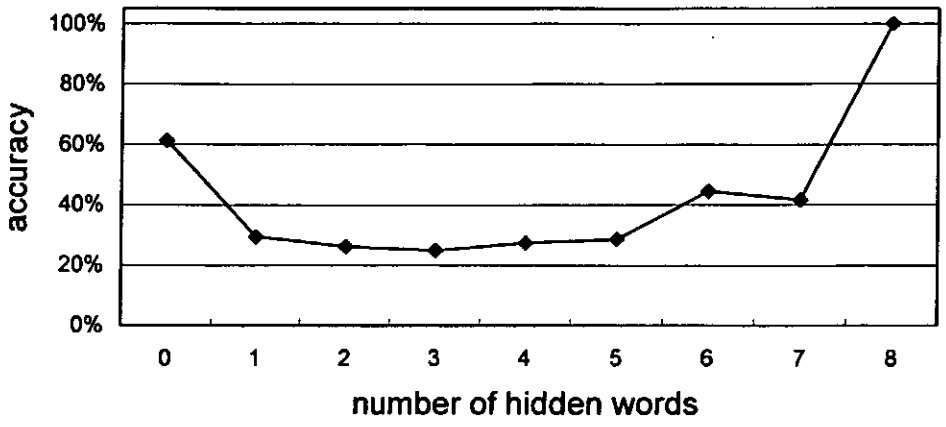

Figure 6.7 Language model accuracy against different number of hidden words (see text).

Another feature for 2-single-character sequences is to examine whether the characters in the two candidate sets can form words that match with the dictionary. These matched words are called hidden words. Figure 6.7 shows that if there are hidden words, the language model accuracy dropped from 60% to 25%. Since there are not many cases with 6-8 hidden words, the accuracy for these cases are not reliable.

6.1.3.3 Two-character words

For 2 character words, the bigram probability (Figure 6.8) can be used as a feature similar to the single-character sequences. The position of these 2 character words in the sentence does not relate to the language model accuracy. Our measured accuracies are 91%, 89% and 91% for the beginning, the end and the middle of the sentence, respectively. Even sentences with a single 2-character word achieved 90% accuracy. Hence, there is no need to assign features for the position of the 2 character words in a sentence. Similar to 2-single-characters sequences, the language model accuracy (Figure 6.9) decreases as the number of hidden words increase in the corresponding 2 sets of candidate characters.

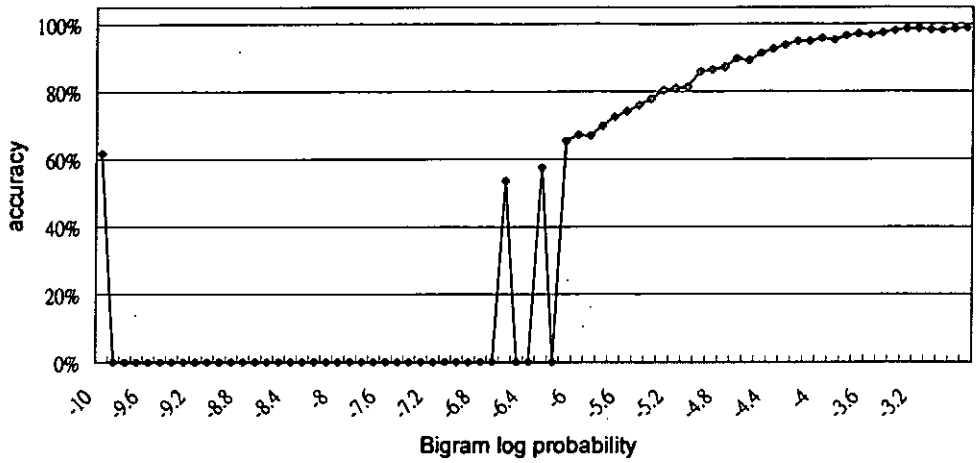

Figure 6.8 The language model accuracy of 2 character words against the bigram probability.

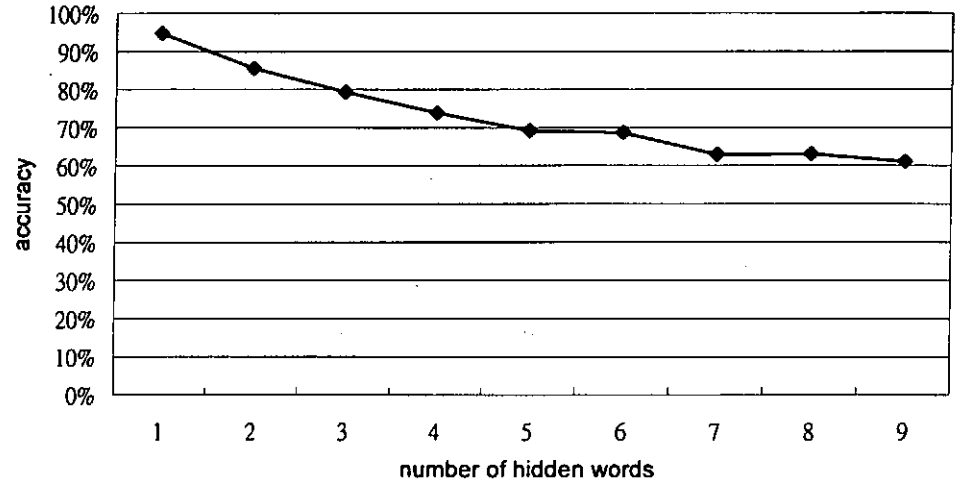

Figure 6.9 The language model accuracy against different number of hidden words.

6.2 Classifiers

One of the problems with using individual features is that the recall and precision are not very high, except the language-specific features. It is also difficult to set the threshold for detection because of the precision-recall trade-off. In addition, there may be some improvement in detection performance if features are combined for detection. Therefore, we adopt a pattern recognition approach to

detect errors.

Several classifiers are used for error identification because we do not know which features work well with which classifiers. Three types of classifiers will be examined: Bayesian, decision tree and neural network.

6.2.1 Bayesian classifier

The Bayesian classifier is simple to implement and is compatible with the model-based features. Given the feature vector $\boldsymbol{x}$, the Bayesian detection scheme assigns the correct class w_c and the error class w_e, using the following rule:

$g_c(\boldsymbol{x}) > g_e(\boldsymbol{x})$ assign w_c

Otherwise assign w_e

where $g_c(.)$ and $g_e(.)$ are:

$$g_c(x) = -(x-\mu_c)^T \Sigma_c^{-1}(x-\mu_c) - \log|\Sigma_c| + 2\log p(w_c)$$
$$g_e(x) = -(x-\mu_e)^T \Sigma_e^{-1}(x-\mu_e) - \log|\Sigma_e| + 2\log p(w_e)$$

μ_c and μ_e are the mean vectors of the class w_c and w_e, respectively, Σ_c and Σ_e are the covariance matrices of the class w_c and w_e, respectively, and $|.|$ is the determinant.

6.2.2 Decision tree

Originally, we tried to use the support vector machine (SVM) [39] but it could not converge. Instead, we used the decision tree algorithm C4.5 by Quinlan [40]. Decision trees are known to produce good classification if clusters can be bounded by some hyper-rectilinear regions. We trained C4.5 with a set of feature vectors, described in sub section 1.3.

6.2.3 Neural network

We use the multi-layer perceptron (MLP) because it can perform non-linear

classification. The MLP has 3 layers of nodes: input, hidden and output. Nodes in the input layer are fully connected with those in the hidden layer. Likewise nodes in the hidden layer are fully connected to the output layer. For our application, one input node corresponds to a feature in sub section 1.3. The value of the feature is the input value of the node. Two output nodes indicate whether the current character is correct or erroneous. The number of hidden nodes is 2-4, calculated according to Fujita [41].

The output of each node in the MLP is the weighted sum of its input, which is transformed by a sigmoid function. Initially, the weights are assigned with small random numbers, which are adjusted by the gradient descend method with learning rate 0.05 and momentum 0.1.

6.3 Evaluation

In the evaluation, the training data is the PH corpus and the testing data is the YZZK magazine articles (4+ Mbytes), downloaded from the Internet. In handwritten character recognition, the optimal size of the number of candidates is 6 [5]. For robustness, each recognized character in our evaluation is selected from 10 candidates.

Error detection is performed only on language model output that based on out-of-domain test data. The recognition rate of bigram model for in-domain test data is already very high, over 95% in our experiment. So, we concentrate only on out-of-domain test data. Much works have been done on topic detection [42][43], we may employ the technique to identify the domain/topic of the test corpus. We do error detection only for out-of-domain test data.

We measured the performance in terms of recall, precision and the manual effort

reduction in scanning the text for errors. The recall is the number of identified errors over the total number of errors. The precision is the number of identified errors over the total number of cases classified as errors. The amount of saving in manual scanning for errors is called the skip ratio, which is the number of blocks classified as correct over the total number of blocks. The recall and the skip ratio are more important than the precision because post error correction (manual or automatic) can improve the recognition accuracy. It is possible to combine the recall and precision into one, using the F measures [44] but the value for rating the relative importance is subjective.

Cases	Distribution	Measure	Random candidate			Visually similar candidate
			Bayes	C4.5	MLP	C4.5
Single character	15%	Recall	71%	56%	28%	50%
		Precision	40%	75%	71%	74%
2 single characters	10%	Recall	60%	84%	83%	93%
		Precision	88%	82%	80%	92%
2-character words	54%	Recall	60%	27%	9%	85%
		Precision	29%	70%	62%	92%
> 2 single character	16%	Recall	100%			
		Precision	90%			
> 2 character words	5%	Accuracy[1]	78%			
Overall		Recall	79%	73%	75%	80%
		Precision	60%	81%	80[1]%	91%
		Skip Ratio	65%	76%	66%	75%

Table 6.1 The performances of the 3 types of classifiers in detecting language model errors.

[1] Since we predict all word block with length longer than two as all correct, we cannot measure the recall or precision for error prediction. We measure the accuracy for correct prediction instead.

Table 6.1 shows the classification performance of the Bayesian classifier. The recall of errors by the Bayesian classifier has reduced slightly from 83% using a single classifier to 79% using 3 classifiers but the precision improved from 51% to 60%. Also, the skip ratio is 65%, which is much higher than the skip ratio of 0.1% if we did not use the classifier. Although the MLP has a higher precision (80%), its recall is slightly lower than the Bayesian classifier. The skip ratio of the both Bayesian and MLP classifiers are about the same. The column "Distribution" represent the relative quantity that each of the five case contributed. Over half of the block identified are 2 character word block.

6.3.1 Using visually similar candidate sets

The previous experiments on bigram model are based on randomly generated candidate set. However, for real character recognizer, candidate character sets are generated based on similarity measure: a character c bitmap inputted to the recognizer is converted to a feature vector v, the vector is compared to a list of feature vectors representing characters in the whole character set. Those characters that have feature vector similar to v will be extracted as candidate characters for c. To ensure our experiment results apply to real world situation, we find a method to extract candidate characters.

To measure the similarity amongst the characters, we render all the characters in the Big5 character set from a Chinese True type font file and convert them to bitmaps. We use freetype 2, a free true type font engine from freetype.org [45], to generate the bitmaps. The bitmaps are 64 pixels by 64 pixels. We match the bitmaps pixel by pixel and count the number of unmatched pixels. The matching ratio or similarity between two characters will be calculated as $1-\frac{n}{64\times 64}$, where

n is the number of unmatched pixels between the two characters. We select ten characters with the highest matching ratio with the interested character as its candidates. Table 6.2 shows some of the sample candidate sets. These samples are randomly picked and we can observe that the candidate characters are visually similar to the interested character.

	Visually similar characters
儿	冗兀兀北丸江扎一八
子	干丁了孑下于亍丫孑
仇	汍仉伉价伉汍犰付仍
賬	張掁悵棖眶脹哻咀哭
鵪	鶉鶉鶉鷦鵚鵡鸛鸛鵪
鴛	鴛鴛鴛鴛鴛鴛鴛鴛鴛
癩	瘥癗癒痟病瘤瘼痌疵

Table 6.2 Samples of candidates generated using pixel difference

We run bigram model on the generated visual similar candidates and detect errors with our error detection approach. The result is tabulated in Table 6.1. The overall error detection recall and precision is 80% and 91% respectively, which are much improved over the case when random candidates are used. The improvement is mainly due to fewer confusing words from the candidate sets for the 2-character word case.

6.3.2 Detection speed

Besides the ability to detect error, the speed of execution is also an important issue. We run the on a Sun UltraSPARC 200MHz machine with 256 Megabytes memory. The speed of error detection is 139.7 characters per second. As a reference, the

speed of applying the bigram model is 147.1 characters per second. The execution speed of error detection is comparable to that of bigram model. That means we need twofold the time if we apply error detection after running the language model. The extra time is justified as much human effort can be saved to scan for errors.

6.4 Summary

We have evaluated both model-based and language-specific features for detecting language model errors. Individual model-based features did not yield good detection accuracy as it suffers from the precision-recall trade-off. The language-specific features detect errors better. In particular, matched multi-character words are usually correct. If the model-based and language-specific features are combined as a single feature vector, the recall and precision of errors are 83% and 35%, respectively, which are the same as the use of the language-specific features alone. Therefore, instead of a single classifier, we separated 3 situations identified by the language-specific features and 3 classifiers were used to detect these errors individually. The Bayesian classifier (simplest) achieved an overall 79% recall, 60% precision and 65% skip ratio and the MLP achieved an overall 75% recall, 80% precision and a 66% skip ratio. Similar recall and precision performances were achieved using decision trees. Since the skip ratio is higher (i.e. 76%), the decision tree approach is preferred. Although the precision (so far) is not high (60% - 80%), it is not the most important result because (1) this only represents a minor waste of checking effort, compared with scanning the entire text, and (2) the identified errors will be checked further or corrected either manually or automatically. In order to simulate

real world environment, we generate visual similar candidate sets instead of random candidate. We input them to the bigram model and detect the error from the output. The result is recall: 80%, precision: 91% and skip ratio: 75% and the error detection speed is 139.7 character per second on a Sun UltraSPARC machine.

Chapter 7

Conclusion and Future Work

We run experiments to obtain the error characteristics of an offline Chinese character recogniser. We observed that nearly all errors are substitution and occur randomly for text that can be segmented easily into characters (common for printed characters). However for text that cannot be easily segmented (common for handwritten characters), insertion errors are frequent and the errors usually occur in burst.

To tackle the problem of burst errors, we proposed a non-contiguous language model. The model uses both the left and right context information and treats the characters in the context as independent units. The performance of the model is 79% in terms of accuracy to extract the right candidate character. The result is promising but not a significant improvement over our baseline bigram model.

We claim that bigram model is already a very good language model. It achieves high accuracy with low storage and computation cost.

We proposed a pattern recognition approach to detect errors in language model which combine both dictionary and statistical language modelling approach. The first step is to segment the language model output into blocks of characters. The character blocks are classified into 5 categories: single character block, two characters sequence block, double characters word block, character sequence block with more than two characters, word block with more than two characters. The second step is to extract some statistically features for the block to form a

feature vector. For each category of character block, we extract some statistical features like the bigram probability between the boundary characters with that of the neighbouring block, the part of speech of the singe character block etc. The feature vector is feed into a classifier, which makes a prediction of whether the block is correct, or not. This detection scheme as demonstrated in our experiment is effective. The performance is 80%, 91% and 75% of precision, recall and skip ratio respectively.

The major contributions of this research work are summarized below:

1. We run experiments to discover the error characteristics of an offline Chinese character recognizer and developed an error model. With the knowledge of the error pattern, researcher can design a better pro-processing technique or fine-tune the existing one.
2. We determine the context window size for Chinese language [P1] . The context window size is an essential parameter for context dependent language processing techniques. We developed a non-contiguous language model based on the context window size determined which achieved accuracy comparable to the bigram model.
3. We developed a pattern recognition approach to detect language model error [P2] . This error detection method can help to improve a language model in different ways:
    a. Automatically identify the errors and ask the user to correct them manually. The information about the correction is used to improve the language model to reduce future occurrences of these errors.
    b. Use another language model to correct the identified errors. The language model can be computationally more expensive because it is applied only to the character blocks that have been identified as

incorrect.

c. Use a more sophisticated recognizer to recognize again the character blocks that have been identified as incorrect.

We believe that detecting errors of language models is an effective way to improve language modelling and is worth to devote more research effort. The error detection approach we proposed gives promising performance but there is still much room for improvement. Below are some suggested future works:

1. From Table 6.1, the classification performance for the case of single character and double character word block is not very good. One way to improve it is to discover more useful features for the two cases. The additional features can provide more information for the classifier to make better decision.
2. Due to the time constraint, we only examined three classifiers in our experiments. There exist many other good classifiers, which may give better performance in our application. We suggest doing a comprehensive research on classifier to pick a best suited one or consider combining the classifiers to contribute a final prediction.

Publications

[P1] Hung,K.Y., R.W.P.Luk, D.Yeung, K.Chung and W.Shu (2001) "Determine the Chinese context window size", *International Journal of Computer Processing of Oriental Languages (IJCPOL)* pp. 71-80.

[P2] Hung,K.Y., R.W.P. Luk, D.Yeung, K.Chung, W.Shu (2000) "Detection of language (model) errors", *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.* pp. 87-94.

Bibliography

[1] Pak-Kwong Wong; Chorkin Chan, "An off-line large vocabulary hand-written Chinese character recognizer". *Image Processing, Proceedings., International Conference on*, Vol. 3 , pp. 324 -327 (1997)

[2] Lee, H.-J.; Tung, C.-H.; Chien, C.-H.C., "A Markov language model in Chinese text recognition", *Document Analysis and Recognition., Proceedings of the Second International Conference on*, pp. 72 –75 (1993)

[3] Hong-Wei Hao; Xu-Hong Xiao; Ru-Wei Dai, "Handwritten Chinese character recognition by metasynthetic approach", *Pattern Recognition,* Vol. 30 No. 8, pp. 1321-1328 (1997)

[4] Wagner, R.A. And M.J. Fisher, "The string to string correction problem", *J.ACM*, 21, pp. 168-173 (1974)

[5] Pak-Kwong Wong; Chorkin Chan, "Postprocessing statistical language models for handwritten Chinese character recognizer", *Systems, Man and Cybernetics, Part B, IEEE Transactions on,* Vol. 29, pp. 286 –291 (1999)

[6] Bahl, L.R., F. Jelinek and R.L. Mercer "A maximum likelihood approach to continuous speech recognition", *A maximum likelihood approach to continuous speech recognition, 5:2*, pp. 179-190 (1983)

[7] Nathan, K.S., H.S.M. Beigi, J. Subrahmonia, G.J. Clary and H. Maruyama (1995) "Real-time on-line unconstrained handwriting recognition using

statistical methods", *Proceedings 1995 International Conference on Acoustics, Speech, and Signal Processing*, pp. 2619--2622 (1995)

[8] Jelinek, F. "Up from trigrams: the struggle for improved language models", *Proceedings of Eurospeech*, Vol. 3, pp. 1037-1040 (1991)

[9] Rosenfeld, R., "Adaptive statistical language modeling" a maximum entropy approach", Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh (1994)

[10] Huang, X., F. Alleva, H. Hon, M. Hwang,, K. Lee and R. Rosenfeld, "The SPHINX-II speech recognition system : an overview", *Computer Speech and Lanaguage*, pp. 137-148 (1993)

[11] Ward, W. and S. Issar, "A class based language model for speech recognition", *Proc. IEEE ICASSP,* Vol. 1, pp.416-418 (1996)

[12] Kenne, P.E. and M. O'Kane, "Hybrid language models and spontaneous legal discourse", *Proc. ICSLP,* Vol. 2, pp. 717-720 (1996)

[13] Mahajan, M., D. Beeferman and X.D. Huang, "Improving topic-dependent modeling using information retrieval techniques", *Proc. IEEE ICASSP*, Vol. 1, pp.541-544 (1999)

[14] Lochovsky, A.F. and K-H. Chung, "Homonym resolution for Chinese phonetic input", *Communications of COLIPS*, 7:1, pp. 5-15 (1997)

[15] Oommen, B.J. and K. Zhang, "The normalized string editing problem revisited", *IEEE Trans. on PAMI*, 18:6, pp. 669-672. (1996)

[16] Jin, Y., Y. Xia and X. Chang, "Using contextual information to guide

Chinese text recognition", *Proc. ICCPOL '95*, pp. 134-139 (1995)

[17] Xia, Y., S. Ma, M. Sun, X. Zhu, Y. Jin and X. Chang, "Automatic post-processing of off-line handwritten Chinese text recognition", *Proc. ICCC*, pp. 413-416 (1996)

[18] Chien, L.F., Chen, K.J., Lee, L.S. "A best-first language processing model integrating the unification grammar and Markov language model for speech recognition applications", *IEEE Trans. Speech and Audio Processing*, 1:2, pp. 221 –240 (1993)

[19] His-Jian Lee; Cheng-Huang Tung, "A language model based on semantically clustered words in a Chinese character recognition system", *Third International Conference on Document Analysis and Recognition Proceedings*, Vol. 1, pp. 450 -453 (1995)

[20] Kae-Cherng Yang; Tai-Hsuan Ho; Lee-Feng Chienq; Lin-Shan Lee, "Statistics-based segment pattern lexicon-a new direction for Chinese language modeling", *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 169 -172 (1998)

[21] Iyer, R., M. Ostendorf and M. Meteer, "Analyzing and predicting language model performance", *Proc. IEEE Workshop Automatic Speech Recognition and Understanding*, pp. 254-261 (1997)

[22] Jian-Yun Nie, Marie-Louise Hannan, Wanying Jin, "Combining dictionary, rules and statistical information in segmentation of Chinese", *Computer processing of Chinese and oriental languages*, Vol. 9, No. 2, pp. 125-143 (1995)

[23] Xiaohong Huang, Zhensheng Luo, Jian Tang, "A quick method for Chinese word segmentation", *IEEE International Conference on Intelligent Processing Systems*, Vol. 2, pp. 1773 -1776 (1997)

[24] Tian Bin; Cheng Jun; Yi Kechu; Wang Hui, "A Chinese word dividing algorithm based on statistical language models", *3rd International Conference on Signal Processing*, Vol. 1, pp. 805 -808 (1996)

[25] Horst Bunke, Alberto Sanfeliu, "String matching for structural pattern recognition", *Syntactic and structural pattern recognition – Theory and applications*, World Scientific, pp. 119-144 (1990)

[26] J.M. Lucassen and R.L. Mercer, "An information theoretic approach to the automatic determination of phonemic baseforms", *Proc. Int. Conf. on Acoustics, Speech, and Signal Proc.*, Vol. 3, pp. 42.5.1-42.5.4 (1984)

[27] F. Smadja, "Retrieving collocations from text: Xtract", *Computational Linguistics*, Vol. 19, No. 1, pp. 143-177 (1993)

[28] D. Wu and Pascale Fung, "Statistical Augmentation of a Chinese Machine-readable dictionary", *Proceedings of the 2nd Annual Workshop on Very Large Corpora*, Kyoto, Japan (1994)

[29] J. Guo and H.C. Liu, "PH – a Chinese corpus for pinyin-hanzi transcription", *ISS Technical Report, TR93-112-0*, Institute of Systems Science, National University of Singapore (1992)

[30] M_S. Chen, J. Han and S. Yu, "Data mining: an overview from a database perspective", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8,

No. 6, pp.866-883 (1996)

[31] L.A. Goodman and W.H. Kruskal, "Measures of association for cross classifications", *Journal of the American Statistical Association*, Vol. 49, pp.732-764 (1954)

[32] John Yen, "Generalizing the Dempster-Shafer Theory to Fuzzy Sets", *IEEE transactions on systems, man, and cybernetics*, pp. 559-569 (1990)

[33] Nagy, G., "Chinese character recognition: twenty-five-year retrospective", *Proc. 9th Int. Conf. on Pattern Recognition*, Vol. 1, pp. 163-167 (1988)

[34] Kit, C., Y. Liu and N. Liang, "On methods of Chinese automatic word segmentation", *Journal of Chinese Information Processing*, 3:1, 13-20 (1989)

[35] Chen, K.-C. and C.R. Huang (eds.) "Chinese word class analysis", *Technical Report 93-05*, Chinese Knowledge Information Processing Group, Institute of Information Science, Academia Sinica, Taiwan (1993)

[36] Lin, M-Y., T-H. Chiang and K-Y. Su "A preliminary study on unknown word problem in Chinese word segmentation", *Proc. ROCLING VI*, pp.119-141 (1993)

[37] D. Yarowsky, "Homograph Disambiguation in Speech Synthesis", *Proceedings of the 2nd ESCA/IEEE Workshop on Speech Synthesis*, New Paltz, NY, pp. 244-247 (1994)

[38] K. Church, "Word association norms, mutual information, and lexicography", *Computational Linguistics*, Vol. 16, No. 1, pp. 22-29 (1990)

[39] Vapnik, V., *The Nature of Statistical Learning Theory*, Springer-Verlag, New

York, (1995)

[40] Quinlan, J.R., *C4.5 programs for machine learning*, Morgan Kaufmann, CA., (1993)

[41] Fujita, O., "Statistical estimation of the number of hidden units for feedforward neural networks", *Neural Networks*, Vol. 2, pp. 851-859 (1998)

[42] K. Seymore, R. Rosenfeld, "Large-scale Topic Detection and Language Model Adaptation", Carnegie Mellon University Technical Report (1997).

[43] Wayne, C. L., "Multilingual Topic Detection and Tracking: Successful Research Enabled by Corpora and Evaluation.", *Proceedings of LREC-2000* (2000)

[44] Van Rijsbergen, *Information Retrieval*, Butterworths, London., (1979)

[45] The FreeType Project, www.freetype.org

[46] Govindaraju, V.; Kim, G., Srihari, S.N., "Paradigms in handwriting recognition", *IEEE International Conference on Systems, Man , and Cybernetics.*, Vol. 2, pp. 1498 -1503 (1997)

[47] Seiler, R., Schenkel, M., Eggimannn, F., "Off-line cursive handwriting recognition compared with on-line recognition", *Proceedings of the 13th International Conference on Pattern Recognition*, Vol. 3, pp. 505 -509 (1996)

[48] K.Y.K. Lau and R.W.P. Luk, "Word-sense classification by hierarchical clustering", *Proceedings of PACLIC 12*, National University of Singapore, pp.236-247 (1998)

[49] R.F. Simmons and Y-H. Yu, "The acquisition and use of context-dependent grammars of English", *Computational Linguistics*, Vol. 18, No. 4, pp. 391-418 (1992)

[50] B. van Coile, "Inductive learning of pronunciation rules with DEPES", *Proceedings of ICASSP*, Vol. 2, Toronto, Canada, pp. 745-748 (1991)

[51] T.J. Sejnowski and C.R. Rosenberg, "NETtalk: A parallel network that learns to read aloud", *Technical Report JHU/EECS-86/01*, Johns Hopkins University, Department of Electrical Engineering and Computer Science, (1986)