Detection of Language (Model) Errors

K.Y. Hung, R.W.P. Luk, D. Yeung, K.F.L. Chung and W. Shu
Department of Computing
Hong Kong Polytechnic University
Hong Kong
E-mail: {cskyhung, csrluk, csdaniel, cskchung, cswshu}@comp.polyu.edu.hk

Abstract

The bigram language models are popular, in much language processing applications, in both Indo-European and Asian languages. However, when the language model for Chinese is applied in a novel domain, the accuracy is reduced significantly, from 96% to 78% in our evaluation. We apply pattern recognition techniques (i.e. Bayesian, decision tree and neural network classifiers) to discover language model errors. We have examined 2 general types of features: model-based and language-specific features. In our evaluation, Bayesian classifiers produce the best recall performance of 80% but the precision is low (60%). Neural network produced good recall (75%) and precision (80%) but both Bayesian and Neural network have low skip ratio (65%). The decision tree classifier produced the best precision (81%) and skip ratio (76%) but its recall is the lowest (73%).

Introduction

Language models are important post-processing modules to improve recognition accuracy of a wide variety of input, namely speech recognition (Balh *et al.*, 1983), handwritten recognition (Elliman and Lancaster, 1990) and printed character recognition (Sun, 1991), for many human languages. They can also be used for text correction (Ron *et al.*, 1994) and part-of-speech tagging.

For Indo-European languages, the word-bigram language model is used in speech recognition (Jelinek, 1989) and handwriting recognition (Nathan *et al.*, 1995). Various ways to improve language models were reported. First, the model has been extended with longer dependencies (e.g. trigram) (Jelinek, 1991) and using non-contiguous dependencies, like trigger pairs (Rosenfeld, 1994) or long distance *n*-gram language models (Huang *et al.*, 1993). For better probability estimation, the model was extended to work with (hidden) word classes (Brown *et al.*, 1992, Ward and Issar, 1996). A more error-driven approach is the use of hybrid language models, in which some detection mechanism (e.g. perplexity measures [Keene and O'Kane, 1996] or topic detection [Mahajan *et al.*, 1999]) 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 (e.g. Lochovsky and Chung, 1997), 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 (Wagner and Fisher, 1974; Oommen and Zhang, 1974) 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 (Jin *et al.*, 1995; Xia *et al.*, 1996) because it (1) achieved 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 for the Asian languages but words need to be identified. For Asian languages, the model was integrated with syntactic rules (Chien, Chen and Lee, 1993). Class based language model (Lee and Tung, 1995) was also examined but the classes are based on semantically related words. A new approach (Yang *et al.*, 1998) 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 (Iyer *et al.*, 1997).

While attempts to improve the (bigram) language models were (quite) successful, the high recognition accuracy (about 96%) is not adequate for professional data entry services, which typically require an error rate lower than 1 in 1,000. As part of the quality control exercises, these services estimate their error rate by sampling, and they identify and correct the errors manually to achieve the required quality. Faced with a large volume of text, the ability to automatically identify where the errors are is perhaps more important than automatically correcting errors, in post-editing because (1) manual correction is more reliable than automatic correction, (2) manual error sampling can be carried out and (3) more manual efforts are required in error identification than correction due to the large volume of text. For example, if the identification of errors is 97% and there are no errors in error correction, then the accuracy of the language model is improved from 96% to 99.9% after error correction.

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 (Mahajan *et al.*, 1999). 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.

One (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.

Another (hybrid) approach uses another language model to correct the identified errors. This language model can be computationally more expensive than the bigram language model because it is applied only to the identified errors. Also, topic detection (Mahajan et al., 1999) and language model selection (Keene and O'Kane, 1996) can be applied to those area to find a more appropriate language model because usually topic-dependent words are those causing errors.

Another (integrative) approach improves the language model accuracy using more sophisticated recognizers, instead of a complementary language model. The more sophisticated recognizer may give a set of different results that the bigram language model can re-apply on or this recognizer simply gives the recognized character. This integrates well with the coarse-fine recognition architecture proposed by Nagy (1988) back in the 1960s. Coarse recognition provides the candidates for the language model to select. Fine, expensive recognition 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).

Given the significance in detecting errors of language models, there is little work in this area. Perhaps, it was considered that these errors were random and therefore hard to detect. However, users can detect errors quickly. We suspect that some of these errors may be systematic due to the properties of the language model used or due to language specific properties.

We adopt a pattern recognition approach to detecting errors of the bigram language model for the Chinese language. 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 discovered for the particular language. In addition, this approach can be adopted for *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.

This paper is organized as follows. Section 1 discusses various features and some preliminary evaluation of their suitability for error identification. Section 2 describes 3 types of classifiers used. In section 3, our evaluation is reported. Finally, we conclude.

1. Features

We evaluate individual features for error detection because they are important to the success of 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.

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,

$$P_{\max} = \arg\max_{s \in S} \{p(s)\}$$

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

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 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. $S = \{s \mid s = c_0 c_1 \ldots c_n, \forall i, c_i \in h(i)\}$.

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 1 shows the likelihood of an error occurring against the percentage of the conditional probabilities that are zero.

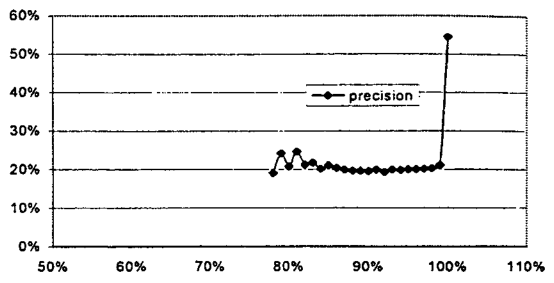

Figure 1: The language model output errors against percentages of zero conditional probabilities.

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}.

In Figure 2, we plot the likelihood of errors identified against the different logarithmic conditional probability values. When the recall increases, unfortunately, the precision drops.

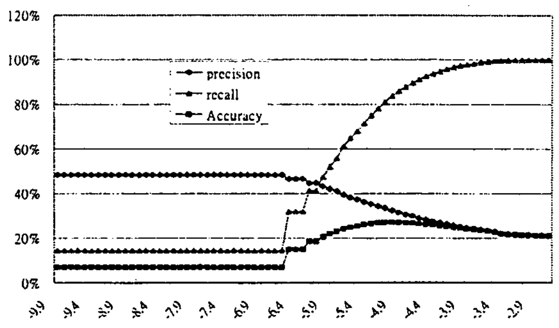

Figure 2: The precision, recall and accuracy (i.e. recall x precision) of detecting language model errors by examining the logarithm conditional probabilities on the maximum likelihood path.

1.2 Language-specific Features

The language-specific features are based on applying the word segmentation algorithm (Kit *et al.*, 1989) to the maximum likelihood path. The ROCLING (Chen and Huang, 1993) word list used for segmentation has 78,000+ entries.

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 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.

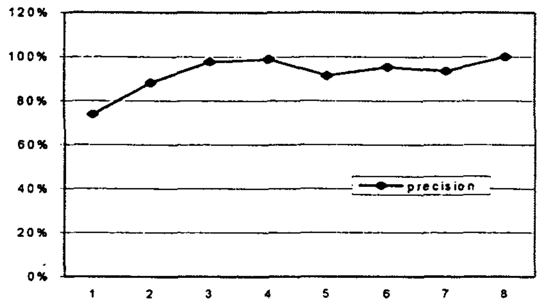

Figure 3: The precision of correct matched words against word lengths.

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

In word segmentation, when there are no entries in the dictionary that can match, the input is segmented into single characters. Thus, Lin *et al* (1993) 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%.

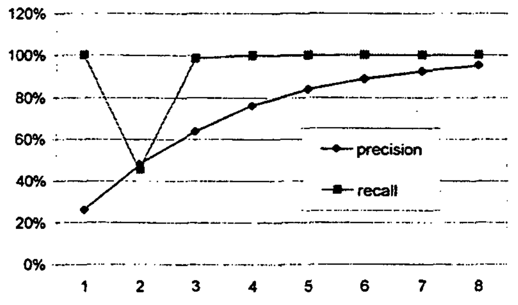

Figure 4: The precision and recall of single-character sequences of different lengths.

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 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.

1.3 Combined use of Features

We carried out a preliminary study using the features mentioned in subsection 1.1 and 1.2. Our Bayesian classifier (Section 2.1) achieved 83% recall but 35% precision, which can be achieved using language specific features only ($F_{2,2}$). Therefore, we try to combine the use of these features in a more careful manner. We divided the detection into 3 scenarios: (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. Single-character sequences longer than 2 are considered as having errors (Figure 4). Words of length longer than 2 are considered correct (Figure 3).

1.3.1 Single characters

After word segmentation, single characters are those cases when there are no multi-character words in the dictionary that can match with it and its following substring. The single characters have different part-of-speech tags.

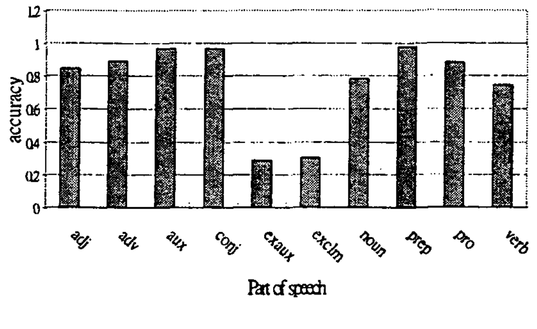

Figure 5: The single characters and their corresponding language model output accuracy for different part-of-speech tags.

Figure 5 shows that the accuracy of the language model for these 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 accuracy of single characters at the beginning (70%) and end (70%) of a sentence is lower than those in the middle (85%) of the sentence. The worst case occurs when the sentence has only a single character, where the measured accuracy is only 8.75% (i.e. no bigram context).

1.3.2 Two-single-characters sequence

Figure 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.

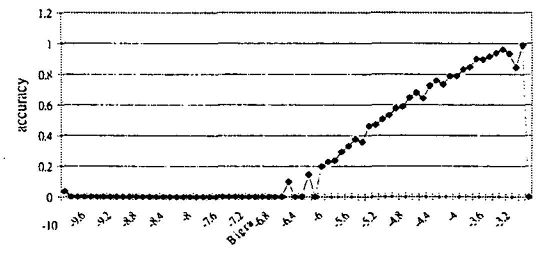

Figure 6: The bigram (logarithm) probability of the single-character sequence of length 2.

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.

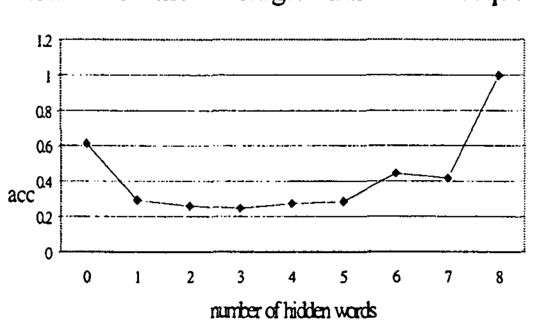

Figure 7: Language model accuracy against different number of hidden words (see text).

Another feature for 2-single-characters 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 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.

1.3.3 Two-character words

For 2 character words, the bigram probability (Figure 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 accuracy is 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 9) decreases as the number of hidden words increase in the corresponding 2 sets of candidate characters.

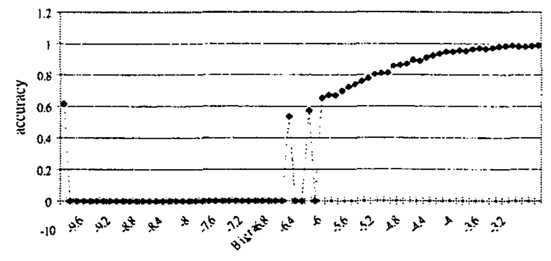

Figure 8: The language model accuracy of 2 character words against the bigram probability.

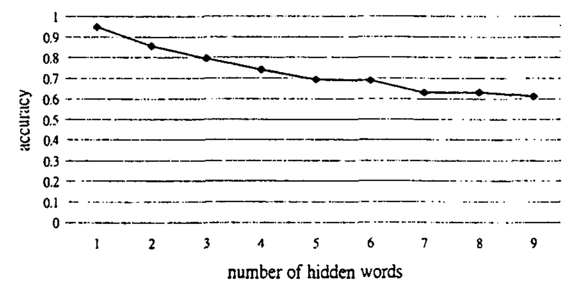

Figure 9: The language model accuracy against different number of hidden words.

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 to decide for error identification because we do not know whether particular features work well with particular classifiers, which make different assumptions about classification. Three types of classifers will be examined: Bayesian, decision tree and neural network.

2.1 Bayesian Classifier

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

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

$$\text{Otherwise} \qquad \text{assign } w_e$$

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.

2.2 Decision Tree

Originally, we tried to use the support vector machine (SVM) (Vanpik, 1995) but it could not converge. Instead, we used the decision tree algorithm C4.5 by Quinlan (1993). 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 Section 1.3.

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 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, 1998).

The output of each node in the MLP is the weighted sum of its input, which is transformed by a sigmoidal 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.

3 Evaluation

In the evaluation, the training data is the PH corpus and the test 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 (Wong and Chan, 1995). For robustness, each recognized character in our evaluation is selected from 10 candidates.

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 (Van Rijsbergen, 1979) but the value for rating the relative importance is subjective.

Table 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.

Cases	*Measure*	*Bayes*	*C4.5*	*MLP*
Single character	Recall	71%	56%	28%
	Precision	40%	75%	71%
2 single characters	Recall	60%	84%	83%
	Precision	88%	82%	80%
2-character words	Recall	60%	17%	9%
	Precision	29%	60%	62%
Overall	Recall	79%	73%	75%
	Precision	60%	81%	80%
	Skip Ratio	65%	76%	66%

Table 1: The performances of the 3 types of classifiers in detecting language model errors.

4 Summary and Future Work

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, suffering 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 aggregated as a single feature vector, the recall and precision of errors are 83% and 35%, respectively, which are the same if we just use language-specific features. Hence, instead of a single classifier, we separated 3 situations identified by the language-specific features and 3 classifiers are used to detect these errors individually. The Bayesian classifier (simpliest) 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 are achieved using decision trees, which are preferred since their skip ratio is higher (i.e. 76%). 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.

Acknowledgement

This work is supported by the (Hong Kong) University Grants Council under project PolyU 5109/97E and the PolyU Central Research Grant G-S603. We are grateful to Guo and Liu for providing the PH corpus and ROCLING for providing their word list.

References

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

Brown, P.F., V.J. Della Pietra, P.V. deSouza and R.L. Mercer (1992) "Class-based n-gram models of natural language", *Computational Linguistics*, **4**, pp.467-479.

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

Chien, L.F., Chen, K.J., Lee, L.S. (1993) "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**, Page(s): 221 – 240.

Elliman, D.G. and I.T. Lancaster (1990) "A review of segmentation and contextual analysis techniques for text recognition", *Pattern Recognition*, **23:3/4**, pp.337-346.

Fujita, O. (1998) "Statistical estimation of the number of hidden units for feedforward neural networks", *Neural Networks*, **11**, 851-859.

Guo, J. 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.

Huang, X., F. Alleva, H. Hon, M. Hwang,, K. Lee and R. Rosenfeld (1993) "The SPHINX-II speech recognition system : an overview", *Computer Speech and Lanaguage*, **2**, 137-148.

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

Jelinek, F. (1989) "Self-organized language modeling for speech recognition", *in Readings in Speech Recognition*, Morgan Kayfmann.

Jelinek, F. (1991) "Up from trigrams", *Proc. Eurospeech 91*, pp.181-184.

Jin, Y., Y. Xia and X. Chang (1995) "Using contextual information to guide Chinese text recognition", *Proc. ICCPOL '95*, pp. 134-139.

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

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

Law, H.H-C. and C. Chan (1996) "N-th order ergodic multigram HMM for modeling of languages without marked word boundaries", *Proc. COLING 96*, pp. 2043-209.

Lee, H-J. and C-H Tang (1995) "A language model based on semantically clustered words in a Chinese character recognition system", *Proc. 3rd Int Conf. on Document Analysis and Recognition, Vol. 1.*, pp. 450-453.

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

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

Mahajan, M., D. Beeferman and X.D. Huang (1999) "Improving topic-dependent modeling using information retrieval techniques", *Proc. IEEE ICASSP 99, Vol. 1*, pp.541-544.

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

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",

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

Quinlan, J.R. (1993) "C4.5 programs for machine learning", Morgan Kaufmann, CA.

Ron, D., Y. Singer and N. Tishby (1994) "The power of Amnesia: learning probabilistic automata with variable memory length", *to appear in Machine Learning*

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

Sun, S.W. (1991), "A Contextual Postprocessing for Optical Chinese Character Recognition", in *Proc.Int. Sym. on Circuits and Systems*, pp. 2641-2644.

Vapnik, V. (1995) *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.

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

Wagner, R.A. and M.J. Fisher (1974) "The string to string correction problem", *J. ACM*, **21:1**, pp. 168-173.

Ward, W. and S. Issar (1996) "A class based language model for speech recognition", *Proc. IEEE ICASSP 96, Vol. 1*, pp.416-418.

Wong, P-K. and C. Chan (1999) "Postprocessing statistical language models for handwritten Chinese character recognizer", *IEEE Trans. SMC, Part B*, **29:2**, 286-291.

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

Yang, K-C., T-H. Ho, L-F. Chien and L-S. Lee (1998) "Statistics-based segment pattern lexicon - a new direction for Chinese language modeling", *Proc. IEEE ICASSP 98, Vol. 1.*, pp.169-172.