A CHARACTER LEVEL PREDICTIVE LANGUAGE MODEL AND ITS APPLICATION TO HANDWRITING RECOGNITION

Homayoon S.M. Beigi and **T. Fujisaki**
T.J. Watson Research Center
International Business Machines
P.O. Box 704
Yorktown Heights, New York 10598

Abstract

One way to increase the accuracy of handwriting recognizers is to take advantage of the context of the text being written. It is a well-known fact that as the best handwriting recognition engines, we humans use a lot of context in reading handwritten text and that our recognition accuracy will dramatically decrease once we are shown individual handwritten chraracters without any context. This paper presents a predictive language model which uses an N-gram for providing it with probability of characters following a sequence of $N-1$ characters. This language is shown to increase the accuracy of the handwriting recognizer close to 4%.

1 Introduction

[1] discusses the development of a character prediction method, based on an N-gram (sequence of N consequent characters in the language). In this method, $N=4$ is shown to be optimal for practical on-line handwriting recognition. This paper will discuss the usage of this prediction method in an on-line handwriting recognizer which will handle Boxed and Run-On writing. [2]

Character prediction could be used in reducing the number of errors in applications such as handwriting recognition. Commonly, in handwriting recognition, once a character is written, its digitized $< x, y >$ coordinates are fed into a shape matcher which will match some features with a set of given prototypes and will return probabilities of each prototype being the intended writing. This is independent of the history of the text being written.

The prediction model of [1] has been established by generating tetra-grams from a businesss correspondance corpus of 320,000,000 words which includes about 270,000 distinct words (excluding punctuations). This model has been shown to produce an average fanout of 5. This means that there are only an average of 5 possible characters which could follow a contextual sequence of three characters. This reduction of average fanout from 27 (a-z and space) to 5, reduces the confusion for the shape matcher and in turn will increase the accuracy. Furthuremore, in this implementation, by reordering the search hypotheses to the favour of highly probable characters, this language model decreases the search error.

The set of characters which are supported in the N-gram character predictor is $a-z$ plus space denoted here as $\#$. The N-gram data-base was made case- insensitive. As an example of the output generated by the character predictor, consider the word *these*. The probabilities available from the character predictor of [1] are $p(t|\#)$, $p(h|\#t)$, $p(e|\#th)$, $p(s|the)$, $p(e|hes)$, and $p(\#|ese)$ could be provided by such a data-base.

The system this predictive model was implemented on is a stroke-based Boxed and Run-On on-line recognition system with elastic matching as its shape matcher.[2] The predictive model works in conjunction with the search engine. The search algorithm used is a special case of the A^* algorithm. The predictive language model affects the search by reordering the search hypotheses and modifying the stroke-matching scores from the elastic matcher.

2 Description

Due to the nature of the elastic-matcher employed and the heuristics involved, the scoring system is not perfectly probabilistic. However, the predictive model is a true probabilistic model. One of the biggest problems is to develop a method for mixing the scores provided by the recognizer and those provided by the predictive model. An array of runs were made using the recognition engine without the predictive language model, to collect statistical data about the heuristic scores and their mapping to probabilities. As a correct answer came back from the recognizer, the recognition scores for the strokes in that word were noted. After finishing the recognition over 24 writers each writing 180 words, the percentages of the successful words were noted with their corresponding recognition scores. These values were then integrated (sumed) to give an accumulated value such that the best possible score would map into a probability of 1. Figure 1 shows a graph mapping recogintion scores to probabilities.

Figure 1: Score-Probability Mapping

The information in figure 1 is used in the form of a look-up table to convert between the recognition system scores and probabilities. Probabilities given by the predictive model have been shown in [1] to be transfered from being probabilities within the space of alphabets to probabilities in the whole character set (including numbers and special characters).

Figure 2 shows the details of the list of hypotheses attached to a search node as it is created. In the abscence of the predictive language model, each search generation (corresponding to a stroke) is provided with a list of stroke hypotheses with stroke-match scores by the matcher. When the predictive language model is used, each node, depending on its language model path could have a different list of stroke hypotheses. Some strokes based on the language model path of the node, are not allowed by the predictive model or they have language model scores which might not compete with their stroke-match scores. Therefore, as in figure 2, there is a list of local stroke hypotheses associated with every new node which is generated through search. This list includes in each entry, an index to its position in the global hypothesis list for that generation. In addition, it includes a score which is a mixture of the language model probability mapped into score and the score given by the basic recognition engine.

Figure 3 shows a flow-chart of the proposed predictive language model. Based on this figure, as a new search node is generated, it is checked for being a last stroke within a character hypothesis. Each stroke is identified as in figure 2, by its position within a character (starting with 0) and the total number of strokes in that chracter. In addition, the stroke has a character prototype I.D. which is unique. If the search node is not the last stroke of a hypothesized prototype, no action is done and the search engine uses the global hypothesis list for continuing the search process.

If the search node is the last stroke of a hypothesized charcater prototype, it is sent to the predictive language model to be provided with a list of language model (LM) hypotheses with their corresponding probabilities of occurance after that search node. These predictive model probabilities are weighed against their context length. Namely, the longer the context at the search node (maximum of 3 characters), the more the predictive model probability is weighed. This provides some smoothing for the effects of the predictive language model.

As the next step, all the stroke hypotheses which have both a poor match score and a poor language model score are killed. At this point the language model probabilities are mapped into scores which the search operates in, through the mapping information available in figure 1.

At this point a new score is created through the mixture of the stroke-match and predictive model scores. This mixture is a weighted sum of the two scores. The weighting factor partly governs the strength of the language model used.

Once these scores are generated, the list of hypotheses is sorted based on the newly generated scores and then the number of strokes in the character prototypes in ascending order of number of strokes. This new list of hypotheses is then used by the search algorithm to generate new children for that node. Table 1 presents a set of numbers in the form of a/b where the a stands for the number of nodes created per generation and b is the average number of stroke hypotheses attached to each node. In table 1, these numbers are given for the case of the basic system and the one containing the predictive model, in two rows. The first row reflects the average value for these numbers over all nodes and all generations and the second row gives the maximum of these numbers over all nodes in a generation, averaged over all generations.

This table shows that the predictive language model greatly reduces the search depth. Actually, in the case of Boxed recognition, the time gain in the reduction of the search space is more than that lost as overhead of using the predictive language model. In the case of Run-on recognition, the overhead of the predictive model seems to be equal to the time lost in the search process when the predictive model is not being used.

3 A Test of the System

The predictive language model was tested on over 12 writers for the Boxed mode and 6 writers for the Run-on mode. These writers were external people who were not familiar with handwriting recognition and the text included about 20% or more non-contextual writing and special symbols. Each writer was asked to write a text of about 1300 characters for the Boxed mode and 180 characters for the Run-On mode. Tests were done with both writer-dependent (trained) and writer-independent (walk-up) prototype sets. Table 2 presents the average values of these accuracies.

4 Conclusion

The predictive language model given in this paper has a very big problem to deal with. This problem is the incompatibility between the scores given through the language model (probabilities) and scores given by the basic recognition engine (Cartesian plus heuristic scores). Another problem was that the recogniton engine was not originally designed to handle this type of a language model.

Despite these problems, the performance of this predictive model is acceptable. It is, however, greatly possible that a different recognizer in abscence of these problems would produce much better results due to the studies provided by [1].This language model has also shown an enhancement to the accuracy of the post-processing error-correction model in the system by providing it with a better base accuracy. [3]

References

[1] Homayoon S.M. Beigi, "Character Prediction for On-Line Handwriting Recognition," Canadian Conference on Electrical and Computer Engineering, IEEE: Toronto, Canada, September 1992.

[2] T. Fujisaki, H.S.M. Beigi, C.C. Tappert, M. Ukelson and C.G.Wolf, "Online Recognitoin of Unconstrained Handprinting: A Stroke-based System and Its Evaluation," _______, S. Impedovo, Editor, ELSEVIER: Italy, 1992.

[3] Homayoon S.M. Beigi, T. Fujisaki, W. Modlin and K. Wenstrup, "A Post-Processing Error-Correction Scheme Using a Dictionary for On-Line Boxed and Run-On Handwriting Recognition," Submitted to the IEEE Canadian Conference on Electrical and Computer Engineering, Toronto, Ontario, Canada, Sep. 1992.

Figure 2: Local Search Hypothesis List

Figure 3: Flow-Chart of the Predictive Language Model

	Basic System	With Prediction
Average	285/34	97/26
Maximum	1001/54	484/50

Table 1: Search Depth Statistics

	Accuracy (No Pred.)	Accuracy (Pred.)
Boxed Walk Up	75.7%	79.6%
Boxed Trained	90.2%	92.1%
Run-on Walk Up	68.2%	70.1%
Run-on Trained	86.7%	87.9%

Table 2: Predictive Language Model