The Effect of Large Training Set Sizes on Online Japanese Kanji and English Cursive Recognizers

Henry A. Rowley Manish Goyal John Bennett
Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, USA
hrowley@microsoft.com mango@microsoft.com jbenn@microsoft.com

Abstract

Much research in handwriting recognition has focused on how to improve recognizers with constrained training set sizes. This paper presents the results of training a nearest-neighbor based online Japanese Kanji recognizer and a neural-network based online cursive English recognizer on a wide range of training set sizes, including sizes not generally available. The experiments demonstrate that increasing the amount of training data improves the accuracy, even when the recognizer's representation power is limited.

1. Introduction

An important question when building a handwriting recognition system is how much training data to collect. Because of limits on available data sets, most researchers have focused on developing algorithms that generalize better from small data sets. This paper looks at the effect of increasing the training set size beyond those generally available for online Japanese Kanji and English cursive recognizers.

The Japanese recognizer uses a nearest-neighbor classification scheme. Each character to be recognized is first converted to a feature vector and its distance to every stored prototype is computed. The prototype labels are the outputs of the system, and the distances are used as scores for each character. Each score is adjusted to take into account such things as the frequency of the character in natural text, and the position at which it was written in the writing box. Training of the recognizer involves choosing the distance metric and the subset of the training samples to be used as prototypes.

The English cursive recognizer uses a Time Delayed Neural Network (TDNN). It can be used to recognize isolated words written either in print or in cursive. The input ink is first segmented and featurized and then fed to the neural net. The neural network outputs are in the form of a sparse matrix of character probabilities. This matrix then goes through a post-processing step which uses a language model to arrive at the final result.

Both of these recognizers were trained with a wide range of training set sizes. Since training a recognizer with a large capacity on a small amount of data can result in overtraining, we also varied the representation power of the recognizers. The results show that increasing the amount of training data increases the accuracy, assuming that the recognizer's representation power is not too severely limited.

We will begin by describing the Japanese recognizer in more detail, followed by the experiments conducted with its training set. We then discuss the English recognizer and its results with different size training sets.

2. Online Japanese Kanji recognizer

The Japanese recognizer used for the experiments in this paper is designed to recognize characters in the JIS-208 character set which are written with three or more strokes. It has three main components: a procedure for converting the input strokes to feature vectors, a distance metric for comparing feature vectors, and a database of prototypes against which the input is compared. Each of these pieces will be described in more detail, followed by descriptions of the experiments.

2.1. Feature Vectors

The strokes of ink are first scaled and shifted horizontally and vertically to fill a fixed square box. The strokes are then smoothed to remove noise from the digitizer, and split at cusps and inflection points. Each resulting stroke fragment is classified into one of nine categories, as illustrated in Figure 1.

Some categories allow the stroke fragments to be written in both directions, while others separate the different writing directions into different categories. The two curved categories allow the fragment to start and end at any location in the writing box, as long as the direction (clockwise or counter-clockwise) matches the category. The last two right-angled categories are special cases of the curves, which only match upper-right and lower-left corners. Each category is further split into two smaller

Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)
0-7695-1692-0/02 $17.00 © 2002 IEEE

categories, based on whether the size of the fragment is larger or smaller than a fixed fraction of the total character size. The stroke smoothing, fragmentation, and categorization are implemented using a hand-built finite state machine, some details of which are described in Reference [3].

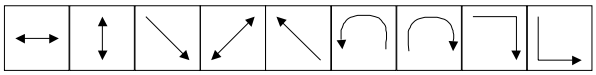

Figure 1. Illustration of the nine main feature categories. For each category shown, there are large and small versions, used when the fragment length is larger or smaller than a fixed fraction of the overall character size.

In addition to the category label, each stroke fragment is also represented by the positions of its start and end points, which are quantized to 16 levels in the horizontal and vertical directions. The fragment categories and start and end points are stored in the order in which they were written, yielding the feature vector used by the rest of the system. Similar sets of features have been used for example by Reference [2].

2.2. Distance Metric

Since we are using a nearest-neighbor classifier, we need a way to measure the closeness of two feature vectors of the type described in the previous section. We will first look at measuring the distance between two fragments.

For the fragment start and end points, we begin by computing the sum of the squared Euclidean distances between the corresponding start and end points of the two fragments. Because of the quantization of the coordinates of the start and end points, there is a small range of values for this distance measure. We then go through the training data, recording the frequency of a particular distance arising from stroke fragments of two instances of the same character relative to the frequency of that distance between any pair of characters. A similar probability table is built up for the categories of pairs of fragments arising from the same character relative to pairs of fragments from any characters. For more details of how to compute these probability tables efficiently, see Reference [6].

These two probabilities are converted to log probabilities and added together (with a tuned weighting factor), then the resulting scores are added for all fragments. This gives the distance measure between two feature vectors. This distance metric can only be computed between samples written with the same number of stroke fragments.

2.3. Prototype Database

The final component of the recognizer is a database of feature vectors, or prototypes, which represent the shapes the recognizer should understand. These vectors are selected from the training data in three main steps. First, the distances between all pairs of samples of a given character are computed. The samples are then ordered by how many times they are the closest to another sample of the same character. Samples with higher counts can be viewed as more representative of other samples than those with lower counts. The second stage goes through all the samples in order, checking to see if they are recognized correctly with the current prototype database (which is initially empty) and adding them to the database if they are not. This may result in overtraining, as large numbers of outliers may be added to the database. The final stage removes prototypes from the database, optimizing for recognizer accuracy while fitting the database into a specified memory budget. Since the running time of the recognizer is roughly proportional to the number of prototypes, this also impacts the recognizer speed.

2.4. Data Collection

The training data we will use for these experiments consists of nearly five million samples of 6847 characters in JIS-208 written with three or more strokes. This data was collected over a period of several years from native Japanese speakers. The data was collected on Wacom tablets and the Fujitsu Stylistic 2300. The collection mainly consists of natural text. Care has been taken to ensure that rare characters also have a sufficient number of samples for training. The data set has been automatically and manually cleaned to ensure that the label for each character matches what was actually written.

2.5. Experiments

With the recognizer and training procedures in hand, we can start to look at experiments with differing sizes of training sets. As a first test, we extracted the 1012 characters from the training set that have 1000 or more samples. We then trained recognizers on varying subsets of this data, from 10 samples per character up to 1000, to see how the accuracy changed. The test sets used for the experiments are separate from the training set. The results are shown in Figure 2 for two test sets, one that approximates the natural frequency distribution (the subset of the natural distribution contained in the 1012 characters selected earlier), and one approximating the uniform distribution. As can be seen, the error rate drops significantly as the amount of training data increases, and

is just beginning to level off at around 1000 samples per character. The uniform error rate is lower than the natural error rate, because the training data is uniformly distributed.

Figure 2. Limited training and test sets to the 1012 characters for which we have 1000 training samples, and trained with varying numbers of samples per character. The natural test set contained 79,747 samples, while the uniform test set contained 35,096 samples.

In the second test, we trained the recognizer to handle the full JIS-208 character set, and varied the upper limit on the number of samples of each character. Since not all characters are equally represented in the training data, some characters will have fewer samples than the limit. The results of this test are shown in Figure 3.

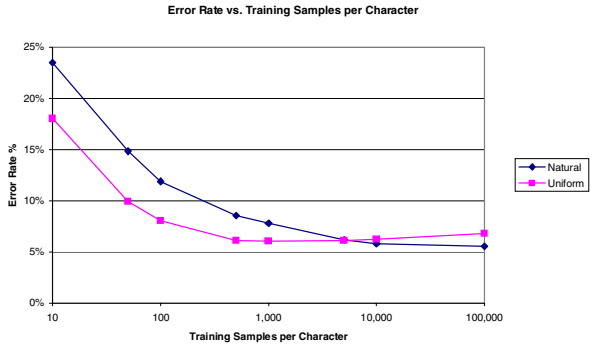

Figure 3. This test used the full JIS-208 character set, with upper bounds on the numbers of samples of each character. The natural test set for this graph contained 85,655 samples, while the uniform test set contained 156,826 samples.

Overall the error rates are higher, because the recognizer now supports 6847 characters instead of 1012. At small numbers of samples per character, the training set is approximately uniformly distributed. However, as the number of samples increases, only the most common characters get more samples added to the training set, so the training data distribution looks more like a natural distribution. This is why initially the uniform test set gives better scores, while the natural test set has better scores at higher numbers of samples per character. In fact, the uniform error rate suffers at higher numbers of samples per character because the recognizer is placing more weight on the common characters.

In the third experiment, we imposed some capacity constraints on the recognizer's prototype database. The results are shown in Figure 4. Each curve in the graph represents prototype databases of a fixed size trained with varying numbers of samples per character, from 10 to 100,000. Database sizes are specified by a memory budget. Each prototype occupies space proportional to the number of stroke fragments it contains. A typical 640KB prototype database contains 21,000 prototypes. The error rates are measured on the natural frequency test set. From this graph we can see that increasing the allowed prototype database size can have a significant effect on the accuracy, decreasing the error rate from 8.25% to 5.55% when using all the training data. The larger effect is that increasing the amount of training data increases the accuracy, even for the smallest prototype database size tested. Increased capacity helps most at the highest numbers of samples per character, where the curves begin to separate.

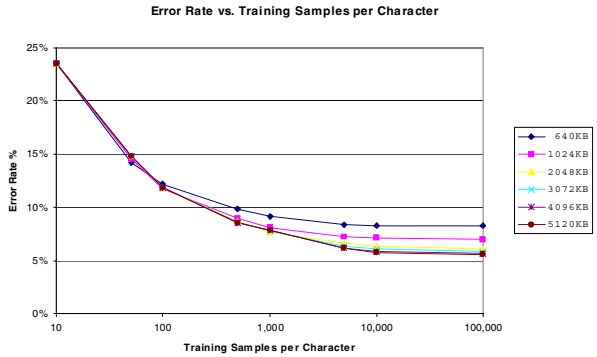

Figure 4. Each curve represents a fixed prototype database size (recognizer capacity), trained with varying numbers of samples per character. While increasing the capacity improves accuracy, increasing the training data is much more helpful. The error rate was measured on a natural frequency test set containing 85,655 samples. Note that the error rates for the largest database sizes almost overlap.

3. Online English Cursive Recognizer

The online English recognizer used for the experiments in this paper is designed to recognize words.

The characters that make up these words are printable ASCII and also include the euro and pound signs. The main components of the recognizer are: a procedure for converting the input strokes to feature vectors, a time delayed neural network, and a post-processing step that involves the use of a language model.

3.1. Feature Vectors

The ink to be recognized is first split into various segments, by cutting the ink at the bottoms of the characters. Segmentation thus takes place where the *y* coordinate reaches a minimum value and starts to move in the other direction. Similar methods for segmentation have been proposed in References [5] and [7].

Each of the segments is then represented in the form of a Chebyshev polynomial. More details on how these polynomials are computed may be found in References [1] and [4]. These feature vectors are then fed as inputs to the neural network.

3.2. The Time Delayed Neural Network

The TDNN used for the recognizer is similar to the one proposed in Reference [7]. The outputs from the network form a sparse matrix of character probabilities that undergo post processing by comparing with a language model, before the final results are obtained.

3.3. Data Collection

A considerable amount of resources were devoted towards collecting the data necessary for making this study possible. Our training set has more than a million words collected from native English speakers. It consists of a mixture of natural text, punctuation, postal addresses, numbers, and email and web addresses. Both print and cursive data are used for training the recognizer. The data set has been randomly sampled into smaller subsets to produce the various data set sizes used for training the different recognizers.

The testing set was collected in a manner similar to that of the training set and consists of 150,495 words (which contain 748,308 characters). The relative weighing of the various sample types in the testing set has been designed to closely approximate the user experience if the user was to use handwriting as the primary method of input to the computer.

3.4. Experiments

The training data for the recognizer is randomly sampled and split up into smaller sizes. We have also used various sizes of neural networks for the experiments and have obtained accuracy numbers for different neural net size against different training set sizes. The results of these experiments are shown in Figure 5.

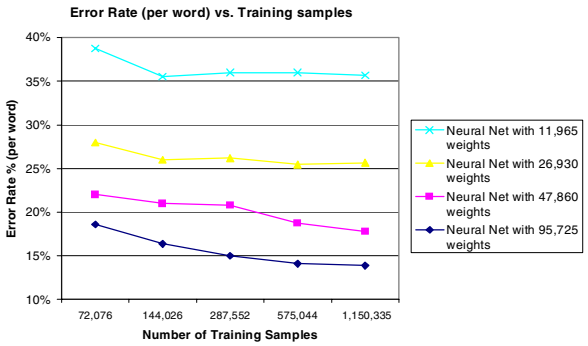

Figure 5. The effect of varying the training set size on the error rate. Each curve is for a fixed neural network size. We see that the error rate drops as we increase the amount of training data.

As can be seen in the above graph, the error rate decreases as the number of training samples increases. Moreover it is seen that the effect of adding more data is more pronounced as the size of the neural network increases. When the network size is small the extra amount of data does not make much of a difference, but as the network size is increased the amount of training data begins to make a significant impact. It also follows from the above figure that for the same neural network size, while increasing the amount of training data increases the accuracy, the accuracy gains might not be very high unless the complexity of the network itself is increased.

4. Conclusions

This paper has presented the results of varying training set sizes over a wide range for two different types of recognizers, a Japanese Kanji recognizer based on a nearest-neighbor classifier, and an English cursive recognizer based on a neural network. Comparing Figure 4 and Figure 5, we can see that the training set size had a much larger impact in the nearest-neighbor classifier. This is because the classifier takes its prototypes directly from the training samples, with no smoothing or generalization to produce better prototypes, while the neural network is better able to generalize from a smaller training set. We can also see that neither recognizer has stopped improving even with the large training sets we used, and that more data, possibly using a recognizer with greater representational power, will improve the accuracy further.

5. Acknowledgements

The authors would like to thank Ahmad Abdulkader, Angshuman Guha, Patrick Haluptzok, Greg Hullender, Jay Pittman, Michael Revow, and Petr Slavik for comments and suggestions on this paper.

6. References

[1] Adcock, James L. "Method and system for modeling handwriting using polynomials as a function of time", US Patent 5,764,797, granted June 9, 1998.

[2] Chou, Sheng-Lin and Tsai, Wen-Hsiang. "Recognizing Handwritten Chinese Characters by Stroke-Segment Matching Using an Iteration Scheme", in *Character and Handwriting Recognition: Expanding Frontiers*, copyright 1991, pages 175-197.

[3] Dai, Xiwei. "Handwritten Symbol Recognizer", US Patent 5,729,629, granted March 17, 1998.

[4] Guha, Angshuman. "A Uniform Compact Representation for Variable Size Ink", US Patent pending, 1998.

[5] Hollerbach, John M. "An Oscillation Theory of Handwriting", in *Biological Cybernetics*, copyright 1981, pages 139-156

[6] Hullender, Gregory N. "Automatic Generation of Handwriting Recognition Crossing Tables", US Patent 6,094,506, granted July 25, 2000.

[7] Rumelhart, David E. "Theory to Practice: A Case Study-Recognizing Cursive Handwriting", in *Computational Learning and Cognition*, Proceedings of the Third NEC Research Symposium, copyright 1992, pages 177-196.