Distinguishing Text from Graphics in On-line Handwritten Ink*

Christopher M. Bishop, Markus Svensén
Microsoft Research
7 J J Thomson Avenue
Cambridge CB3 0FB, UK
{cmbishop,markussv}@microsoft.com

Geoffrey E. Hinton
Department of Computer Science
University of Toronto
King's College Road, Toronto
Ontario, M5S 3G4, CANADA
hinton@cs.toronto.edu

Abstract

We present a system that separates text from graphics strokes in handwritten digital ink. It utilizes not just the characteristics of the strokes, but also the information provided by the gaps between the strokes, as well as the temporal characteristics of the stroke sequence. It is built using machine learning techniques that infer the internal parameters of the system from real digital ink, collected using a Tablet PC.

1 Introduction

Pen controlled computing devices such as PDAs and the Tablet PC are becoming increasingly widespread, and the digital ink captured by such devices offers many potential advantages compared to traditional pen and paper. However, in order to realize many of these advantages it is essential for the device to separate the ink strokes so that text strokes can be sent to a recognition engine, and graphics strokes can be grouped and recognized as higher level graphical entities. Such analysis is essential even if the ink is to be displayed to the user in its original form since search and intelligent editing require a high degree of interpretation of the ink.

In this paper, we consider the fundamental problem of classifying strokes of digital ink as either text or non-text (which we shall refer to as 'graphics'). An example of a page of ink is shown in Figure 1. Features extracted from an individual stroke provide some relevant information regarding the identity of the strokes, and can already give a reasonable separation of the two classes [2]. However,

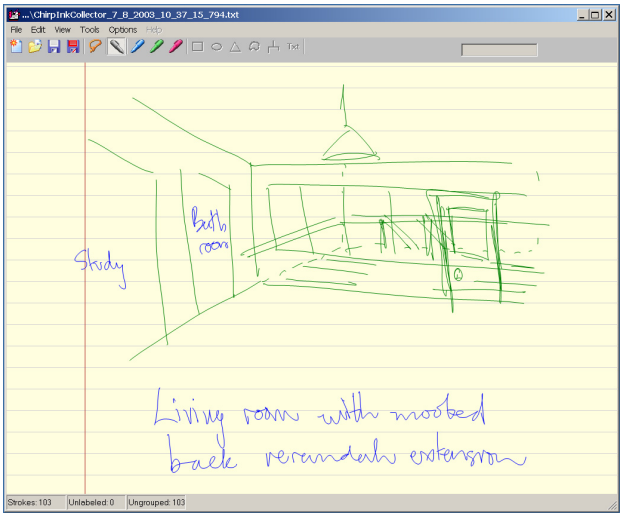

Figure 1. A screen shot of the purpose built software used to collect and label data, together with a page of ink showing the text strokes (blue) and graphics strokes (green) resulting from manual labelling.

to achieve improved performance we need to take into account the context of the stroke [5]. Dealing with spatial context leads to a two-dimensional problem which can easily become computationally intensive. Since low level parsing must be performed rapidly, we consider a different approach which exploits the temporal information associated with on-line ink, leading to a one-dimensional optimization problem.

Our approach is based on the use of discriminative machine learning techniques to infer the class of each stroke on the page (text or graphics) given the observed ink. By

*Accepted for oral prentation at the ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR), October 26–29, 2004, Tokyo, Japan.

adopting a probabilistic framework we can fuse evidence from local stroke-based features with that arising from the temporal context in a principled manner. Furthermore, computing probabilities is significantly more powerful than making hard classifications [1] since the resulting probabilities can be passed to later stages of processing, thereby allowing classification to be made in a decision-theoretic optimal manner in the context of the overall system. In this paper we shall use this probabilistic output to present our results in terms of ROC (receiver operating characteristic) curves which display a range of trade-offs between misclassifying text strokes and misclassifying graphics strokes.

2 The Proposed System

Here we shall discuss three approaches of successively increasing complexity. We start in Section 2.1 by considering each stroke in isolation, extracting a set of features and then training a probabilistic classifier based on a feed-forward neural network, also known as a multi-layer perceptron or MLP [1]. In Section 2.2 we then augment this model by including temporal information to capture the correlation between successive class labels. Finally in Section 2.3 we further extend the model to also consider information extracted from the gaps between successive strokes.

2.1 Independent Stroke Model

The input data for our system consists of sequences of strokes, separated by gaps. A stroke is a sequence of *points* (x-y-coordinates) recorded between a pen-down event (when pen was put into contact with the screen) and a pen-up event (when the pen was lifted from the screen). We will refer to the line between two consecutive points in stroke as a *segment*. In addition to the spatial data, each stroke has a time stamp indicating the pen-down time, and so we have access to the temporal ordering of the strokes. Potentially, we could also use the temporal information within each stroke, but since the temporal interval between two points may vary between platforms and may also be affected by system latencies during periods of high system load, we choose not to use this information. In the experiments descibed in Section 3, we will consider a page of ink, which may contain anything from just a few strokes to several hundred.

For each stroke, a total of eleven real-valued features were extracted. The feature extraction itself consists of several steps. After extraction of features available directly from the stroke data (e.g., stroke arc length), a total least squares (TLS) model was fitted to the stroke; this is equivalent to applying principal component analysis to the set of stroke points, and primarily extracts the direction and the length-width ratio of the stroke. Subsequently, the stroke was divided into *fragments* at points corresponding to local maxima in the stroke curvature and TLS was applied again to the largest resulting fragment. This procedure resulted in the following set of features:

1. The stroke arc length, i.e., the sum of the lengths of the stroke segments.
2. The total absolute curvature, defined as the sum of the absolute angles between the consecutive segments.
3. The main direction (x- and y-components) of the stroke, as given by the TLS fit.
4. The eigenvalue (length-width) ratio of the TLS fit of the stroke.
5. The total number of fragments found in the stroke.
6. The arc length of the largest fragment of the stroke.
7. The total absolute curvature of the largest fragment.
8. The main direction of the largest fragment.
9. The length of the long side of the bounding rectangle (not axis-aligned) of the largest fragment.

Features 1, 6 and 9 are likely to be affected by the overall scale of the text or sketches on the page and so we normalize them on per page basis, by scaling them with the inverse of the median fragment length. The directional features, 3 and 8, are transformed to the auxiliary features

$$u = \sin(\theta), \qquad v = \cos(\theta),$$

where

$$\theta = 2\arctan\left(y/|x|\right).$$

This removes symmetries around the origin and ensures that the two extremes (corresponding to angles $-\pi/2$ and $\pi/2$) map to identical feature values. The use of features 6–9 is motivated by the assumption that if the largest fragment is indeed large—it may include the entire stroke—and has a high length-to-width TLS ratio, this is an indicator that the stroke is a graphics stroke.

We denote the complete feature vector by $\mathbf{x}$. The training data set then consists of a set of N ordered strokes with feature vectors $\mathbf{x}_n$, where $n = 1, \ldots, N$ and class labels $t_n \in \{0, 1\}$ where $t_n = 1$ denotes a text stroke and $t_n = 0$ denotes graphics. For the classification of independent strokes we trained multilayer perceptron (MLP) models [1] using the scaled conjugate gradients optimization algorithm [1, 3]. The output $y_n = y(\mathbf{x}_n)$ of the resulting model represents the probability of a stroke being text given the feature vector $\mathbf{x}_n$. The probability distribution of t_n is then given by $p(t_n|\mathbf{x}_n) = y_n^{t_n}(1 - y_n)^{1-t_n}$.

We need to control the model complexity of the MLP, in order to obtain a model which flexible enough to capture

relevant correlations in the data, yet sufficiently constrained enough to avoid over-fitting [1]. We do so by controlling the number of hidden units in the MLP, which we determine using ten-fold cross-validation [6, 1]. The MLP models were constructed using the Netlab toolbox [3].

Typical digital ink data poses the problem that the class distribution is often strongly biased towards text. In practice such skewed class distributions can cause complications in the training of parametric models, resulting in poor performance on the under-represented class (graphics in this application). We can deal with this problem by adjusting the objective function used for fitting the model. The normal objective function for classification is the cross-entropy error, which in the binary case is defined as

$$E = -\sum_{n=1}^{N} \left(t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \right).$$

Minimizing this error function corresponds to maximizing the log likelihood function. We can modify this error function to give

$$\tilde{E} = -\sum_{n=1}^{N} \left(\frac{1}{\pi_{\mathrm{t}}} t_n \ln y_n + \frac{1}{\pi_{\mathrm{g}}} (1 - t_n) \ln(1 - y_n) \right), \quad (1)$$

where π_{t} and π_{g} are the estimated (from the training data) a-priori probabilities of text and graphics, respectively, in the stroke population from which the data were drawn. This scaling corresponds to a balanced data set, and is compensated for when the trained model is used for prediction using Bayes' theorem so that

$$\tilde{y}_n = \frac{\pi_{\mathrm{t}} y_n}{\pi_{\mathrm{t}} y_n + \pi_{\mathrm{g}} (1 - y_n)},$$

where $\tilde{y}_n$ denotes the corrected prediction and represents the posterior probability that the particular stroke is text in the context of the real-world imbalanced priors.

2.2 Hidden Markov Model

While the features of individual strokes treated in isolation can give useful separation of text from graphics, we would expect to achieve much better performance if we take account of the context for each stroke provided by other ink on the page. For on-line ink (as opposed to off-line images) we have both temporal and spatial context. Here we focus on the use of temporal context since this leads to a one-dimensional inference problem which can be solved efficiently using dynamic programming techniques.

The intuition is that the identity of successive strokes will tend to be correlated, since a user will typically make several graphics strokes in succession in order to draw a diagram, or will make multiple text strokes in succession while writing a line of text. This is described by the transition probability $p(t_n|t_{n-1})$. Given a training set comprising pages of ink in which each stroke has been labelled as text or graphics, we find this transition probability simply by measuring the frequencies of text and graphics strokes given the label of the previous stroke. Results from the training data set described in Section 3 are shown in Table 1. As can be seen from this table, there is indeed a strong correlation between the labels of successive strokes. We also have a marginal distribution for the first stroke, which for this data set corresponds to $p(t_1 = 1) = 0.5467$.

	$t_{n-1} = 1$	$t_{n-1} = 0$
$t_n = 1$	0.9530	0.1638
$t_n = 0$	0.0470	0.8362

Table 1. Transition matrix for stroke sequences, in which the first and second rows denote respectively the probability of a stroke being text ($t_n = 1$) or graphics ($t_n = 0$), given only the class of the previous stroke (t_{n-1}).

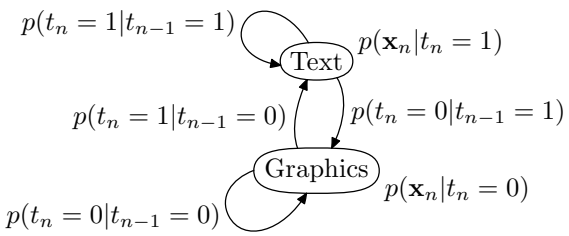

Figure 2. The uni-partite HMM described in Section 2.2. The edges of the graph have been labelled with the corresponding transition probabilities, given in Table 1. Each of the states has an associated emission probability distribution over stroke features, denoted $p(\mathbf{x}_n|\cdot)$.

We now have two sources of information regarding the identity of the strokes namely the predictive distribution $p(t_n|\mathbf{x}_n)$ of the discriminative model as described in Section 2.1, and the transition probability $p(t_n|t_{n-1})$. These have been 'learned' separately from a trained data set, and our goal is now to combine these two sources of probabilistic information in order to arrive at an overall posterior probability for the class label. Since we are looking at the conditional probability of a stroke label given only the previous stroke label, we are implicitly considering a first order Markov process over the labels, as illustrated in Figure 2.

We can therefore construct a hidden Markov model (HMM) to represent a whole sequence of strokes, which

corresponds to a particular factorization of the joint distribution of feature vectors and labels of the form [4]

$$p(t_1,\ldots,t_N,\mathbf{x}_1,\ldots,\mathbf{x}_N) = p(t_1)\prod_{n=2}^{N} p(t_n|t_{n-1})\left[\prod_{n=1}^{N} p(\mathbf{x}_n|t_n)\right]. \quad (2)$$

Once we have constructed the HMM we can find the most probable sequence of stroke labels by running the Viterbi algorithm [4] which is a dynamic programming technique whose cost is linear in the number of strokes. It efficiently solves the optimization problem

$$\arg\max_{t_1,\ldots,t_N} p(t_1,\ldots,t_N,\mathbf{x}_1,\ldots,\mathbf{x}_N) = \arg\max_{t_1,\ldots,t_N} p(t_1,\ldots,t_N|\mathbf{x}_1,\ldots,\mathbf{x}_N) \quad (3)$$

where the equivalence between the left and right sides of this equation comes from omitting the factor $p(\mathbf{x}_1,\ldots,\mathbf{x}_N)$ which is independent of the stroke labels $t_1,\ldots,t_N$.

Note, however, that the HMM requires the conditional densities $p(\mathbf{x}_n|t_n)$ whereas our discriminative models of Section 2.1 provide the posterior probability $p(t_n|\mathbf{x}_n)$. These two conditional distributions are related through Bayes' theorem

$$p(\mathbf{x}_n|t_n) = \frac{p(t_n|\mathbf{x}_n)p(\mathbf{x}_n)}{p(t_n)}. \quad (4)$$

Substituting (4) into (2), and omitting factors which are independent of the $\{t_n\}$, we obtain

$$p(t_1,\ldots,t_N,\mathbf{x}_1,\ldots,\mathbf{x}_N) \propto p(t_1)\prod_{n=2}^{N} p(t_n|t_{n-1})\left[\prod_{n=1}^{N} \frac{p(t_n|\mathbf{x}_n)}{p(t_n)}\right]. \quad (5)$$

Thus we can make use of the predictions from the MLP model of Section 2.1 by simply scaling the predictions by the marginal class probabilities.

Applied this way, the Viterbi algorithm yields the most likely sequence of states given an observed sequence of strokes. However, by re-weighting the (improper) emission probabilities in (5), we can trade-off a better performance on text for a worse performance on graphics and vice versa, allowing us to plot the full ROC curves in Section 3.

2.3 Bi-partite HMM

Stroke sequences from pen controlled devices also contain a further source of information, in a perhaps less obvious form, namely the gaps between the strokes. For example, we might expect that the gap between two consecutive text strokes has characteristics different to those of a gap between a text stroke and a graphics stroke. We now show how to incorporate gap information by extending the approach of the previous section.

For the gaps, as for the strokes, we first of all extract a set of features, $\mathbf{z}$. Note that we typically are not able to capture the pen position between strokes, and so the raw information comprises only the coordinates of the pen up event and the following pen down event, together with the pen-down times at the start of successive strokes. From this information we extract a set of 5 features, consisting of:

1. the logarithm of the difference of the pen-down times for the surrounding strokes,
2. the x- and y-differences of the pen-down locations for the surrounding strokes and
3. the x- and y-differences of the pen-up location of the preceding stroke and the pen-down location of the following stroke.

Features 2 and 3 are normalized on per page basis, by scaling them with the inverse of the median fragment length. As a variant of feature 1, we also tried to use the logarithm of the difference of the pen-up time of the preceeding stroke and the pen-down time of the following stroke. However, this did not yield better performance and since the pen-up time was only available for some of our data (the Cambridge data; see Section 3), we decided not to use this variant.

There are potentially four possible labels associated with the gaps corresponding to the transition *text* → *text*, *graphics* → *graphics*, *text* → *graphics* and *graphics* → *text*. We found that the performance of the system improved if we grouped these into two classes corresponding to whether the successive strokes had the same class label ($t_n = t_{n+1}$) or different labels ($t_n \neq t_{n+1}$).

Using the gap features and labels, we trained MLPs following the procedures described in Section 2.1. The resulting models were integrated with bi-partite HMM shown in Figure 3 in the same way the stroke classification models were integrated with the uni-partite HMM in Section 2.2.

3 Experiments and Results

Data were collected among the employees at Microsoft Research in Cambridge, using a purpose-written piece of software, of which a screen shot is shown in Figure 1. Subjects were instructed to use both text and graphics on the pages they created. The strokes in the data were labelled by one person (again using the same software) and the labelling was checked by another person. In total, 21824 strokes were collected from 41 subjects. The data were divided into a training set (10944 strokes, 19 subjects) and a test set (10830 strokes, 22 subjects), in such a way that data from a single subject did not end up in both training

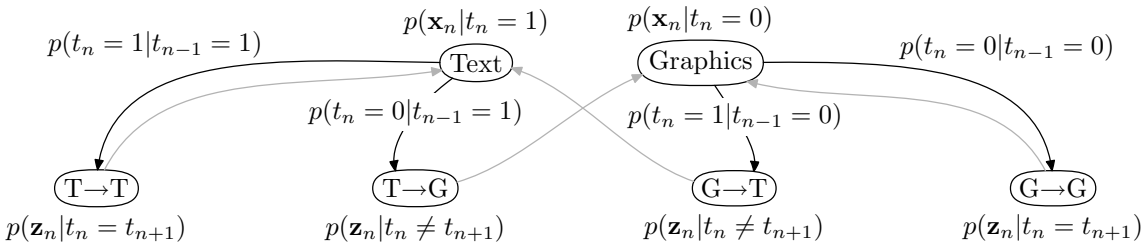

Figure 3. The the bi-partite HMM described in Section 2.3. Here, the edges that go from the stroke states (upper row) to the gap states (lower row) have been labelled with the corresponding transition probabilities, given in Table 1. Since there is only one edge going out of each gap state, these edges (grey) all have associated transition probability of 1 and have therefore not been labelled. As in Figure 2, all states have associated emission probabilities, but note that these are shared across pairs for the gap states.

and test set. Additional, independent test data were also obtained from the Tablet PC Ink Parsing Team at Microsoft in Redmond, USA.

The performance of the independent classifiers and the uni-partite and bi-partite HMM models were assessed using the two test sets available. The results for the Cambridge test set are shown by the ROC-curves in the top left half of Figure 4 and the confusion matrices in the upper two rows of Table 2.

The ROC-curves were created by effectively varying the cost of misclassifying text, while keeping the cost for misclassifying graphics constant and positive. If the former cost is zero, all strokes will be classified as graphics, which corresponds to the bottom-left corner of the Camridge data plot (top-right of the Redmond data plot). If, on the other hand, the cost is sufficiently high, all strokes will instead be classified as text (top-right and bottom-left corners, respectively). The curve of the ideal model would follow the left and top (right and bottom) axes. The confusion matrices all show the true class along the rows and the predicted class along the columns.

The characteristics of the Redmond data were clearly different to those of the Cambridge data, on which the models had been trained. Just the fractions of text and graphics (93.9% and 6.1%, respectively) differed significantly from those of the training data (77.5% and 22.5%, respectively). In the light of this, the results shown in the bottom right half of the plot in Figure 4 and the bottom two rows of Table 2 are encouraging, although the the bi-partite HMM actually gave poorer results than the uni-partite HMM. The confusion matrices corresponds to the text weightings that gave the best performance (lowest misclassification rate) on these data.

Rather than using the Viterbi algorithm for the sequence classification, we could have picked, for each stroke in the sequence, the most likely class [4]. We did evaluate this

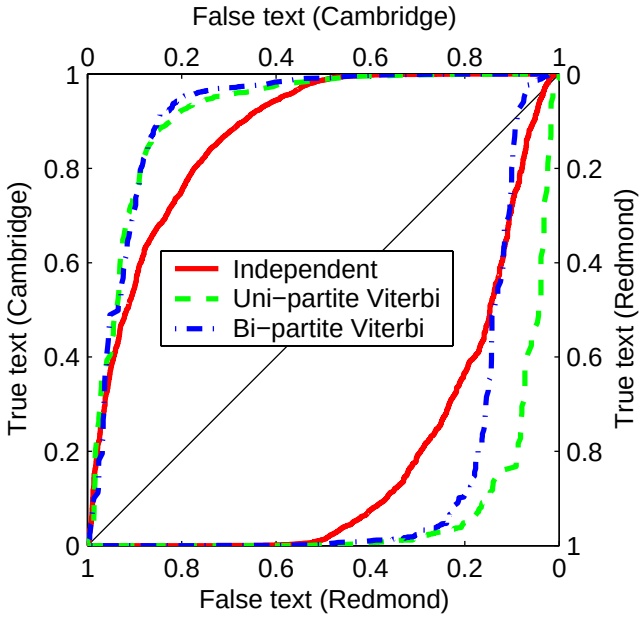

Figure 4. The top left plot shows ROC-curves for the Cambridge test set, plotted in the coordinates of the top left axes. The line style indicate the model used, as detailed by the central legend. The bottom right plot shows corresponding ROC-curves using the Redmond test set, plotted in the coordinates of the bottom right axes; note that these axes have there directions reversed relative to the top left axes.

	Independent		Uni-partite HMM		Bi-partite HMM	
	Text	Graphics	Text	Graphics	Text	Graphics
Text	7857	291	7629	519	7803	345
Graphics	1181	1501	605	2077	577	2105
Text	21421	52	21395	78	21211	262
Graphics	792	592	598	786	489	895

Table 2. Confusion matrices showing the results for the two test sets. The upper pair of rows correspond to the Cambridge data set whereas the lower pair correspond to the Redmond data. In each of the six matrices, the rows correspond to the true class and the columns to the predicted classes of the respective system variant.

approach using the Cambridge test data, but it was consistently outperformed by the Viterbi classification.

4 Conclusion and Discussion

We have presented a system for classifying sequences of digital ink strokes, where the individual strokes are classified as either text or graphics. The system exploits features of the gaps between the strokes as well as features of the strokes themselves, and combines these two sources of information with a temporal model for stroke sequences. This temporal model encodes the fact that strokes of one kind (text or graphics) are more likely to be followed by another stroke of the same kind rather than a stroke of the opposite kind. Our experiments demonstrate that the use of temporal context clearly improves performance compared to classification of individual strokes. The results regarding the use of gap information are less clear, but suggest that this can lead to better performance, provided the model is trained using representative training data.

Although we do exploit the temporal information in the data, our current method can most probably be refined. At the moment, we ignore the length of temporal gaps and treat all strokes from one page of ink as a single sequence. It seems reasonable to assume that after longer temporal gaps, the last stroke before the gap will offer little guidance on what the next stroke might be. In this case, it might be beneficial to cut the stroke sequence and run the Viterbi algorithm on the sub-sequences separately. Preliminary results suggest this is indeed the case.

A related and perhaps important issue is that of copied-and-pasted ink. If we assume that ink strokes retain their time stamps when copied and pasted, this means that the time stamps of such strokes will typically be fairly different to those of the target ink document, in which case it also clearly makes sense to cut the sequence at the resulting large temporal gap. The same principle applies when existing ink documents are edited, where added strokes can be significantly younger than original strokes.

Here we chose to classify the stroke sequences using the temporal context only. However, one could imagine deferring this decision to a stage where more information gathered, such as spatial context. Provided such a context model has a probabilistic formulation, integrating it with the current system should, at least theoretically, be straightforward.

Acknowledgements

The authors would like to thank M. Ye and S. Raghupathy of the Tablet PC Ink Parsing Team at Microsoft Corp., Redmond, for useful discussions and for providing Matlab code for the feature extraction step. They also thanks M. Szummer, M. Gangnet, P. Simard, M. Shilman and P. Viola at Microsoft Research for numerous useful discussions.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[2] A. K. Jain, A. M. Namboodiri, and J. Subrahmonia. Structure in on-line documents. In *ICDAR-6*, pages 844–848. IEEE, 2001.

[3] I. T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer-Verlag, 2001. http://www.ncrg.aston.ac.uk/netlab/.

[4] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[5] M. Shilman, Z. Wei, S. Raghupathy, P. Simard, and D. Jones. Discerning structure from freeform handwritten notes. In *ICDAR-7*. IEEE, 2003.

[6] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, B*, 36(1):111–147, 1974.