SMC '95 final version

An HMM-Based Legal Amount Field OCR System for Checks

András Kornai[1], K.M. Mohiuddin[1], Scott D. Connell[2]
[1]IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
[2]Department of Computer Science
Michigan State University
East Lansing, MI 48824

Abstract

The system described in this paper applies Hidden Markov technology to the task of recognizing the handwritten legal amount *on personal checks. We argue that the most significant source of error in handwriting recognition is the segmentation process. In traditional handwriting OCR systems, recognition is performed at the character level, using the output of an independent segmentation step. Using a fixed stepsize series of vertical slices from the image, the HMM system described in this paper avoids taking segmentation decisions early in the recognition process.*

0 Introduction

The current generation of Optical Character Recognition (OCR) systems can be characterized as a pipeline composed of *Preprocessing, Segmentation, Classification,* and *Identification* stages. None of these stages are immune to error. Preprocessing may fail to remove existing noise, it may remove portions of the image or add noise by some other mechanism. Segmentation may fail to establish a boundary where there should be one (joining error), it may mistakenly introduce a boundary where there should not be one (splitting error), it may ascribe the wrong coordinates to a boundary (misalignment error), or display any combination of these errors over multisegment stretches of text. Classification may be mistaken (substitution error) or may provide no output at all (rejection error). Identification of significant units (words, phrases, etc.) may fail because of low quality character-level input or because of inadequacies in the system dictionary or context model. In addition to misidentification here we must also consider the interpolation of spurious material as well as outright rejection (no output).

From an engineering standpoint, the main problem with the pipeline architecture is the propagation of segmentation errors. The segmentation stage, which is notoriously sensitive to the quality of the image, has to make decisions about the location of segment boundaries early on, with little or no help from later stages. But once a split, join, or misalignment error is present in the output of the segmentation stage, even otherwise perfect classifiers will generally fail, and recovery at the identification stage, normally geared toward recovery from simple substitution errors, is made harder by the spurious insertions and deletions created by split or join errors. The problem cannot be solved by passing alternative segmentation hypotheses to the classifier because the number of such hypotheses grows exponentially in the length of the input.

Because the development of reliable classifiers requires considerable engineering effort, and is still not an entirely solved problem, most commercial efforts in OCR concentrate on machine print, forms with pre-set character boxes, or discrete handwriting styles where the segmentation problem is less acute. But with the development of highly reliable machine print classifiers and with the emergence of pen-based input, in the last few years character segmentation became the weakest point of both image-based and stroke-based recognition systems. Though most of the experiments presented here have used pre-segmented data, the system described in this paper is designed to work on cursive material, namely the *legal amount* field of personal checks, with no help from the numbers appearing in the *courtesy amount.* Section 1 describes the raw input to the system provided by the scanner and the preprocessing steps taken to isolate the handwritten text in the legal amount field. Section 2 describes the feature extraction and dimension reduction processes. Section 3 discusses the impact of various architectures on the performance of HMMs. Word-level segmentation is discussed in Section 4.

1 Preprocessing the check images

The system's input are 240 dpi bilevel images having the dimensions (6 by 2.7 inches, 1440 by 648 pixels) of the standard US personal check. These images come from a commercial check processing product, a high speed/high volume system which currently uses the numerically written *courtesy* amount field for OCR. Because of the high speed, image quality is uneven and some of the images are quite skewed.

After loading the image, first skew is estimated on the basis of beginning- and endpoints of horizontal black pixel runs. Only those images showing significant skew (currently defined as 1% or more) are corrected. Since rotating or shearing the image would be very expensive, the correction is *passive* in the sense that it is the subsequent algorithms that compensate for the skew rather than actively changing the bitmap.

Next the baseline for the legal amount and the baseline for the check recipient, called the **legal** and **pay_to** baselines, are established on the basis of row projections along the skew angle. First, blackness peaks of the row projections are formed by joining adjacent rows that meet various empirical criteria for overall blackness, peak width, and peak shape. Next the peaks are evaluated both for overall blackness content and vertical position, and the topmost one with sufficient blackness within a pre-defined vertical range is selected as the legal line. Finally, the pay_to line is located by the same method in a vertical range defined relative to the pay_to line.

Unlike typical forms applications, where the location of the fields of interest relative to some registration points are known in advance, US personal checks show considerable variation in the location of the legal amount field, which complicates baseline determination. In our experience, simpler algorithms based e.g. on horizontal black runs have a lesser chance of finding the correct pay_to and legal lines because of their high sensitivity to image quality in general and skew in particular.

We found that the handwriting on checks almost always sits on the legal line, so establishing a separate virtual baseline for the writing was not necessary. But in order to capture descenders, which are very common in the cursive handwriting style commonly used for legal amount, the legal field is extended below the legal baseline by half of the distance between the pay_to and the legal lines. Thus, at the end of the preprocessing stage, the legal line, which is the actual baseline for the handwriting, always appears at two-thirds of the total height of the field. The pay_to line is removed, but the baseline is left in place.

2 Feature extraction and dimension reduction

The preprocessing steps described so far would be necessary for every OCR system that attempts to output a single numerical value as determined by the handwritten text (legal amount field) on personal checks. Traditional (as opposed to Hidden Markov) systems would also employ moment normalization [4] or similar steps which somewhat blur the line between preprocessing and feature extraction. We currently perform only one such step, slant detection and normalization. The dominant near-vertical direction of the writing is found by a modified Hough-transform, and again normalization is performed passively.

Assuming the legal amount is located within a strip of (fixed) height H and (indefinite) width W, a sliding window of height h, width w, and slant k is used to sample the image with stepsize s. In some experiments, w and s were fixed at 16 and 8 pixels respectively, so the successive windows overlap by 50%, and the average character image is sampled 3-4 times, but in most experiments, w and s are both 1 so that the typical character in check images is sampled over 30 times. In the ETL and NIST data sets discussed shortly, each character is sampled 16 times.

In most experiments height h is normalized to 24 pixels by means of various subsampling procedures applied to each column within a character, word, or field bounding box. In linear subsampling, every c horizontal lines are replaced by a single horizontal line which has a black pixel wherever any of the original lines had black. In nonlinear subsampling, c increases with the distance from the regions of greatest interest, so that descenders and ascenders get squeezed into fewer lines than strokes within the central region. In on-going experiments global height normalization (adjusting h only once per legal amount field) is replaced by a local height normalization process which places height, together with displacement from the baseline, in a separate codebook [7]. So far, the advantages of this technique have been demonstrated only on isolated character (NIST) data, where it provides consistent, but modest improvements in recognition rate – its effects on actual check data remain to be determined.

We baselined against two standard techniques, Learning Vector Quantization [10] and Multi-Layer Perceptrons [12] to two data sets that were extensively used in later tests. The first of these was extracted from the Electrotechnics Laboratory of Japan (ETL) CD-ROM, and comprises one thousand 64 by 63 pixel black and white images per uppercase letter for training and one hundred images per character for testing. These images were selected from a larger set by removing ∼10% for which no human decision could be made on the basis of the image, so the results are not truly indicative of actual system performance. The second was extracted from the NIST CD-ROMs and contains two thousand training and one hundred testing images (per character) downsampled to 16 by 24 bilevel images of the 16 lowercase letters *(e f g h i l n o r s t u v w x y)* that appear in the words for the numerals

1-99. Here no images were removed, so the training and test sets approximate live data more closely.

algorithm	tr_set/size	feat_dim	% correct
LVQ	ETL/500	50	78.04
LVQ	ETL/500	72	65.48
LVQ	ETL/500	88	65.96
LVQ	ETL/1000	50	79.23
LVQ	ETL/1000	72	66.50
LVQ	ETL/1000	88	66.73
MLP	ETL/2000	88	99.01
MLP	NIST/1000	88	92.50
MLP	NIST/2000	88	93.81

Table 1: Reco rates of standard algorithms

Though neither LVQ nor MLP are incorporated in the HMM system described here, the figures in Table 1 (MLP data courtesy of Jianchang Mao) are indicative of the complexity of the task and can serve as a baseline in assessing the impact of the various feature extraction, data reduction, and HMM topologies used in the HMM system proper. One feature set used in the experiments is the 88 *Contour Direction Features* (see [15]) – we will refer to this as the CDF/88 set. The CDF/72 set was obtained by omitting 16 features corresponding to the four corners of the image. To further reduce the dimensionality for the HMM stages (as well as for other algorithms), principal component analysis (PCA, see [8]) was performed. Using the IBM Hawthorne on-line recognition system (see [2] [3] [13]) we projected CDF and other feature vectors onto the space spanned by the eigenvectors corresponding to the d largest eigenvalues of the overall covariance matrix.

set/size	feat/dim	after PCA	% correct
ETL/1000	CDF/72	27	96.12
ETL/1000	CDF/88	23	96.96
ETL/1000	CDF/88	27	96.92
NIST/1000	CDF/72	27	88.00
NIST/2000	CDF/72	27	88.00
NIST/1000	CDF/88	23	85.56
NIST/2000	CDF/88	23	87.25
NIST/1000	CDF/88	27	88.25
NIST/2000	CDF/88	27	89.06

Table 2: Reco rates for single-state HMM

Several other methods of feature extraction were used. Windowing in the vertical direction (see [9]) i.e. averaging blackness over 12 or 24 horizontal stripes, referred to as the CO/12 and CO/24 features, yields a 12 (24) dimensional feature vector for each horizontal step. Fixing the number of such steps at 16, we trained 8, 12, and 16-state models.

set/size	feat/dim	states	% correct
ETL/1000	CO/12	8	90.12
ETL/1000	CO/12	12	93.35
ETL/1000	CO/12	16	95.08
ETL/1000	CO/24	8	88.73
ETL/1000	CO/24	12	92.24
ETL/1000	CO/24	16	94.19
NIST/1000	CO/12	12	73.38
NIST/2000	CO/12	12	76.06
NIST/1000	CO/12	16	79.12
NIST/2000	CO/12	16	79.81

Table 3: Reco rates for multi-state HMM

In the literature, several methods of feature extraction using sliding windows have been proposed: computing the blackness of individual pixels within the window [11], FFT estimation of the magnitude spectrum for vertical lines [5], and tracking the vertical displacement of black lines. So far we have concentrated on this last method: in each column of pixels we detect the (vertical) runs of black, and take the center of gravity and the width of the first 5 runs to form a 10-dimensional feature vector for each (passively slant-corrected) column of the image. If there are fewer than 5 runs, we use zeros as needed to fill up the feature vector. With the image normalized to 16 columns to keep results comparable with those presented so far, we have a total of 160 features per image prior to PCA. While HMM-based recognition of isolated characters with these features is not as accurate as the best (neural net or HMM) results presented above, we hypothesize that this disadvantage at the recognition level is amply compensated for by increased accuracy of segmentation. We are currently testing this hypothesis at the word level.

In the following tables we report results only for the NIST data, using "x/y" format for the 1000/2000 member training sets. In all experiments, the testing sets are kept constant (100 images per character, disjoint from the training data) across the training sets, and the 1000 training set is a subset of the 2000 training set. (An 'F' following the number of states refers to full covariance models, as opposed to the diagonal covariance used elsewhere – we give some results here but defer the discussion to Section 3.)

As can be seen from Tables 2 and 4, the effect of dimension reduction depends greatly on the horizontal granularity of the vectors we start with. The CDF span all 16 columns, and reducing the overall dimension from 88 (or 72) to 27 or even 23 produces recognition rates in the 85-89% range. The 40*4 features span only 4 columns (PCA is performed on vectors concatenated from 4 adjacent columns) and recognition rates are reduced to 75-76%. If dimensions are jointly reduced for the 16 vectors, we get slightly better results (77-78%, but only in full covariance models). The same conclusion is supported by looking at models which use similar encodings but scan the images top to bottom, rather than left to right. By voting the best left to right and top to bottom models we get 91.84% recognition rate at 3.5% rejection or 89.88% at 0 rejection on the NIST/2000 set. Altogether, our results confirm that PCA works well for handwriting not only in the on-line but also in the image-based domain.

Not surprisingly, feature sets encompassing the totality of the character image have better discrimination properties than feature sets restricted to a horizontal or vertical slice. But our preliminary experiments on word-level recognition indicate that this advantage can turn into a disadvantage if no reliable segmentation is provided in advance, inasmuch as parts of characters can (and do) strongly resemble other characters.

dim∗columns	after PCA	states	% correct
10∗16	27	1	69.44/70.75
10∗16	27	1F	78.88/78.62
10∗16	30	1	n.a./68.31
10∗16	30	1F	77.94/77.94
10∗16	32	1	68.81/70.75
10∗16	32	1F	77.56/77.56
10∗16	35	1	68.75/70.31
40∗4	10∗4	4	68.31/69.31
40∗4	10∗4	7	73.94/75.62
40∗4	15∗4	4	68.56/69.31
40∗4	15∗4	7	75.56/76.06
40∗4	20∗4	4	66.81/66.88
40∗4	20∗4	7	71.75/72.31

Table 4: The impact of PCA

The performance of the more highly optimized recognizers do not greatly depend on the exact number of dimensions used in the dimension reduction step: here and in what follows our results with 23 or 30 dimensions are quite comparable (see also Table 5).

3 Training the HMM

Since full (as opposed to diagonal) covariance models can take into account that the results of PCA are decorrelated only for the whole data set, but not necessarily within the individual classes, in principle it should be more advantageous to use full covariance models, especially for unreduced feature sets. However, as the number of parameters grows quadratically in the dimension of the feature vector, it is often problematic to train full covariance models in higher dimensions. For example, Table 4. above contains no full covariance results for dimension 35 or higher because the training set simply does not contain a sufficient number of data points. *Mixture models*, a standard tool in speech recognition [1], offer a way out of this problem. In the following Table 5, the numbers in boldface show the points where the increased number of mixtures begins to provide better results than full covariances (where the latter is available). As can be seen, the best result in this series, over 3% better than the best full covariance model, is found in a range (35 dimensions, 40 mixtures) where full covariance models can no longer be trained for lack of data.

# of mixtures:	10	20	30	40
dim/tr_set size				
27/1000	**79.19**	79.69	80.25	80.38
27/2000	**80.44**	81.88	82.94	82.12
30/1000	77.56	**80.12**	80.75	81.69
30/2000	78.62	**82.50**	82.44	82.31
32/1000	**78.00**	79.81	79.00	80.50
32/2000	**79.94**	82.19	82.81	82.44
35/1000	78.56	79.44	78.88	79.88
35/2000	79.31	82.31	82.25	83.25
40/1000	76.94	79.19	79.56	79.06
40/2000	79.19	80.44	80.00	80.88
27/2000[1]	90.12	**91.31**	91.62	91.93

Table 5: The impact of more mixtures

Since in general increasing the number of mixtures has beneficial effects both in single state and in multistate models, the possibility of using *tied mixtures* [6] was also investigated. In one set of experiments, the 10*16 LO10 features were first reduced to 27 dimensions, and the resulting data file, containing altogether 32,000 feature vectors, were subjected to unsupervised clustering into a set of n 27-dimensional gaussians.

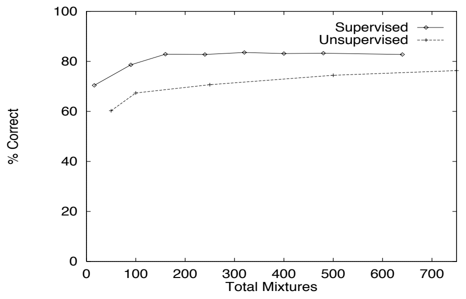

Figure 1: Supervised vs. unsupervised clustering

The lower curve of Figure 1. shows a plot of the total number of clusters versus the recognition rates for the LO10 features obtained from unsupervised clustering. Comparison of these results with Table 5. shows that even a large number of unsupervised clusters remains quite ineffective compared to clusters that are derived character by character, state by state. To support this conclusion, another set of experiments was run in which clustering was supervised at the character level but not at a state by state level. The results

[1]CDF/88 features reduced to 27 dimensions.

are shown in the top curve of Figure 1. It is evident from comparison of the two curves in this figure, that the models created using character level supervised clustering have an advantage over the tied mixture models.

Table 6. shows the results obtained from different multistate models where the number of clusters equals the number of states. Models in the first column were created from character level and state level supervised clustering of the 10*16 LO10 features broken into 7 windows of width 4 columns, with a step size of 2 columns. After Principal Component Analysis these windows are each reduced to 10 dimensions. Each state of these models is defined by one unique cluster only. The second column contains models created from these same features using supervised clustering at the character level only and therefore each state contains some combination of the same set of gaussians that are shared by all states. The results in this table indicate that there is a further benefit in supervised clustering at the state level in addition to the character level, or, in other words, that tieing of clusters offers no benefits for the data sets considered here.

states	char sup	char & state sup
1	53.50	55.06
3	55.62	66.12
5	61.75	73.00
7	64.75	75.62
9	65.50	76.62

Table 6: State level supervised clustering

Since increasing the number of parameters will in general increase the fit with the data, our results (Tables 2, 3, 4, and 6) showing improved recognition rates with more states are not particularly surprising. However, there is a more subtle effect in the trade-off between increased number of states vs. increased number of mixtures that has not, to our knowledge, been emphasized (or even mentioned) in the literature. For a k-state diagonal covariance model with m mixtures per state, d-dimensional features require the estimation of km d-dimensional means, the same number of covariances, and $2k-3$ transition probabilities (assuming the conventional left-to-right architecture with self-loops, left-to-right transitions, and jumps). The last two columns in Table 7 show recognition rates for HMMs trained on the NIST/1000 and NIST/2000 data sets. We use boldface wherever we see increasing performance in spite of a decreasing number of parameters. As the reader can see, these are almost always located at points where the number of states is increased (exceptions are marked by italics).

d	m	k	p	1000	2000
12	20	16	7709	84.38	85.00
12	20	12	5781	82.38	84.56
16	5	24	3885	**85.81**	**87.69**
12	10	16	3869	85.00	85.81
16	4	24	3117	84.25	**87.38**
12	10	12	2901	82.31	84.19
16	5	16	2589	**82.81**	**84.25**
16	3	24	2349	**85.62**	**86.44**
16	4	16	2077	82.88	84.62
12	5	16	1949	*84.62*	*85.69*
16	2	24	1581	83.69	**86.62**
16	3	16	1565	82.50	84.88

Table 7: The impact of k vs. m for comparable number of parameters

The isolated character experiments that we performed influenced the design of the full cursive system in three major ways. First, we have shown that that by careful selection of feature sets and model architecture HMMs can be tuned to perform at or near the recognition rates of the best MLP systems. Second, we concluded that the best strategy to increase the number of parameters in the model is to increase the number of states, so in the full cursive system step size is taken to be one pixel. Finally, we established that of two well-known techniques of reducing computational complexity PCA performs well on our domain but tied mixtures fail to live up to their promise.

4 Word-level segmentation

To show that HMMs actually outperform the pipeline architecture discussed in the Introduction we considered segmentation as a separate problem. Within a single line two segmentation tasks are relevant: segmentation of the line into words and segmentation of the words into characters. Given that character-level segmentation by the usual method [14] of searching for extrema in the continuous curves that make up cursive words is known to be highly sensitive to image quality, we concentrated on the word-level segmentation problem.

As a baseline, we built a word-level segmenter based on the same projection technique that proved quite satisfactory in finding the line: within the legal amount field we calculate column projections along the slant angle and use the troughs in the resulting blackness profile as indicators of word boundaries. Here we concentrate on finding the content words starting with the first written number and ending with the last one. On the image, these words are typically followed by a handwritten horizontal line to leave no free space, by the fractional part ‘cc/100’ or ‘cc/XX’, or the preprinted word ‘dollars’. In the following Table 8, the percentage of boundaries correctly placed within 4, 8, ...32 pixels of the hand-verified boundaries is given for the left boundary and the right boundary both for the projection-based segmenter P and the HMM segmenter H.

	4	8	12	16	20	24	28	32
l/P	36	51	55	58	60	62	65	66
l/H	34	55	65	71	76	78	81	83
r/P	1.5	2.6	4.2	5.0	5.4	5.4	5.4	5.4
r/H	25	42	53	60	64	66	69	71
b/P	0.2	0.8	1.8	2.2	2.8	3.0	3.0	3.0
b/H	9	24	35	44	50	53	57	60

Table 8: Finding content words

As can be seen, the traditional projection method is quite comparable to the HMM method at the left edge, for very high pixel precision. However, if errors of 1 mm (24 pixels) are tolerable, the HMM method is considerably better. At the right edge, the projection method falls apart, because it is incapable of distinguishing content words from contentless words such as the pre-printed 'dollars', while the performance of the HMM model is much less impacted. Finally, if both left and right ends need to be within a specified pixel distance, the HMM method, still in its infancy, already provides more reliable results.

Acknowledgements

The authors would like to acknowledge the positive impact of many discussions held with the on-line handwriting recognition group at IBM Hawthorne, in particular with Krishna Nathan, Jayashree Subrahmonia, and Homayoon Beigi. We benefited a great deal from their advice and from our study of their system.

References

[1] J.K. Baker, "Stochastic modeling for automatic speech understanding," Reprinted in A. Waibel and Kai-Fu Lee (eds) *Readings in Speech recognition*, Morgan Kaufmann, San Mateo CA, 1990. pp. 297-307

[2] E.J. Bellegarda, J.R. Bellegarda, D. Nahamoo and K.S. Nathan, "A Probabilistic Framework for the Recognition of On-line Handwriting," *Proc. 3rd International Workshop on Frontiers in Handwriting Recognition,* Buffalo, NY, pp. 225-234, May 1993

[3] J.R. Bellegarda, D. Nahamoo, K.S. Nathan and E.J. Bellegarda, "Supervised Hidden Markov Modeling for On-line Handwriting Recognition," *Proc. 1994 ICASSP,* Adelaide, South Australia, Vol 5, pp 149-152, April 1994

[4] Richard G. Casey: Moment Normalization of Handprinted Characters. *IBM Journal of Research and Development* 1970, 548-553

[5] A. J. Elms, "A connected character recognizer using Level Building of HMMs," *Proc. 12th IAPR International Conference on Pattern Recognition*, pp. 439-441, 1994.

[6] S.A. Euler, B.-H. Juang, C.-H. Lee, and F.K. Soong. "Statistical segmentation and word modeling techniques in isolated word recognition," In *ICASSP-90*, pages 745-748, Albuquerque, 1990.

[7] V.N. Gupta, M. Lennig, and P. Mermelstein, "Integration of acoustic information in a large vocabulary word recognizer," *Proc. ICASSP* 1987

[8] H. Hotelling, "Analysis of a complex of statistical variables into principal components," JEP 24:417-41, 498-520, 1933.

[9] A. Kaltenmeyer, T. Caesar, J.M. Gloger and E. Mandler, "Sophisticated Topology of HMMs for Cursive Script Recognition," *Proc. 2nd ICDAR* 1993, 139-142.

[10] T. Kohonen, J. Kangas, J. Laaksonen, K. Torkkola. "LVQ_PAK: A program package for the correct application of Learning Vector Quantization algorithms," *Proceedings of the International Joint Conference on Neural Networks*, pages I 725-730, Baltimore, June 1992. IEEE.

[11] S.-S. Kuo, O. Agazzi, "Visual Keyword Recognition Using Hidden Markov Models," *Proc. CVPR*, pp. 329-334, 1993.

[12] K.M. Mohiuddin, J. Mao, "A Comparative Study of Different Classifiers for Handprinted Character Recognition," *Pattern Recognition in Practice IV*, pp. 437-448, 1994.

[13] K.S. Nathan, J.R. Bellegarda, D. Nahamoo and E.J. Bellegarda, "On-line Handwriting Recognition Using Continuous Parameter Hidden Markov Models," *Proc. 1993 ICASSP,* Minneapolis, MN, pp. 121-124, April 1993

[14] K.M. Sayre, "Machine Recognition of Handwritten Words: A Project Report," *Pattern Recognition*, Vol. 5, pp.213-228, 1973.

[15] H. Takahashi, "A Neural Net OCR Using Geometrical and Zonal-pattern Features," *Proc. 1st ICDAR*, pp. 821-828, 1991.