Robust Word Recognition for Museum Archive Card Indexing

S. M. Lucas*, A.C. Tams**, Sung J. Cho+, Sungho Ryu+ , A.C. Downton**
*Dept. of Computer Science
**Dept. of Electronic Systems Engineering
University of Essex, Colchester CO4 3SQ, UK
+AI Lab, Division of Computer Science, KAIST, Taejon, South Korea

Abstract

This paper describes a novel robust approach to enable efficient searching of the type-written text on museum archive cards. Depending on such factors as the state of the typewriter and its ribbon, these text images may be faint with parts of the character missing, or be in heavy type with adjacent characters merging together. Both these problems can make this kind of text hard to read with conventional OCR methods that rely on the use of a limited number of segmentation hypotheses prior to recognition. Our method involves sliding a classifier over the entire word or card image, such that we get a set of recognition hypotheses for each possible window position which gives rise to a large character hypothesis graph. We then apply a graph reduction followed by an efficient graph search method to search for words in the reduced graph. Results so far are promising, with our system achieving 45% word recognition accuracy compared to the 25% achieved by a leading commercial package. However, searching the original larger graphs is much slower but yields 85% accuracy, so further work is needed either in improving the graph reduction method, or in improving the efficiency with which we can search the larger graph.

A. Introduction

This paper describes a novel robust method for word recognition that is especially appropriate for indexing museum archive cards. The quality and typography of the textual information on such cards varies, but typically may be characterised like this:

- Most information is machine printed, although hand-written annotations are common.
- Machine-printed information has often been created using a type-writer with a fixed-space font.
- The quality and thickness of the type varies, but may contain faded or broken characters, or over-heavy touching characters. These can make segmentation difficult.
- Much of the information may be assumed to come from known dictionaries. For example, in the current Pyrolidea archive we are using, the set of all species is known in advance (but which cards they occur on is not known).
- A significant minority of words are outside any currently available dictionary; however, methods that can cope with very large dictionaries (e.g. formed by taking the union of many different dictionaries) may still be appropriate.

When dictionaries are not available it should be possible to exploit statistical methods such as n-gram techniques[10, 5] for index construction and searching. This paper, however, deals exclusively with the dictionary-based approach.

Given the accuracy of current commercial off-the-shelf (COTS) OCR packages, one might assume that converting the machine print on these archive cards would be a solved problem. This, however, is not the case. Most COTS systems seem to be too brittle to handle difficult cases. We believe the reasons for this are as follows:

- Touching or broken characters cause segmentation problems, and most existing systems appear to consider only a limited number of segmentation hypotheses.
- The OCR engine is generally used in a *hard* classification mode, where it only returns the top few character hypotheses.
- Dictionary search (if done) is done sub-optimally.

On the other hand, there are hand-writing recognition systems[2, 4] that are designed from the outset to cope with hard-to-segment images, and like the method we propose here are based heavily on graph-search algorithms.

The method we propose does not involve any prior segmentation, and considers any number of recognition hypotheses (up to the size of the alphabet) at each point in the image. This overcomes the first two problems. We then use a highly efficient method to search the character hypothesis graph for the n best scoring paths, where each of these paths satisfies the constraint that its concatenated arc labels form a word in the dictionary. This graph-based dictionary search method has been described previously[6, 7] and here we shall just treat it as a black box. The main features of the technique are that it offers best-first retrieval and it scales well in the sense that retrieval speed is independent of the size of the dictionary. This should be contrasted with more conventional approaches that either get slower linearly with respect to the size of a flat dictionary [9] or with respect to the size of a Trie structured dictionary [3]. On the other hand, lexical knowledge can be applied in an approximate way in constant time [8] (again, with respect to dictionary size) just by considering the best recognition candidate in each position, with the occasional use of a wild card for places where the recognizer fails. We are more interested, however, in how we can apply lexical constraints to optimally interpret the outputs of the sliding window OCR process.

The rest of this paper is as follows: the next Section describes the method with the aid of an example; Section C describes the experimental setup; Section D reports some initial results; Section E concludes.

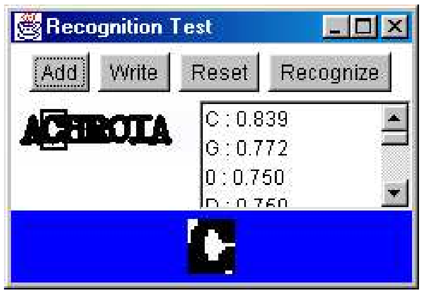
B. Mapping Text Images to Character Hypothesis Graphs

In this Section we describe with the aid of an example the process of mapping a text image into a character hypothesis graph. An example hard-to-segment image is shown in Figures 1 and 2. These Figures show the the same image displayed in our sliding window recognition tool. This tool allows an investigator to position the OCR window (shown by the rectangle) at any point on the image. The contents of the OCR window are immediately displayed in the lower panel, and clicking a mouse button fires the recognition engine, which produces a sorted set of character hypotheses (in best-first order) as shown in the right-hand panel.

Figure 1 shows a correct character (‘C’) at the head of the list with a confidence of 0.839. Figure 2, on the other hand, shows an incorrect character (‘M’) at the head of the list with a confidence of 0.812. Note that this value of 0.812 is higher than many of the correct recognition scores for this image. This indicates that a simplistic ’peak-picking’ segmentation strategy is unlikely to succeed.

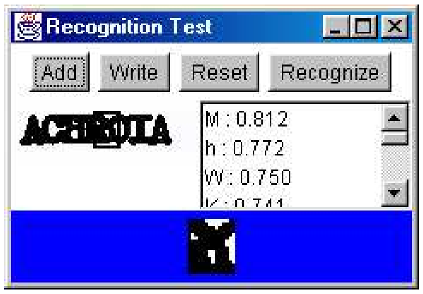
Figure 3 shows the intensity map created by mapping the maximum OCR output at each point in the ’ACHROIA’ image of Figure 1 to a grey level in the range 0 (black) to 1 (white).

Figure 4 shows the maximum of each column of the intensity map of Figure 3 plotted against its horizontal offset. This again illustrates the difficulties in making simplistic segmentations of this kind of image.



C : 0.839
G : 0.772
0 : 0.750
D : 0.750

Figure 1. An example window position producing a good response.



M : 0.812
h : 0.772
W : 0.750
K : 0.741

Figure 2. An example window position producing a spurious response.

B.1. Creating the Graph

We have now covered all the concepts necessary for our graph construction algorithm. The current algorithm we use can be outlined as follows.

1. Apply a soft classifier to each possible window position defined by the its top-left corner at point (x, y) in the word image. The output of the classifier is a vector whose ith dimension is its ‘confidence’ in the ith class being present at that point and is stored in a table $r[x][y][i]$. Figure 3 shows the maximum response over all classes at each (x, y) point as an intensity map.

2. Create a table $a[x][i]$ indexed on x coordinate and class i where $a_{xi} = \text{MAX}_{y=1}^{h}(r_{xyi})$; h being the maximum value of y.

3. Create a table of maximum responses in each column over all classes $m[x] = \text{MAX}_{i=1}^{n}(a_{xi})$ where there are n classes. Figure 4 plots this.

4. Group the recognition hypotheses according to spatial considerations as explained below. Each group will form a node in our graph. The confidence in the ith class in the group is again the *max* over all the hypotheses for that class.

Figure 3. Intensity map showing level of maxmimum OCR response for each window position.

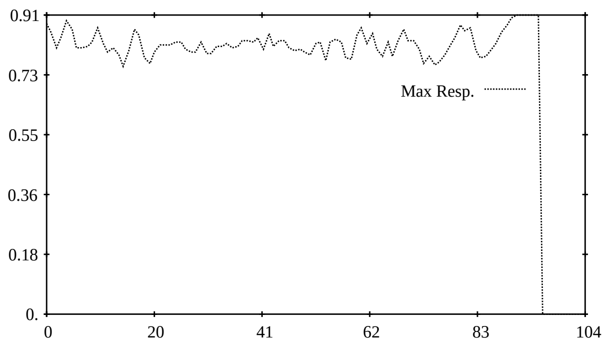


Figure 4. Plot of maximum OCR response versus horizontal pixel offset.

5. To simplify the graph, prune all hypotheses at a node that either fall below a certain confidence level or a certain rank.

6. Create a graph where all the recognition hypotheses in a node are made into arcs connecting all the nodes that pass a *connectable* test, described below.

This leaves two critical features unspecified: how we form the groups (nodes) and how we judge two nodes to be connectable.

For the experiments described below we did this as follows. Group-divides were placed wherever m_x fell below a critical cutoff - we chose 0.8 for all the experiments run here. We considered groups (nodes) connectable according to the following test: Two nodes N1 and N2 were deemed connectable if:

```
minDiff := N2.min - N1.max;
maxDiff := N2.max - N1.min;
connect if ((minDiff <= 14) AND
            (maxDiff >= 11))
```

If two nodes were connectable, then all the hypotheses on the first node were made into arcs to connect to the second node. Keeping only the best five hypotheses in each node, we get the graph shown in Figure 5. Each arc label has two parts: a character hypothesis and a probability or confidence value (truncated for display purposes). There is an implied order in that the graph should be traversed from left-to-right. This graph can then be used directly with our dictionary search system.

Using this graph, the search system failed to find any dictionary words in it starting at nodes 0 or 1 and finishing at nodes 10 or 11. However, another image (‘ALPHEIAS’) that is nearly as hard to segment and is shown in the bottom row of Figure 6 was recognized perfectly.

C. Experimental Setup

The image data comes from a set of about 30,000 Pyrolidea archive cards we scanned in as part of a joint project with the Natural History Museum, London. In general we use four datasets to define a trainable dictionary word recognition experiment:

OCR training set: a set of isolated single character images similar in nature to the set of characters found in whole words in the test set - but disjoint from that set;

Whole word training set: a set of whole word images together with the true word represented in the image (given as a character string).

dictionary: ideally this is the set of all words that we might wish to recognise on the cards;

test set: a set of whole-word images where each image is tagged with a transcription of the word (represented as a sequence of ASCII characters) contained in the image.

For the experiments given below however, our system had no word-level adjustable parameters (which should be used to optimise the graph-construction process), and so we did not use a whole-word training set.

We created a training set of individual characters by manually tagging data using a fixed size window of 14 by 16 pixels. We manually tagged all the type-written characters on five cards, and then looked through some more cards in order to find at least one example of each character. The result of this process was a highly skewed training set with 1063 character images spread over 62 classes (digits plus upper and lower case alphabetic). The number of characters per class varied between one (upper case ‘X’) and 36 (lower-case ‘n’).

The dictionary supplied did not have complete coverage of all words in the test set. For these experiments we extended the dictionary where necessary to include all the words in the test set. The augmented dictionary contained 16769 words with an average length of 9.4 characters and a maximum length of 19 characters.

We created a test set of 100 words, again by manually tagging them. These were captured from an entirely different batch of cards to the batch used to create the single character training images. The tagging was done at the level of a whole word corresponding to the entire image of that word e.g. no positional information of individual characters within the image was recorded.

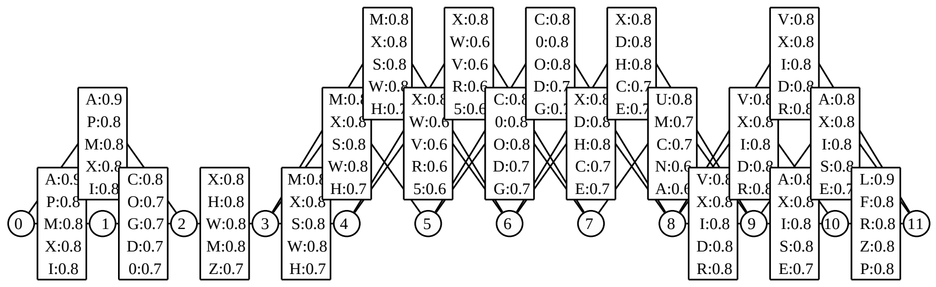
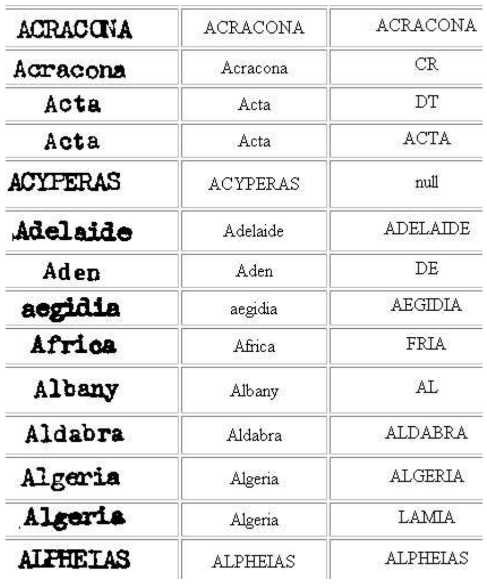

0
1
2
3
4
5
6
7
8
9
10
11
A:0.9 P:0.8 M:0.8 X:0.8 I:0.8
A:0.9 P:0.8 M:0.8 X:0.8 I:0.8
C:0.8 O:0.7 G:0.7 D:0.7 0:0.7
X:0.8 H:0.8 W:0.8 M:0.8 Z:0.7
M:0.8 X:0.8 S:0.8 W:0.8 H:0.7
M:0.8 X:0.8 S:0.8 W:0.8 H:0.7
M:0.8 X:0.8 S:0.8 W:0.8 H:0.7
X:0.8 W:0.6 V:0.6 R:0.6 5:0.6
X:0.8 W:0.6 V:0.6 R:0.6 5:0.6
C:0.8 0:0.8 O:0.8 D:0.7 G:0.7
C:0.8 0:0.8 O:0.8 D:0.7 G:0.7
X:0.8 D:0.8 H:0.8 C:0.7 E:0.7
X:0.8 D:0.8 H:0.8 C:0.7 E:0.7
U:0.8 M:0.7 C:0.7 N:0.6 A:0.6
V:0.8 X:0.8 I:0.8 D:0.8 R:0.8
V:0.8 X:0.8 I:0.8 D:0.8 R:0.8
V:0.8 X:0.8 I:0.8 D:0.8 R:0.8
A:0.8 X:0.8 I:0.8 S:0.8 E:0.7
A:0.8 X:0.8 I:0.8 S:0.8 E:0.7
L:0.9 F:0.8 R:0.8 Z:0.8 P:0.8

Figure 5. A character hypothesis graph for the ‘ACHROIA’ image in Figure 1.

D. Initial Results

We have tested the system, using the simple graph construction procedure outlined in Section B, on the first 100 images of the test set. We used a simple hamming distance nearest-neighbour classifier as our OCR engine. We chose this classifer by testing it against a support vector machine and an n-tuple classifier. The nearest neighbour method was the most accurate and also the fastest, probably due to the small size of the training set. The test accuracy of our nearest neighbour classifier when trained on 80% of the training data was around 91%, though it returned the correct character within the top five candidates in over 99% of cases.

A sample of the recognition results can be seen in Figure 6. Results on this set were 45% whole-word accuracy, compared with 25% for a leading COTS package.



ACRACCNA	ACRACONA	ACRACONA
Acracona	Acracona	CR
Acta	Acta	DT
Acta	Acta	ACTA
ACYPERAS	ACYPERAS	null
Adelaide	Adelaide	ADELAIDE
Aden	Aden	DE
aegidia	aegidia	AEGIDIA
Africa	Africa	FRIA
Albany	Albany	AL
Aldabra	Aldabra	ALDABRA
Algeria	Algeria	ALGERIA
Algeria	Algeria	LAMIA
ALPHEIAS	ALPHEIAS	ALPHEIAS

Figure 6. Sample recognition results. Left column shows the word image; middle and right columns show the true transcription and our system transcription respectively.

At present, our system is much slower than the COTS package, with timing figures for the processing stages as follows for a Java implementation of all the algorithms running on a 450 MHz Pentium III. For the example ‘ACHROIA’ image from Section B sliding the classifier over the window (which involved applying the 62-class classifier at about 2,000 different points in the image) took about 25s. Building the graph then took about 0.1s. Loading the graph into the graph search system took about 0.4s, and getting the best 10 retrievals took about 0.05s. Clearly, the sliding OCR takes the vast majority of the time. Although this may at first appear to be a significant disadvantage of the approach, this is only true when we naively apply the OCR engine independently to each point in the image (as we do currently). We are also investigating special scanning n-tuple methods that work by shifting each line of the image through a recognition buffer and then connecting these vertically to generate recognition hypotheses in a much more efficient manner.

D.1 Searching the Larger Hypothesis Graph

On inspection it appeared that the main source of error was in the graph reduction process i.e. the mapping of the large hypothesis graph that can be obtained directly from sliding the OCR engine over the word image into the reduced graph of the kind shown in Figure 5. Our current implementation of the fast dictionary search method reported in [7] would not fit in RAM (128Mb on the current machine) when searching graphs of this size.

This led us to make some experiments on searching the larger graph, using a dynamic programming procedure[1] to find the best path through thr graph for each dictionary word independently. The larger graph is created by applying the OCR engine to every point in the image, then taking the maximum output for each class in each column i.e. the $a[x][i]$ array from Section B.1 step 3. We create a node in the graph labelled with the integer x for every column x in the array, then add arcs to the graph between each node x and node $(x+13)$ labelled with every pattern class i together weight $a[x][i]$. Furthermore, we also add a ‘skip’ arc with a weight of 0.95 between all adjacent nodes. This skip-weight should really be estimated from the whole-word training data, of course.

We performed an independent match between this graph and every word in our dictionary. We defined the score for a path as the product of all the arc weights for that path, and defined the match score for a word as being the score of the best (highest scoring) path for that word. This is consistent with a probabilistic interpretation of the OCR class outputs, although in fact the class score output by our nearest-neighbour classifier was simply the ratio of the number of matching pixels to the total number of pixels in the character image for the best matching image in that class.

This matching process is relatively slow, and takes about 3 minutes per word on average to score all the words in our 16k word dictionary (i.e. about 11ms per individual dictionary word match). It should be possible to speed this up significantly by storing the dictionary in a trie structure.

After each word has been scored we sort them into order of best-score first. Table 1 shows some word images where both the COTS package and our fast reduced-graph method fail to recognise the word. It is notable that the large-graph matching method correctly recognises the ACHROIA image, which appears to be very difficult. This method achieved 85% word recognition accuracy on the 100 word set (compared with 25% and 45% for the COTS and fast match methods respectively). Nonetheless, the system still fails in some cases that look to be relatively easy, but this may be due to the poor quality of the OCR training data.

E. Discussion and Conclusions

This paper has presented a simple method for segmentation-free word recognition which is especially appropriate for indexing the type-written text on museum archive cards.

The approach is novel in two ways. First, in the method of applying an OCR engine in 'sliding window mode' to every point in a word image; this may be seen as an extreme form of over-segmentation. Second, in application of the fast dictionary search method reported in [7] to finding the best matching dictionary word in the graph.

There are many rather *ad hoc* features to our graph construction method that cause it to mis-recognise images that can be recognised correctly by dealing directly with the unreduced graph, as we showed in Section D.1. We are currently investigating better ways of reducing the graph, and also more efficient methods for searching the larger unreduced graph.

References

[1] R. Bellman. *Dynamic Programming*. Princeton University Press, (1957).

[2] T. M. Breuel. A system for off-line recognition of handwritten text. In *Proceedings of 12th International Conference on Pattern Recognition (ICPR)*, pages 129 – 134 vol 2, 1994.

[3] D. Chen, J. Mao, and K. Mohiuddin. An efficient algorithm for matching a lexicon with a segmentation graph. In W. Lea, editor, *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pages 543 – 546. IEEE, 1999.

[4] G. Dzuba, A. Filatov, and A. Volgunin. Handwritten zip code recognition. In *Proceedings of the 4th IEEE Int. Conf. on Document Analysis and Recognition*, pages 766–770, Ulm, Germany, 1997.

[5] J. Hull and S. Srihari. Experiments in text recognition with binary n-grams and viterbi algorithms. *IEEE Transactions on Pattern analysis and Machine Intelligence*, 4(5):520 – 530, (1982).

[6] S. Lucas. Efficient best-first dictionary search given graph-based input. *Proceedings of International Conference on Pattern Recognition*, 2:471–474, 2000.

[7] S. Lucas. Efficient graph-based dictionary search and its application to text-image searching. *Pattern Recognition Letters*, 22:551 – 562, 2001.

[8] S. Madhvanath, S. McCauliff, and K. Mohiuddin. Extracting patron data from check images. In *Proceedings of International Conference on Document Analysis and Recognition (ICDAR)*, pages 519 – 522, 1999.

[9] M. Mohamed and P. Gader. Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 5:548 – 554, (1996).

[10] C. Suen. n-gram statistic for natural language understanding and text processing. *IEEE Transactions on Pattern analysis and Machine Intelligence*, 1(2):164 – 172, (1979).

Acknowledgements This work was funded under BBSRC/EPSRC Grant 84/BIO11933 and builds on earlier work funded by The Post Office (UK).

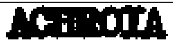
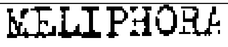
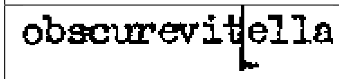
Word Image	True Word	Top Three Matches (in order)
	ACHROIA	ACHROIA, ACHROEA, ASEMIA
	MELIPHORA	MELIPHORA, MELITTIA, MELITENE
	obscurevitella	obscuripennis, coenulentella, obscurevitella

Table 1. Sample test word images and their large graph recognition results.