Correcting broken characters in the recognition of historical printed documents

Michael Droettboom
Digital Knowledge Center, Johns Hopkins University
3400 N. Charles St., Baltimore, MD 21218
mdboom@jhu.edu

Abstract

This paper presents a new technique for dealing with broken characters, one of the major challenges in the optical character recognition (OCR) of degraded historical printed documents. A technique based on graph combinatorics is used to rejoin the appropriate connected components. It has been applied to real data with successful results.

1 Introduction

Most commercial optical character recognition (OCR) systems are designed for well-formed, modern business documents. Recognizing older documents with low-quality or degraded printing is more challenging, due to the high occurrence of broken and touching characters. This paper presents an algorithm that rejoins the pieces of broken characters so they can be more robustly identified. The success of this algorithm is then measured using a set of real-world historical documents.

This research is part of the Gamera project [2], which aims to create a tool for building document-specific recognition systems for humanities documents. One of our goals for this project has always been to provide general solutions that will work across multiple languages and printing techniques.

For the purposes of this paper, a *connected component* (CC) is a set of black pixels that are contiguous. By definition, characters are *broken* when they are made up of too many CCs. Broken characters can not be joined simply by the distance of the CCs alone, since two intentionally separate characters can often be closer than the two parts of an accidentally broken character.

The inverse problem of touching characters is solved using another approach where a symbol classifier is trained to split connected components using projection-based heuristics [3]. While the original intent of that approach was for the splitting of musical symbols, it also performs extremely well on Roman text (93% accuracy on our dataset).

2 Other approaches

Thresholding converts a color or greyscale image to a bi-level image, such that black is used to indicate the presence of ink on the page and white is used to indicate its absence. Improving thresholding by looking for shades of grey in the areas where CCs almost touch, using entropy, can reduce the number of broken characters [7]. However, in many historical documents, the characters are completely broken on the page and intelligent thresholding, since it has no knowledge of the shapes of the target characters, performs poorly.

Active contour models (ACM), or snakes [5], find a vector outline for each symbol using certain constraints on the elasticity of the outline. Unfortunately, ACMs, which were designed for gross shape recognition, perform poorly on the fine details that are required to recognize printed characters.

Post-processing using some kind of language model, including a dictionary or n-grams of a language [4] has also been used to handle broken characters. However, such models are less useful for documents containing ancient languages, mixed-languages or a high occurrence of proper nouns.

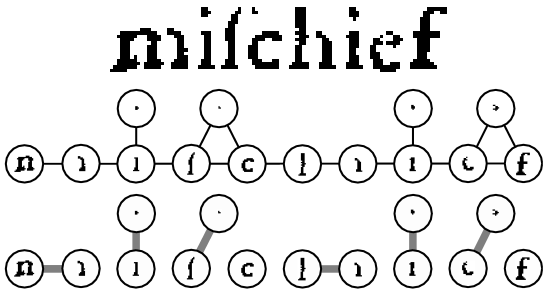
Therefore, an ideal solution would include knowledge of the individual symbols without requiring a language-specific model.

3 The algorithm

The goal of the broken character connection (BCC) algorithm is to find an optimal way to join CCs on a given page that maximizes the mean confidence of all characters.

The algorithm begins by building an undirected graph in which each vertex represents a CC in the image. Two vertices are connected by an edge if the border of the bounding boxes are within a certain threshold of distance. Experiments demonstrated that this threshold is best set to $^3/_4$ of the average distance between all bounding boxes. (CCs

(a)



(b)

(c)

Figure 1. (a) an original image of a word from the testbed; (b) how the connected components are connected to form a graph; (c) the correct solution.

can also be connected by morphological dilation, though the bounding box method is much faster and produces only slightly less accurate results.) This creates a forest of graphs where each graph is roughly equivalent to a word in the document. Figure 1(b) shows one such graph. A graph representation, rather than a string representation, of connectivity is necessary, since characters can be broken in the x- and/or y-direction and cycles can occur between CCs.

Next, all of the different ways in which the CCs can be joined are evaluated. Every possible connected subgraph is enumerated, by performing a depth-first search from each vertex. To avoid enumerating duplicate possibilities, the vertex v assigned a number N_v, and an edge is traversed from vertex a to b only if $N_b > N_a$. To improve runtimes, the depth of the search is limited to the maximum number of CCs that would typically make up a single broken character. This constant is adjusted automatically based on the amount of degradation in the image and is usually between 3 and 5. Each of these subgraphs is evaluated by merging all of its CCs into a single image and sending it to the symbol classifier. The symbol classifier returns a confidence value that indicates how similar the merged image is to known symbols in the database.

Since we are using a k nearest neighbor (k-NN) classifier [1] for symbol classification, it was most convenient to use a confidence measure based on distance. More elaborate ways of determining confidence, such as analyzing the clustering of symbols within the database, have been suggested, but they do not significantly affect the success of the BCC algorithm.

Once the parts of the subgraph have been evaluated, dynamic programming is used to find an optimal combination of these parts that maximizes the mean confidence of the characters across the entire subgraph (word).

While a full runtime complexity analysis of the algorithm is beyond the scope of this paper, the asymptotic upper bound is $O(n \ln n)$, where n is the number of vertices in the graph. However, when there are no cycles, the runtime is reduced to roughly $O(kn)$, where k is the maximum size of the subgraphs.

complete characters	81.3%
broken characters	10.7%
legitimately broken characters (**i**, **j**, **;**, **:** etc.)	6.2%
touching characters	1.8%

Table 1. Distribution of character types in the sample data. Note that failure to deal with broken and touching characters gives a maximum possible accuracy of 81.3%.

4 Results

To evaulate the BCC algorithm, we used the Statistical Accounts of Scotland [6]. This collection of census-like data was printed in 1799, with reused metal type on wooden blocks. The age of the paper, combined with the low-quality type and press-work, presents challenges for OCR. Table 1 shows the distribution of the types of characters in the collection. The manually-generated groundtruth data also makes this collection valuable for research.

The results below were obtained by training the classifier using five pages, and then testing the algorithm on five additional unseen pages with similar typeface.

If the symbol classifier only has knowledge of the complete characters in the image, BCC correctly finds 71% of the broken characters in our test data. By training the symbol classifier with examples of broken characters that were manually identified, BCC correctly finds 91% of the broken characters.

BCC also performs well with legitimately broken characters, such as **i**, **j**, **;** and **:**. By training the symbol classifier with examples of each of these characters, BCC was able to find and join 93% of the legitimately broken characters. This renders further procedural programming of heuristic rules (such as to attach **ı**'s to dots) unnecessary. Therefore, it easy to support new character sets that have other legitimately broken characters, such as the Greek majuscule xi (Ξ).

5 Conclusion

The technique presented here performs very well on the test dataset, not only for "accidentally" broken characters, but also for those that are legitimately broken. In addition, it seems to satisfy our goal of being relatively independent of document type: it seems to adapt to new kinds of symbols gracefully with a minimum of manual intervention.

References

[1] T. Cover, and P. Hart. 1967. Nearest neighbour pattern classification. *IEEE Trans. on Inform. Theory.* 13(1): 21–7.

[2] M. Droettboom, K. MacMillan, I. Fujinaga, G. S. Choudhury, T. DiLauro, M. Patton, and T. Anderson. 2002. Using the Gamera framework for the recognition of cultural heritage materials. *JCDL*. 11–7.

[3] I. Fujinaga, B. Alphonce, B. Pennycook, and K. Hogan. 1991. Optical music recognition: Progress report. *Int. Comp. Music Conf.* 66–73.

[4] S. M. Harding, W. B. Croft, and C. Weir. 1997. Probabilistic retrieval of OCR degraded text using N-grams. *Europ. Conf. on Dig. Libraries.* 345–59.

[5] M. Kass, A. Witkin and D. Terzopolous. 1987. Snakes: Active contour models. *Int. Conf. on Comp. Vision.* 259–68.

[6] *Statistical Accounts of Scotland*. 1799. `http://edina.ac.uk/statacc/`

[7] Ø. D. Trier, and A. K. Jain. 1995. Goal-directed evaluation of binarization methods. *IEEE Trans. on Pattern Analysis & Machine Intelligence.* 17(12): 1191–201.