Word Spotting: A New Approach to Indexing Handwriting

R. Manmatha, Chengfeng Han and E. M. Riseman, *
Center for Intelligent Information Retrieval,
Computer Science Department,
University of Massachusetts, Amherst, MA-01003.
manmatha@cs.umass.edu

Abstract

There are many historical manuscripts written in a single hand which it would be useful to index. Examples include the W. B. DuBois collection at the University of Massachusetts and the early Presidential libraries at the Library of Congress. Since Optical Character Recognition (OCR) does not work well on handwriting, an alternative scheme based on matching the images of the words is proposed for indexing such texts.

The current paper deals with the matching aspects of this process. Two different techniques for matching words are discussed. The first method matches words assuming that the transformation between the words may be modelled by a translation (shift). The second method matches words assuming that the transformation between the words may be modelled by an affine transform.

Experiments are shown demonstrating the feasibility of the approach for indexing handwriting.

The method should also be applicable to retrieving previously stored material from personal digital assistants (PDAs).

1 Introduction

Text has always been the primary source of information in conventional, and more recently digital libraries. If the text is in machine readable form (ASCII), it can be indexed using standard text retrieval engines. However, much of the text in both historical and even current collections is contained in paper documents. One solution is to use Optical Character Recognition (OCR) to convert scanned paper documents into ASCII. Existing OCR technology works well with standard machine printed fonts against clean backgrounds. It works poorly if the originals are of poor quality or if the text is handwritten. We propose an alternative solution for indexing handwritten text when a large corpus of texts written by a single person exists.

In the context of digital libraries, the problem being addressed in this paper is primarily related to the indexing of historical manuscripts. These manuscripts are largely written in a single hand. Examples include the collected works of W. E. B. Du Bois, the African American civil rights leader, and Margaret Sanger, a pioneer in birth control, which are stored at archives at the University of Massachusetts and Smith College respectively.

Such manuscripts are valuable resources for scholars as well as others who wish to consult the original manuscripts and considerable effort has gone into manually producing indices for them. For example, a substantial collection of Margaret Sanger's work has been recently put on microfilm (see http://MEP.cla.sc.edu/Sanger/SangBase.HTM) with an item by item index. The indexing scheme proposed here will help in the creation and production of indices and concordances for such archives.

Since the document is written by a single person, the assumption is that the variation in the word images will be small. The proposed solution will first segment the page into words and then match the actual word images against each other to create equivalence classes. Each equivalence class will consist of multiple instances of the same word. Each word will have a link to the page it came from. The number of words in each equivalence class will be tabulated. Those classes with the largest numbers of words will probably be stopwords, i.e. conjunctions such as "and" or articles such as "the". Classes containing stopwords are eliminated (since they are not very useful for indexing). A list is made of the remaining classes. This list is ordered occuring to the number of words contained in them. The user provides ASCII equivalents for a representative word in each of the top m (say m = 2000) classes. The words in these classes can now be indexed. This technique will be called "word spotting" as it is analogous to "word spotting" in speech processing [4].

The proposed solution completely avoids machine recognition of handwritten words as this is a difficult task [8]. Robustness is achieved compared to OCR systems for two reasons:

1. Matching is based on entire words. This is in con-

*This research was supported by the NSF Center for Intelligent Information Retrieval and by ARPA grant number N66001-94-D-6054.

trast to conventional OCR systems which essentially recognize characters rather than words.

2. Recognition is avoided. Instead a human is placed in the loop when ASCII equivalents of the words must be provided.

The present paper deals with the matching aspects of the problem (for a discussion of page segmentation into words, see [7]). A future paper will deal with the rest of the system. The matching phase of the problem is expected to be the most difficult part of the problem. This is because unlike machine fonts, there is some variation in even a single person's handwriting. This variation is difficult to model. Figure (1) shows two examples of the word "Lloyd" written by the same person. The last image is produced by XOR'ing these two images. The white areas in the XOR image indicate where the two versions of "Lloyd" differ. This result is not unusual. In fact, the differences are sometimes even larger.

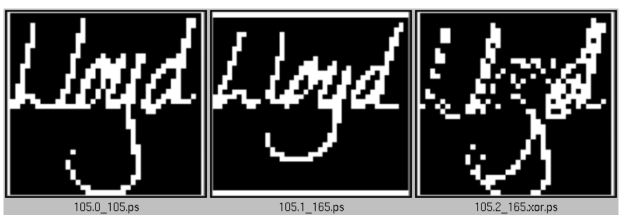

Figure 1: Two examples of the word "Lloyd" and the XOR image

In this paper, two different matching techniques are discussed. The first, based on Euclidean distance mapping [2], assumes that the deformation between words can be modelled by a translation (shift). The second, based on an algorithm by Scott and Longuet Higgins [9] models the transformation between words using an affine transform.

2 Prior Work

The traditional approach to indexing documents involves first converting them to ASCII and then using a text based retrieval engine [10]. Scanned documents printed in standard machine fonts against clean backgrounds can be converted into ASCII using an OCR [1]. However, handwriting is much more difficult for OCRs to handle because of the wide variability present in handwriting (not only is there variability between writers, but a given person's writing also varies).

Image matching of words has been used to recognize words in documents which use machine fonts [5]. Recognition rates are much higher than when the OCR is used directly [5]. Machine fonts are simpler to match than handwritten fonts since the variation is much smaller; multiple instances of a given word printed in the same font are identical except for noise. In handwriting, however, multiple instances of the same word on the same page by the same writer show variations. The first two pictures in Figure 1 are two identical words from the same document, written by the same writer. It may thus be necessary to account for these variations.

3 Outline of Algorithm

1. A scanned greylevel image of the document is obtained.

2. The image is first reduced by half by gaussian filtering and subsampling.

3. The reduced image is then binarized by thresholding the image.

4. The binary image is now segmented into words. this is done by a process of smoothing and thresholding (see [7]).

5. A given word image (i.e. the image of a word) is used as a template. and matched against all the other word images. This is repeated for every word in the document. The matching is done in two phases. First, the number of words to be matched is pruned using the areas and aspect ratios of the word images - the word to be matched cannot have an area or aspect ratio which is too different from the template. Next, the actual matching is done by using a matching algorithm. Two different matching algorithms are tried here. One of them only accounts for translation shifts, while the other accounts for affine matches. The matching divides the word images into equivalence classes - each class presumably containing other instances of the same word.

6. Indexing is done as follows. For each equivalence class, the number of elements in it is counted. The top n equivalence classes are then determined from this list. The equivalence classes with the highest number of words (elements) are likely to be stopwords (i.e. conjunctions like 'and' , articles like 'the', and prepositions like 'of') and are therefore eliminated from further consideration. Let us assume that of the top n, m are left after the stopwords have been eliminated. The user then displays one member of each of these m equivalence classes and assigns their ASCII interpretation. These m words can now be indexed anywhere they appear in the document.

We will now discuss the matching techniques in detail.

4 Determination of Equivalence Classes

The list of words to be matched is first pruned using the areas and aspect ratios of the word images. The pruned list of words is then matched using a matching algorithm.

4.1 Pruning

It is assumed that

$$\frac{1}{\alpha} \leq \frac{A_{word}}{A_{template}} \leq \alpha \tag{1}$$

where $A_{template}$ is the area of the template and A_{word} is the area of the word to be matched. Typical values of α used in the experiments range between 1.2 and 1.3. A similar filtering step is performed using aspect ratios (ie. the width/height ratio). It is assumed that

$$\frac{1}{\beta} \leq \frac{Aspect_{word}}{Aspect_{template}} \leq \beta. \tag{2}$$

The value of β used in the experiments range between 1.4 and 1.7. In both the above equations, the exact factors are not important but it should not be so large so that valid words are omitted, nor so small so that too many words are passed onto the matching phase.

4.2 Matching

The template is then matched against the image of each word in the pruned list. The matching function must satisfy two criteria:

1. It must produce a low match error for words which are similar to the template.

2. It must produce a high match error for words which are dissimilar.

Two matching algorithms have been tried. The first algorithm - Euclidean Distance Mapping (EDM) - assumes that no distortions have occured except for relative translation and is fast. This algorithm usually ranks the matched words in the correct order (i.e. valid words first, followed by invalid words) when the variations in words is not too large. Although, it returns the lowest errors for words which are similar to the template, it also returns low errors for words which are dissimilar to the template. The second algorithm [9],referred to as SLH here, assumes an affine transformation between the words. It thus compensates for some of the variations in the words. This algorithm not only ranks the words in the correct order for all examples tried so far, it also seems to be able to better discriminate between valid words and invalid words. As currently implemented the SLH algorithm is much slower than the EDM algorithm (we expect to be able to speed it up).

5 Using Euclidean Distance Mapping for Matching

This approach is similar to that used by [3] to match machine generated fonts. A brief description of the method follows (more details are available from [7]).

Consider two images to be matched. There are three steps in the matching:

1. First the images are roughly aligned. In the vertical direction, this is done by aligning the baselines of the two images. In the horizontal direction, the images are aligned by making their left hand sides coincide.

   The alignment is, therefore, expected to be accurate in the vertical direction and not as good in the horizontal direction. This is borne out in practice.

2. Next the XOR image is computed. This is done by XOR'ing corresponding pixels (see Figure 1).

3. An Euclidean distance mapping [2] is computed from the XOR image by assigning to each white pixel in the image, its minimum distance to a black pixel. Thus a white pixel inside a blob is assigned a larger distance than an isolated white pixel. An error measure E_{EDM} can now be computed by adding up the distance measures for each pixel.

4. Although the approximate translation has been computed using step 1, this may not be accurate and may need to be fine-tuned. Thus steps (2) and (3) are repeated while sampling the translation space in both x and y. A minimum error measure E_{EDMmin} is computed over all the translation samples.

6 SLH Algorithm for Matching

The EDM algorithm does not discriminate well between good and bad matches. In addition, it fails when there is significant distortion in the words. This happens with the writing of Erasmus Hudson (Figure 2). Thus a matching algorithm which models some of the variation is needed. A second matching algorithm (SLH), which models the distortion as an affine transformations, was therefore tried (note that it is expected that the real variation is probably much more complex). An affine transform is a linear transformation between coordinate systems. In two dimensions, it is described by

$$\mathbf{r}' = \mathbf{A}\mathbf{r} + \mathbf{t} \tag{3}$$

where $\mathbf{t}$ is a 2-D vector describing the translation, $\mathbf{A}$ is a 2 by 2 matrix which captures the deformation, $\mathbf{r}'$ and $\mathbf{r}$ are the coordinates of corresponding points in the two images between which the affine transformation must be recovered. An affine transform allows for the following