UC Berkeley, Computer Science

Gender Classification of Handwritten Text

Peter Cheng, Jeff Tsui, Alice Wang

May 14, 2013

1 Introduction

For our project we designed and tested an off-line classifier for gender prediction, using handwritten text. The inspiration for this project came from a Kaggle machine learning competition [1]. Kaggle provides sample training data, in the form of high-resolution (300 dpi) jpg images. Each image corresponds to a writing sample, and there are 4 writing samples for each of 475 writers. The 4 samples correspond to:

1. Arabic text, different text for each writer
2. Arabic text, same text for each writer
3. English text, different text for each writer
4. English text, same text for each writer

Section 2 provides some details that lead to preprocessing and feature extraction. Section 3 details the preprocessing tasks we performed on the Kaggle jpg images provided. Section 4 describes in detail the methodology behind feature extraction of the processed images. Section 5 demonstrates the performance of our extracted features using various canonical classifiers.

2 Background

For this particular contest, Kaggle provided 700+ extracted features for each writing sample. Upon examining relevant literature, we concluded that generating relevant features is as important, if not more so than honing the optimal classification model. We insert doc image

Figure 2.1: A sample input document image

diverged our project to focus on our own feature extraction. In order to do so on the highly unconstrained Kaggle dataset, we will perform a number of preprocessing steps.
The data we used consists of only the fourth sample for each writer, which is the same English text written by each writer. Since the data supplied is for a competition, the test data did not have gender labels. We decided to limit our dataset to the labeled training data, where label = 1 for male and label = 0 for female. Of these 282 writing samples, we reserved 25% to be our own testing set. An example of a sample document image is shown in Figure **??**.
We started with trying to classify our dataset with each characterâĂŹs appearance via Optical Character Recognition, and quickly concluded that it introduced more error and uncertainty than necessary. Instead, we perform a number of preprocessing and segmentation steps described in the next section. In this paper we refer to document image and word image, or line image, where each is an image containing a single document, word, or line respectively separately. Using the processed images, we generate a set of 56 features for each line. Then we calculate error rates from building several classification models, and summarize the results below.

3 Preprocessing and Segmentation

Typically, when performing handwriting classification and feature extraction, binary images with a number of constraints are assumed. Typical constraints seen in previous work include having images in black and white, text all equally scaled among different writers, and lines written at consistent angles [2]. Furthermore, many feature extraction approaches are tuned to work on images of individual characters, words, or lines, instead of an entire document altogether. As a result, before extracting features, we perform a series of preprocessing steps, to normalize each document image [1], followed by segmentation procedures to segment out lines and words.

3.1 Preprocessing

The first step in preprocessing is to translate each input color image into a binary black and white image. To accomplish this, an intensity threshold is calculated, such that pixels with intensity above that value are set to 1, while the rest are set to 0. This intensity threshold is calculated using MatlabâĂŹs âĂIJgraythreshâĂİ function, which performs OtsuâĂŹs method for binary thresholding [3]. Following this binarization, we then trim off the margins of each image, as writing samples are centered around different locations on the page. This is done simply by retaining the image within the minimum axis-aligned bounding box containing all text in the image.

[1]in this paper we refer to document image, word image, or line image, where each is an image containing a single document, word, or line respectively.

Figure 3.1: Line segments detected on a document image roughly correspond to a line of text

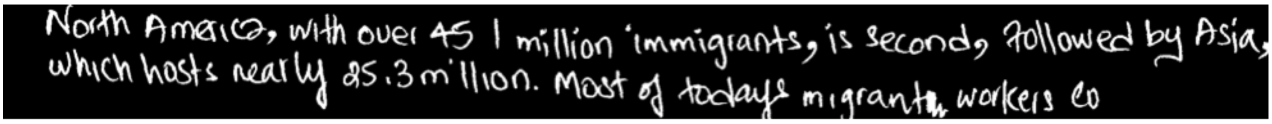

Figure 3.2: In this case, line segmentation failed to perform properly because two lines were very close together and a single detected line segment spanned both lines.

3.2 Segmentation

While features that are calculated on handwriting samples could theoretically be extracted from entire document images all at once, many of the features we use are specific to words or lines, so it makes sense to first break down the input space into line images and word images, so as to potentially reduce error when extracting these features.

3.2.1 Line Segmentation

To perform line segmentation, the Hough-transform-based algorithm proposed by Louloudis, G., et al, [4] is loosely followed. First, we use MatlabâĂŹs probabilistic Hough line segment detector (HoughlinesP) implementation to detect lines on the document images, preprocessed after the previous section. We restrict Hough peak detection to only occur in the angle domain such that detected lines deviate at most 5 degrees from horizontal. We bin angles at a resolution of 1 degree, and the rho parameter at a resolution of 1 pixel. The parameters for minimum line length and maximum gap within a line are more important, and are set to be 75% of the imageâĂŹs width and 10% of the imageâĂŹs width. An example of such detected lines is shown in Figure **??**. Each detected line is then drawn onto the word image, such that pixel values corresponding to the line are set to 1. Now, connected components are detected on the image with lines added, and if accurate lines were detected, each character and word in a line of text should all be in one contiguous connected component, held together by the lines drawn over them. Connected components with fewer than 10000 pixels are filtered out, as are those with a width less than 8 times the height, as the former case corresponds to punctuation or characters that were not properly grouped, while the latter case corresponds to cases where multiple lines of text were detected together. Each connected component, (without the detected lines), is now output at as a separate line image. An example of a failure case is shown in Figure 3.2.

3.2.2 Word Segmentation

The word segmentation procedure we use is similar to the line segmentation method described in the previous section. Essentially, smaller lines are detected within each line

Figure 3.3: A case where word rotation does not perform properly, as the minimum bounding box for the shown word is not upright.

image, such that individual words become the connected components. For this application we use the same angle and offset thresholds and resolutions as before, but set the minimum line length to be 2% of the line imageâĂŹs width, while the maximum gap in a line is 1.5% of the line imageâĂŹs width. After after filtering out connected components with fewer than 500 pixels, we can then extract word images via the connected components.

However when extracting features from words, it is useful to have words that are rotated such that their baseline is horizontal. There are a number of approaches to accomplish this [2], though we take a fairly simple approach. For each word, we compute its (rotated) bounding box, and rotate the image such that the bounding box is axis aligned. This tends to work well for a high percentage of words, though in certain cases it does not perform well, for instance as shown in Figure **??**.

4 Feature Extraction

In this section, we describe the features recovered from line images and word images, after preprocessing and segmentation, as described in the previous two sections. A listing of each feature is shown in Figure **??**

4.1 Word Features

The first group of features are generated based on word images, referencing the work of Marti, U.-V., et al [5] where a feature vector is extracted from each word. These feature vectors are averaged by line to combine them with line-specific feature vectors. The word features include the width, slant, and height of the three main writing zones.

4.1.1 Word Height

The three writing zones are separated by the upper and lower baselines. The upper and lower baselines are defined via analysis via the histogram of number of dark pixels in every line, where 15% of dark pixels exist above the upper baseline and 90% of dark pixels exist above the lower baseline. The baselines are used to calculate features f1-f6, where the top line is first row of the image and bottom line is the bottom row. These features encode the height of words. The ratios of f1, f2, and f3 are taken to adjust for the variation in word size between writing samples.

4.1.2 Word Width

The width of words is measured by first finding the row with the most black and white transitions. The number of white gaps between every group of black-white-black pixels is

1. upper baseline - topline
2. lower baseline - upper baseline
3. bottom line - lower-baseline
4. f1 / f2
5. f1 / f3
6. f2 / f3
7. median of the gap lengths
8. f2 / f7
9. average of slant angles (degrees)
10. std dev of slant angles (degrees)
11. line angle
12. slant of lower contour
13. mean sq error of lower contour
14. freq of local max for lower contour
15. freq of local min for lower contour
16. avg left slope of local max for lower contour
17. avg right slope of local max for lower contour
18. avg left slope of local min for lower contour
19. avg right slope of local min for lower contour
20. 12 for upper contour
21. 13 for upper contour
22. 14 for upper contour
23. 15 for upper contour
24. 16 for upper contour
25. 17 for upper contour
26. 18 for upper contour
27. 19 for upper contour
28. avg width of connected components
29. avg height of connected components
30. std dev of width of ccs
31. std dev of height of ccs
32. avg dist between adjacent ccs
33. std dev of dist between adjacent ccs
34. avg area of enclosed regions
35. avg length of major axis of ers
36. avg length of minor axis of ers
37. avg orientation of ers
38. avg eccentricity of ers
39. avg equiv diameter squared of ers
40. avg extent of ers
41. avg perimeter of ers
42. avg form factor of ers
43. avg roundness of ers
44. std dev of 34
45. .
46. .
47. .
48. .
49. .
50. .
51. .
52. .
53. std dev of 43
54. fractal slope 1
55. fractal slope 2
56. fractal slope 3

Figure 4.1: The line drawn in this image contains the most black/white transitions

Figure 4.2: The slant of various parts of letters, as detected via adjacent pixels

calculated and the median of these values is taken to represent the width of the writing. Again, to account for the word size variation between writers, the ratio is taken with f2, the vertical height of the middle portion of the writing. An example of this is shown in Figure **??**

4.1.3 Word Slant

Slant of writing is another useful characteristic that we encode by creating a histogram of angles for each word. First we convert the image to an outline, leaving only the perimeter pixels at the edge of each word. For each black pixel in the middle row of the middle zone (between upper and lower baselines), we calculate the angle from the pixel to a connected pixel intersecting with the upper and lower baselines. The angle between pairs of pixels and their connected intersecting pixels are collected and the average and standard deviations are used to encode the slant for the word. An example of this is shown in Figure **??**

4.2 Line Features

The next group of features we generated were extracted from line images. This includes the angle of each line, the characteristics of its contour, various calculations performed on connected components and enclosed regions, as well as fractal dimension. We referenced work from Bouletreau, V., et al [6], Hertel, C. and Bunke, H., [7], and Vincent, N. [8] to create these features.

4.2.1 Line Angle

The first feature comes from the angle of the line. This is simply the angles of the lines detected in Section 3 when performing line segmentation. If multiple line segments were detected for a single line, the average of their angles is used. For subsequent features, we use the rotated line so that writing is parallel with the x axis.

4.2.2 Contour

The contour of the writing is a useful characteristic of handwriting. The upper and lower contours of a line are the sequence of uppermost and lowermost pixels in each column of a line. Gaps in which a column of the image has no black pixels are removed. The characteristic contours are generated by eliminating discontinuities in the upper and lower contours. This is done by shifting y coordinates of consecutive points so they are at most 1 pixel apart on the y axis. From the characteristic lower and upper contours,

Figure 4.3: A contour extracted from a line of text

several features are extracted. The slope of the least squared regression line is calculated, as well as the mean squared error for the line. The frequency of local minima and maxima is determined for the contours by dividing the number of local extrema by the length of the contour. The local extrema are found by comparing the neighboring three points on either side. We also compute the average slopes of the left three points and the right three points for each local maxima. All contour features we generated are summarized below. Figure **??** also demonstrates an example of an extracted contour.

4.2.3 Connected Components and Enclosed Regions

The next two sets of features correspond to characteristics of connected components and enclosed regions within a line of text. For a writer whose handwriting tends to overlap itself, connected components, or rectangular boxes that bound together the regions of connected objects will generally be larger, whereas for other writers, each connected component corresponds to a separate letter. The height, width, and spacing between connected components also reveals characteristics about writing as well. Enclosed regions are, in some sense, the opposite of connected components. These correspond to the areas within loops and other enclosed areas within a line of handwriting. Calculating values such as the roundness or eccentricity of each enclosed region can encapsulate information about how slanted a sample of text is, and the curvature of certain letters. The features derived from connected components and enclosed regions correspond to features 28-53 from the table in Figure **??**.

4.2.4 Fractal Dimensions

Fractal dimensions encode the degree of irregularity and fragmentation of the handwriting, from which our final three features are derived[8]. At a high level, they measure the area (measured in number of pixels) a handwritten text grows as a dilation operation is applied onto a line image. Given X as the contour of the handwriting sample, its fractal behavior is generated via the evolution of the areas of successive dilation sets of boxes on its contour. Xn is defined as the n-sized structuring square element, and A(X) denotes the area of set X. The x values are log(n), and corresponding y values are log[A(Xn)] - log(n).
Using the Minkowski-Bouligand dimension, the fractal behavior of the X set is expressed by the linear relationship between log[A(Xn)] and log(n) [6]. In plotting the xâĂŹs and yâĂŹs, the fractal features are the slopes of the three-part linear regression line that fits all possible points on the x-axis and and minimizes the mean squared error between the original points of the graph and the line segments[9]. The three regressions correspond to three zones of the image: zone 0 characterizes the line thickness, which is omitted since it varies based on resolution and image quality; zone 1 characterizes the writing shape; and zone 2 matches the dilations from which the writing is hidden. An example plot of

Figure 4.4: The slopes of the 3 straight lines that produce the best least squares fit constitute 3 features.

3 detected lines correspond to the 3 fractal dimension features is shown in Figure **??**

5 Results

need to add table here

6 Conclusion

In our study of gender classification based on handwriting, we extracted 56 features based on word and line properties and applied several classification models. To benchmark our results, we also used the features provided in the Kaggle competition in comparison to our results. In general, our features resulted in slightly higher error rates than the Kaggle feature set, but neither performed extremely well. For line average, our features performed best with a [INSERT] average using a [INSERT] classifier. For document average, our features performed the best with [INSERT] average and using a [INSERT] classifier.

There are several ways to improve our classification system. A modification to our preprocessing steps that segment documents into words and lines could result in higher accuracy in what each feature is measuring. Furthermore, we noticed that dividing highly variable handwritten documents with skewed lines and different sizes is not a trivial task. Much of our work consisted of improving the accuracy of segmentation and reducing false positives. To further decrease the error rate, we can continue to generate additional features. The Kaggle dataset has roughly 1000 features, while we achieved similar results at 56 features. Finally, we extracted most of our features based on methods developed for writer identification, since gender classification based on handwriting is similar to writer identification. However, it can be argued the same features used in writer identification may not be best applied to gender classification. Adding more features that measure different properties of the handwriting will improve the accuracy of our classifier.

References

[1] "Icdar2013 - gender prediction from handwriting." `http://www.kaggle.com/c/icdar2013-gender-prediction-from-handwriting`.

[2] T. Caesar, J. Gloger, and E. Mandler, "Preprocessing and feature extraction for a handwriting recognition system," in *IEEE 1993*, 1993.

[3] N. Otsu, "A threshold selection method from gray-level histograms," *Systems, Man and Cybernetics, IEEE Transactions on* **9**(1), pp. 62–66, 1979.

[4] G. Louloudis, B. Gatos, I. Pratikakis, and K. Halatsis, "A block-based hough transform mapping for text line detection in handwritten documents."

[5] U.-V. Marti, R. Messerli, and H. Bunke, "Writer identification using text line based features," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pp. 101–105, 2001.

[6] V. Bouletreau, Vincent, R. N., Sabourin, and H. Emptoz, "Parameters for handwriting classification," in *IEEE 1997*, 1997.

[7] C. Hertel and H. Bunke, "A set of novel features for writer identification," in *AVBPA*, 2003.

[8] N. Vincent, V. Bouletreau, H. Emptoz, and R. Sabourin, "How to use fractal dimensions to qualify writings and writers," in *Fractals*, 2000.

[9] A. Hassaine, A. Somaya, and A. Bouridane, "A set of geometrical features for writer identification, neural information processing," in *Neural Information Processing*, 2012.