

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 relevant literature review, we concluded that generating relevant features

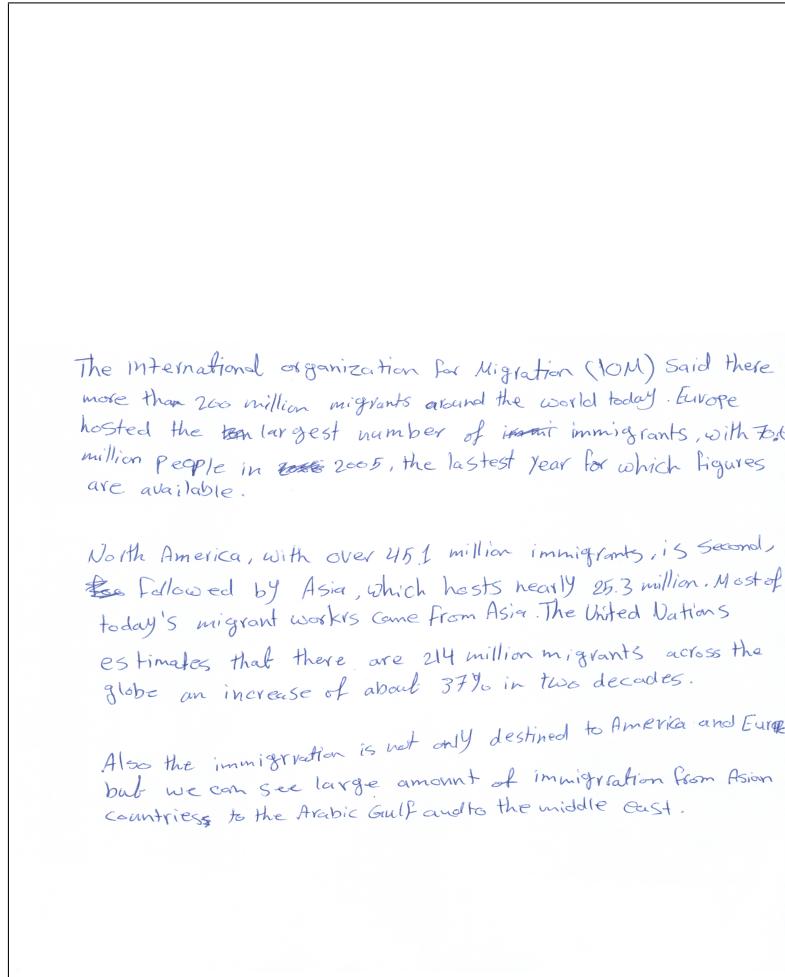


Figure 2.1: A sample input document image

is as important, if not more so than honing the optimal classification model. We diverged our project to focus on our own feature extraction. However, prior to doing so we will perform a number of preprocessing steps in order to do so on the highly unconstrained Kaggle dataset.

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 testing set. An example of a sample document image is shown in Figure 2.1.

We started with trying to classify our dataset with each character's appearance via Optical Character Recognition, and quickly concluded that it introduced more error and uncertainty than needed. Instead, we performed a number of preprocessing and

segmentation steps as will be described in the next section. We refer to document image as an image containing a single document, and word image, or line image, as an image containing a single word, or line respectively. Using the processed images, we generated a set of 56 features for each line image. From then on we calculate error rates from 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, prior to feature extraction, we perform a series of preprocessing steps to normalize each document image¹, 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 Matlab's "graythresh" function, which performs Otsu's method for binary thresholding [3]. Following this binarization, we then trimmed off the margins of each image, as writing samples are centered around different locations on the page. This is accomplished by retaining the image within the minimum axis-aligned bounding box containing all text in the image.

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 Matlab's probabilistic Hough line segment detector (HoughlinesP) implementation to detect lines on the document images, preprocessed after the previous section. We restricted Hough peak detection to only occur in the angle domain such that detected lines deviate at most 5 degrees from horizontal. We

¹in 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.

The International Organization for Migration (IOM) said there
more than 200 million migrants around the world today. Europe
50 hosted the largest number of migrant immigrants, with 70.1
million people in 2005, the latest year for which figures
100 are available.

North America, with over 65.1 million immigrants, is second
150 followed by Asia, which hosts nearly 89.3 million. Most of
today's migrant workers come from Asia. The United Nations
200 estimates that there are 214 million migrants across the
globe—an increase of about 37% in two decades.
250

Also the immigration is not only destined to America and Europe
300 but we can see large amount of immigration from Asian
countries to the Arabic Gulf and to the middle east.

500

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

North America, with over 45.1 million immigrants, is second, followed by Asia, which hosts nearly 25.3 million. Most of today's migrant workers co

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.

binned angles at a resolution of 1 degree, and the rho parameter at a resolution of 1 pixel. Since the parameters for minimum line length and maximum gap within a line are more important, they are set to be 75% of the image's width and 10% of the image's width. An example of such detected lines is shown in Figure 3.1. Each detected line is then "drawn" onto the word image, such that pixel values corresponding to the line are set to 1. Next, connected components are detected on the image with added lines, and for lines that were accurately 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 were filtered as it corresponds to punctuation or characters that were not properly grouped, along with those with a width less than 8 times the height since they corresponds to cases where multiple lines of text were detected together. Each connected component is now output as a separate line image without the lines on top.

In some cases, our line segmentation technique was unable to correctly separate adjacent lines, where we are unable to correctly separate the two lines. This is seen in Figure 3.2, where the lines are written very closely together and there is a slight angle to each line, where the first line dips in the center slightly and the second line slants downwards. These factors make it difficult to perform line segmentation.

3.2.2 Word Segmentation

The word segmentation procedure we use is similar to the line segmentation method described in the previous section. Individual words become the connected components as smaller lines are detected within each line image. We use the same angle and offset thresholds and resolutions as before, but set the minimum line length to be 2% of the line image's width, and the maximum gap in a line is 1.5% of the line image's width. After filtering out connected components with fewer than 500 pixels, we can then extract word images via the connected components.

It is useful to have words that are rotated such that their baseline is horizontal during feature extraction. Caesar, T., et al [2] cite a number of approaches to accomplish this, but we took a fairly simple approach. For each word, we computed its (rotated) bounding box, and rotated the image such that the bounding box is axis-aligned. As seen in Figure 3.3, this worked well for a high percentage of words but did not in certain cases. The writer used cursive writing with little spacing between consecutive words, which resulted in the three words "I J there", "I J are", and "I J 21" being segmented as one word.



Figure 3.3: A case where word segmentation does not perform properly, as adjacent words are connected.

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

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 are comprised of the width, slant, and height of the three main writing zones, described below.

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. An example is shown in Figure 4.2. The baselines are used to calculate features f1 through 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 row with the most black and white transitions defines the width of words. To represent the width of the writing, the number of white gaps between every group of black-white-black pixels is calculated and the median of these values is taken. Again, the ratio is taken with f2, the vertical height of the middle portion of the writing to account for the word size variation between writers. An example of this is shown in Figure 4.3

- | | | |
|--|--|---------------------------------------|
| 1. Upper baseline to top line distance | min for lower contour | axis of ers |
| 2. Lower baseline to upper baseline distance | 19. Avg right slope of local min for lower contour | 37. Avg orientation of ers |
| 3. Bottom line to lower baseline distance | 20. 12 for upper contour | 38. Avg eccentricity of ers |
| 4. 1 / 2 | 21. 13 for upper contour | 39. Avg equiv diameter squared of ers |
| 5. 1 / 3 | 22. 14 for upper contour | 40. Avg extent of ers |
| 6. 2 / 3 | 23. 15 for upper contour | 41. Avg perimeter of ers |
| 7. Median of the gap lengths | 24. 16 for upper contour | 42. Avg form factor of ers |
| 8. 2 / 7 | 25. 17 for upper contour | 43. Avg roundness of ers |
| 9. Average slant angle | 26. 18 for upper contour | 44. Std dev of 34 |
| 10. Std dev of slant angles | 27. 19 for upper contour | 45. Std dev of 35 |
| 11. Line angle | 28. Avg width of connected components | 46. Std dev of 36 |
| 12. Slant of lower contour | 29. Avg height of connected components | 47. Std dev of 37 |
| 13. Mean squared error of lower contour | 30. Std dev of width of ccs | 48. Std dev of 38 |
| 14. Freq of local max for lower contour | 31. Std dev of height of ccs | 49. Std dev of 39 |
| 15. Freq of local min for lower contour | 32. Avg dist between adjacent ccs | 50. Std dev of 40 |
| 16. Avg left slope of local max for lower contour | 33. Std dev of dist between adjacent ccs | 51. Std dev of 41 |
| 17. Avg right slope of local max for lower contour | 34. Avg area of enclosed regions | 52. Std dev of 42 |
| 18. Avg left slope of local | 35. Avg length of major axis of ers | 53. std dev of 43 |
| | 36. Avg length of minor | 54. Fractal dimension slope 1 |
| | | 55. Fractal dimension slope 2 |
| | | 56. Fractal dimension slope 3 |

Figure 4.1: The features we extracted from each input document image

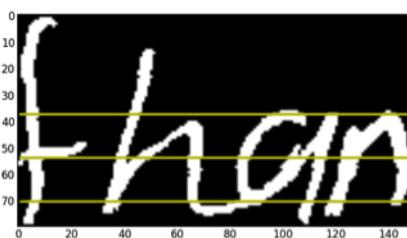


Figure 4.2: The upper and lower baselines are shown in this example. The middle baseline halfway between the upper and lower baselines is also shown.

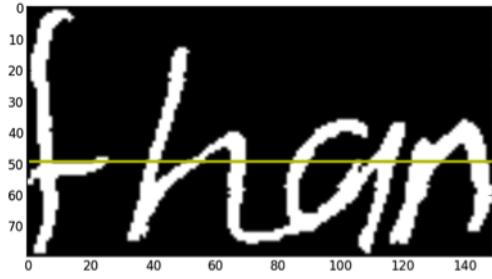


Figure 4.3: The line drawn in this image contains the most black/white transitions

4.1.3 Word Slant

We created a histogram of angles for each word in order to generate the slant of writing. First we converted the image to an outline such that only the perimeter pixels at the edge of each word remain. For each black pixel in the middle row of the middle zone (found between upper and lower baselines), we calculated the angle from the pixel to a connected pixel intersecting with the upper and lower baselines. To encode the slant for the word, the angle between pairs of pixels and their connected intersecting pixels are averaged and the standard deviations used. An example of this is shown in Figure 4.4

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 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

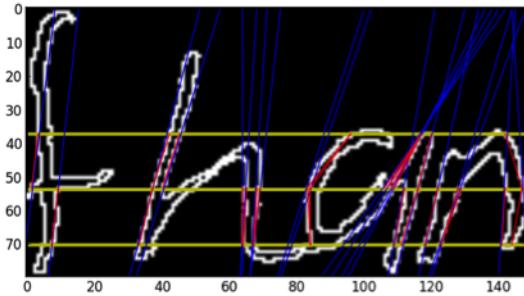


Figure 4.4: The slant of various parts of letters, as detected via adjacent pixels

pixels are removed. By eliminating discontinuities in the upper and lower contours, the characteristic contours are generated via shifts of y-coordinates of consecutive points such that they are at most 1 pixel apart on the y-axis. Several features are extracted from the characteristic lower and upper contours, such as 1) the slope of the least squared regression line 2) the mean squared error for the line 3) the frequency of local minima and maxima, found by dividing the number of local extrema by the length of the contour, 4) the local extrema, determined by comparing the the neighboring three points on either side. The average slopes of the left three points and the right three points for each local maxima are computed, as well. All contour features we generated are summarized below. Figure 4.5 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. Connected components, or rectangular boxes that bound together the regions of connected objects will generally be larger for writers whose handwriting tends to overlap itself, and corresponds to a separate letter for writers who fall outside this category. The height, width, and spacing between connected components provide additional characteristics about writing as well. Enclosed regions, on the other hand are the opposite of connected components. In a line of handwriting, they correspond to areas within loops and other enclosed areas. The roundness or eccentricity of each enclosed region can reveal information about how slanted a text sample is as well as the curvature of certain letters. The features derived from connected components and enclosed regions correspond to features 28-53 from the table in Figure 4.1.

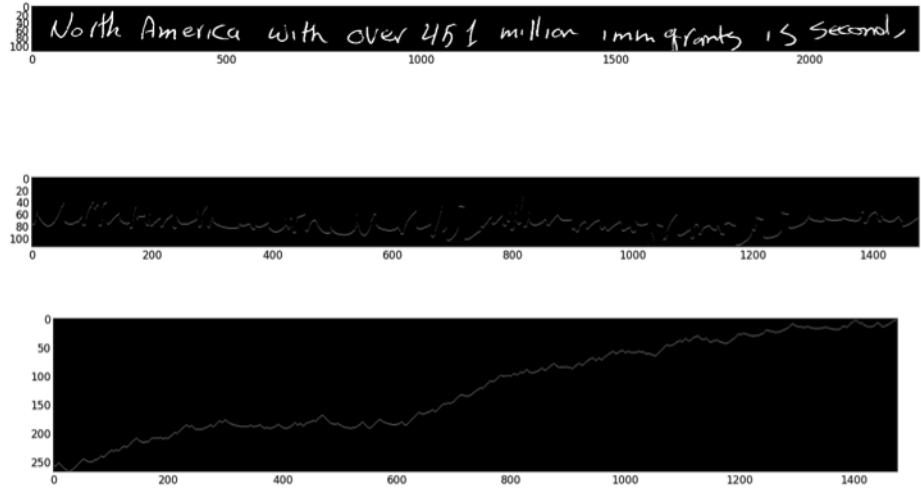


Figure 4.5: From top to bottom: the rotated line image, the lower contour of the line, and the lower characteristic contour.

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. X_n 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 $\log[A(X_n)] - \log(n)$.

Using the Minkowski-Bouligand dimension, the fractal behavior of the X set is expressed by the linear relationship between $\log[A(X_n)]$ and $\log(n)$ [6]. In plotting the x 's and y 's, the fractal features are the slopes of the three-part linear regression line that fits all possible points on the x -axis 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 3 detected lines correspond to the 3 fractal dimension features is shown in Figure 4.6

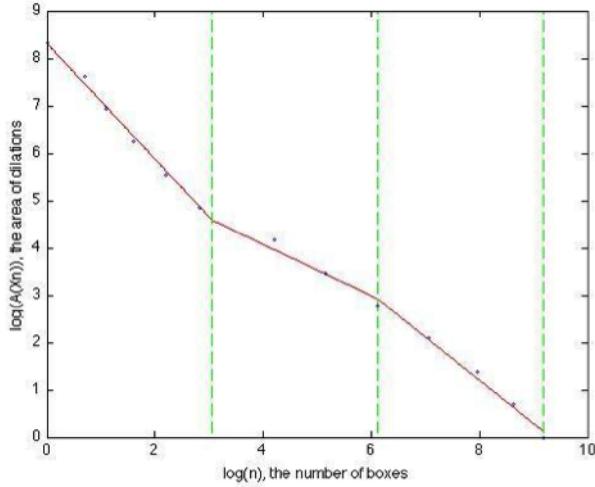


Figure 4.6: The slopes of the 3 straight lines that produce the best least squares fit constitute 3 features.

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.