Writer Identification

Pattern Recognition Course

Dec 24th, 2018

Introduction

For centuries, handwriting analysis has been a field of interest for numerous graphologists, psychologists, and forensic experts. With the advances in technologies and artificial intelligence, researchers from all over the world have dedicated their time and energy in automating the process of handwriting analysis using computers to ensure high efficiency and fast computations.

Writer identification is a sub-field of handwriting analysis that generally refers to the process of identifying the writer of a piece handwritten text.

In this report, we present a writer identification system, where the system is required to identify the writer identity of a handwritten paragraph after getting trained upon handwriting of some different writers.

In our work, we used texture-based approach for writer identification, we used the potential texture descriptor, namely Local Binary Pattern (LBP), and we used Support Vector Machine (SVM) as a classifier to distinguish between feature vectors of different writers. We tested our system on IAM handwritten dataset, and we were able to achieve correct identification accuracy of about 99%.

Project Structure

Our system is composed of three main modules: pre-processing module, feature extractions module, and classification module.

In brief words, our system pipeline goes as follows:

First, we load the test case images and pass them to the pre-processing module that is responsible for removing the noise from the handwritten text images, detecting the handwritten text region in the image, and segmenting the paragraph into separate lines.
Second, we the pre-processed handwritten text lines are passed to the feature extraction module where we extract the handwriting features using texture-based approach. Finally, all the training feature vectors along with their corresponding writer IDs are passed to a Support Vector Machine (SVM) classifier to identify the writer of the test image.

In the following sections, we are going to describe every module in a bit of detail.

Pre-processing Module

As mentioned above, the purpose of this module is to extract noise-free handwritten text lines from the whole image.

We begin by applying Gaussian filter of kernel size 5×5 to reduce the noise in the image. After that, we apply Otsu thresholding to get a binarized version of that image.

Our pre-processing module is composed of two main sub-modules: paragraph detection, and line segmentation.

1. Paragraph Detection

As we were testing our system on IAM dataset, we needed to detect and remove the printed header and footer of the dataset images and keep only the handwritten text region. We locate the main horizontal lines of the header and footer by finding the connected components of the image. The three connected components having bounding rectangle with the largest horizontal width are considered to be these three main horizontal lines.

After detecting the three main horizontal lines we erode the binarized image to further remove any left noise and random dots. We did the erosion step after the three main horizontal lines detection because erosion splits these lines into multiple discontinuous thinner lines which cannot be detected by the above method. After erosion, we detect the minimum surrounding bounding box around the handwritten text region by skipping the white rows and columns.

2. Line Segmentation

Even though we mentioned that we are using texture-based features that can be applied on the entire paragraph without the need of line segmentation, we found that extracting features from lines is much faster than extracting them from the entire paragraph. In addition, we need line segmentation in classification, as we are classifying each line separately and use a vote-like classification system.

In this pre-processing sub-module, we designed a sophisticated algorithm for segmenting the handwritten paragraph image into lines which results in segmenting about 98% of IAM dataset lines correctly. The good thing is that even if the algorithm fails in segmenting any lines, it will not affect the feature extraction process, because it is meant to work on the entire handwritten paragraphs.

The details of this segmentation algorithm are beyond the scope of this report. We are just going to explain the main steps of the algorithm.

First, we begin by projecting the binarized image horizontally, forming a horizontal histogram h, where h_r is the number of black pixels in row r. Second, we start detecting the peaks in this histogram that are above a certain threshold value that is calculated by the following equation

$$threshold = \frac{\max_i(h_i)}{3}$$

After detecting those peaks, we begin detecting the valleys between every two consecutive peaks.

By using the above threshold value, the algorithm is going to miss short lines, and if we decrease the threshold value, the algorithm starts detecting multiple peaks corresponding to a single line. So, to handle this issue, we are doing a second run to detect all missing peaks using another approach. We iterate on every two consecutive valleys, if the distance between them is greater than a certain threshold value that is calculated by the below equation, then we are probably missing a line or more between these two valleys.

$$threshold = 1.5\ average\_distance\_between\_peaks$$

We are not mentioning lots of implementation details that enabled us to detect peaks and valleys in a robust way, without getting easily affected by tilted, skewed, and close lines.

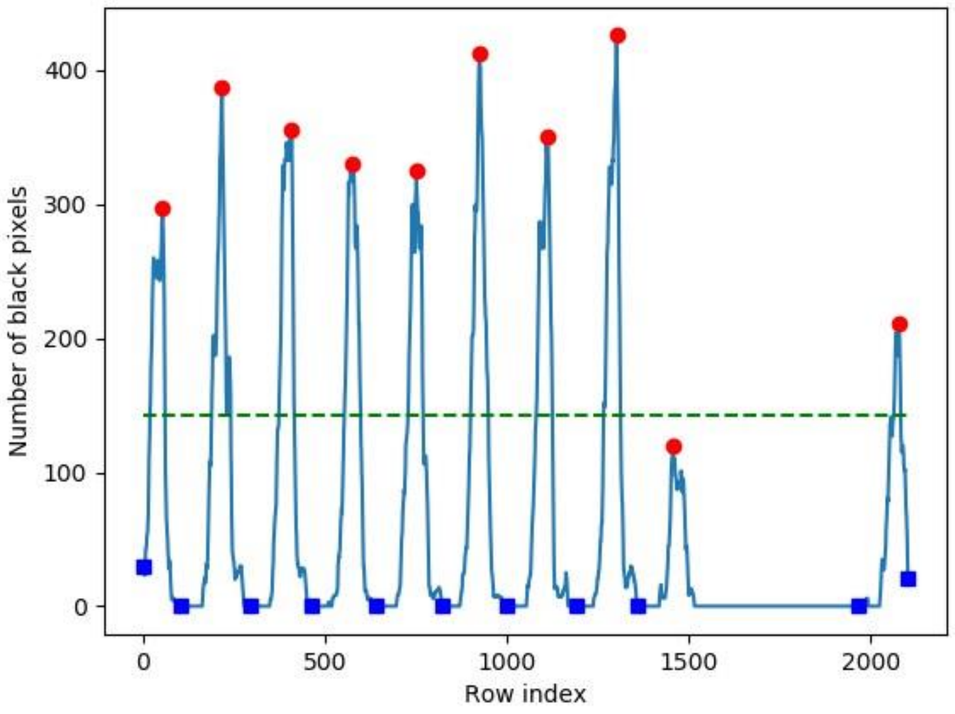
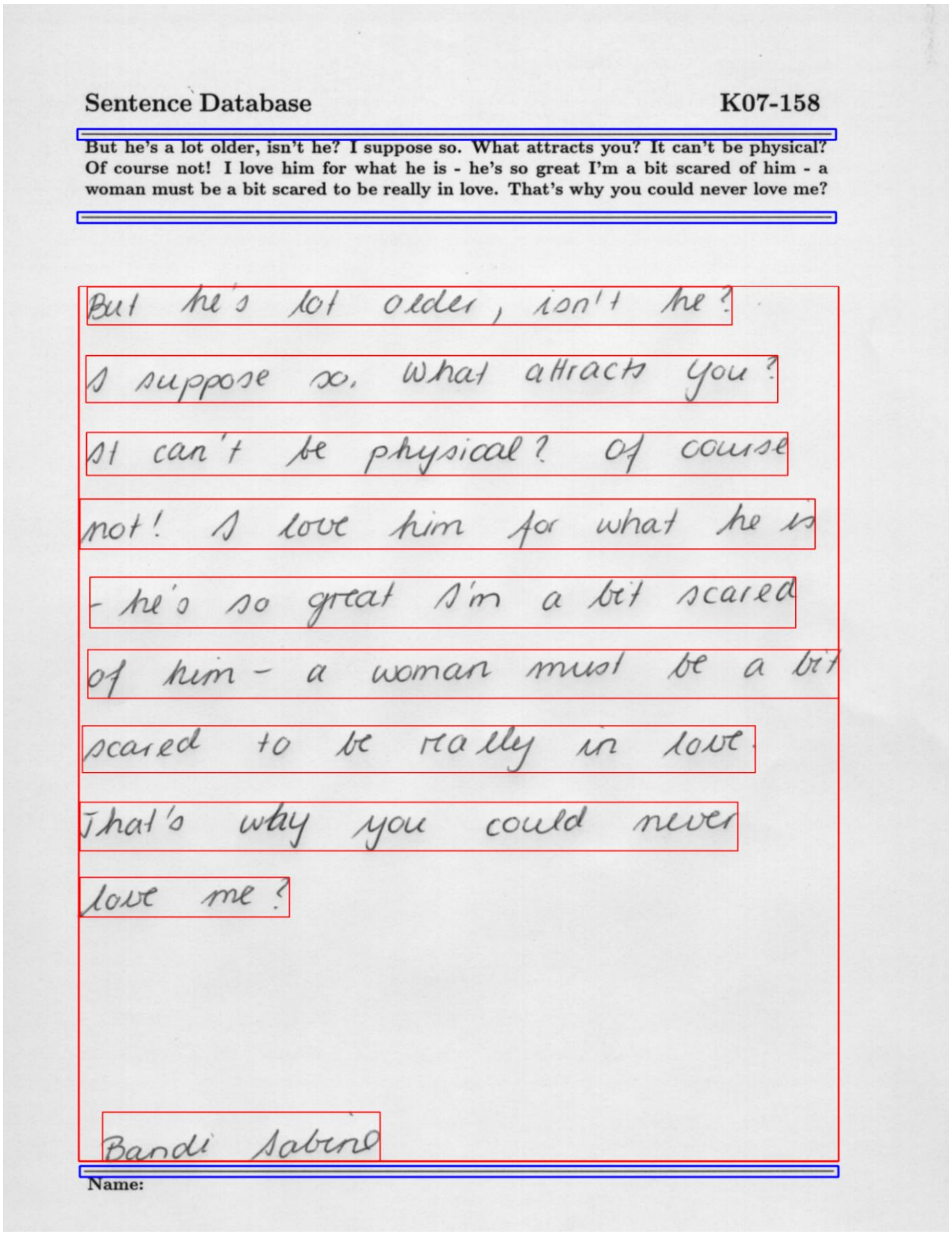


Figure 1: Horizontal histogram projection,
with the dashed line represents the threshold value, the red circles represent the peaks (lines),
and the blue squares represent the valleys.



Sentence Database K07-158

But he's a lot older, isn't he? I suppose so. What attracts you? It can't be physical? Of course not! I love him for what he is - he's so great I'm a bit scared of him - a woman must be a bit scared to be really in love. That's why you could never love me?

But he's lot older, isn't he?
I suppose so, What attracts you?
It can't be physical? Of course
not! I love him for what he is
- he's so great I'm a bit scared
of him - a woman must be a bit
scared to be really in love.
That's why you could never
love me?

Bandi Sabina

Name:

Figure 2: IAM handwritten text image showing the segmentation process.

Feature Extraction Module

This module is considered the core module of the system. The purpose of this module is to extract the most discriminant features to enable us to distinguish between different handwritings easily. As mentioned earlier, we are using a texture-based approach in the feature extraction process that can be applied on the entire handwritten paragraph. We are using one of the fastest and efficient texture descriptors in Image Processing field which is the Local Binary Pattern (LBP) descriptor.

The LBP of an image is another image of the same size where cell (x, y) is computed as

$$LBP_{P,R}(x, y) = \sum_{i=0}^{P-1} s(g_c - g_i)2^i$$

Where P is the number of neighbor pixels, g_c and g_i represent the gray level values of the central pixel and the $i\text{-}th$ neighbor pixel in a circle with radius R and centered at cell (x, y), with $s(x)$ defined as follows

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

The original LBP was defined for the neighborhood of 8 adjacent pixels (i.e. $P = 8, R = 1$).
It does not matter the order in which we are considering the neighbors of cell (x, y) as long as we are keeping this order consistent for all our computations. It also does not matter if we used function $s'(x) = 1 - s(x)$ instead of $s(x)$.

After computing the LBP of the handwriting image, we take its histogram as our feature vector.

We found that ignoring cells corresponding to white pixels in the binarized image from being counted in the histogram results in much better classification accuracies. So, we take the histogram of the LBP masked by binarized image (i.e. counting only $LBP(x, y)$ corresponding to $img_{binary}(x, y) = BLACK$). We also found that computing the LBP on the gray level handwriting image results in better accuracy than computing it on the binary handwriting image. Finally, we tested the effect of the LBP neighborhood. We found that computing the LBP on 8 adjacent cells at distance 3 results in much better and robust classification accuracies.

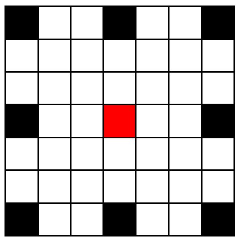


Figure 3: Our used LBP neighborhood.
(i.e. $P = 8, R = 3$)

Finally, we normalize the resulting histogram by dividing it by its mean.

The following equations summarize the feature extraction module.

$f = hist(LBP_{8,3}(img_{gray}), img_{binary})$, where $hist(img, mask)$ is the histogram of img masked by $mask$.

$f_{normalized} = \frac{f}{mean(f)}$ is our final extracted feature vector of size 256×1.

Classification Module

After feature extraction, the final thing in the system is to classify or to identify the writers of the test images. And this is the responsibility of this module.

After extracting the feature vector of every handwritten line separately from all the training text images, we pass all these feature vectors along with their corresponding writer IDs to an SVM classifier (with penalty parameter $C = 5$).

Because we carried all our tests having only 2 handwritten text images per writer, we found that separating the lines and dealing with every line as an independent training example results in better classification accuracy.

After that, we use a vote-like system to classify the test handwritten text image. For every line in the test image, we calculate the probability of this line to be belonging to each writer w_j. That is, for every line i of the test image, we calculate a probability vector $\vec{p}_i$, where $\vec{p}_{ij}$ is the probability of the line to be belonging to writer w_j. After that, we sum all these probability vectors of each line of the test image and classify the image to be belonging to the writer with the highest probability.

$writer(img) = argmax(\sum_i \vec{p}_i)$, where $\vec{p}_i$ is the probability vector of line i.

We also tried using Nearest Neighbor (NN) classifier but on the entire images (without separating the lines), it gives a very good accuracy of about 98.5%, but not as good as SVM with separated lines.

Tests Generation Module

We needed this module in our development and experiment phases. It is simply used to generate random test cases from the IAM handwriting dataset to enable us to monitor our progress in classification accuracy.

This module takes as input three parameters: N, M, and T, where:

- T is the number of test cases to be generated.
- N is the number of writers to generate per test case.
- M is the number of handwritten text image to generate per writer per test case.

In all our experiments, we fixed $N = 3$ and $M = 2$.

For each test case, we pick N different random writers from IAM dataset, then for each of those writers we select M different random handwritten text images written by this writer. Finally, we pick a different random handwritten text image for a random writer from the previously selected N writers as the test image of this test case.

Other Experimented Work

In the early stage of the project, we tested lots of algorithms and features extractions methods. We also tried a numerous number of combinations of and classifiers.

In the following lines, we are going to describe our experimented work in brief words.

GMM Model

We tried a different approach for writer identification. We modeled each writer as a Gaussian mixture model (GMM). The process starts with line segmentation. Then, to remove the effect of varying line widths caused by the usage of different pens, we have skeletonized the lines before extracting the features. Afterwards, we pass the features to train the GMM model that would be a separate distribution for each writer. We classify the new image by selecting the writer whose Gaussian distribution produced the maximum likelihood score.

The features extracted from the lines are the following:

- The position of the upper and the lower contour.
- The center of gravity of the text pixels.
- The second order moments.
- The number of text pixels.
- The number of black pixels between the upper and the lower contour

We have tried different configurations for the GMM training model parameters. The best configuration that reached the highest accuracy was setting the number of components to 2, the number of iterations to 10000.

GMM reached an accuracy of 91% for 3 writers, which is less than that of the LBP descriptor.

GSCM Properties

We also give a try to some other texture-based features based on the Gray Level Co-occurrence Matrix (GLCM), such as:

- Contrast
- Dissimilarity
- Homogeneity
- Energy
- Correlation

Actually, the GLCM features were not bad, they result in classification accuracy of about 80%. But the problem is whenever we add them along with the LBP histogram the overall accuracy decreases a lot.

Other Trials

We also tested some features related to the visible characteristics of the handwriting such as:

- The average *height* of the lines.
- The average *width* of the words.
- The average *space distance* between words and letters.
- The distance between the *upper* and *lower baselines* of the handwriting.
- The *aspect ratio* of the words and letters.
- The average *extend* of the handwriting contours (i.e. the ratio of contour area to its bounding rectangle area).
- The average *solidity* of the handwriting contours (i.e. the ratio of contour area to its convex hull area).

But unfortunately, these features result in a very poor classification accuracy.

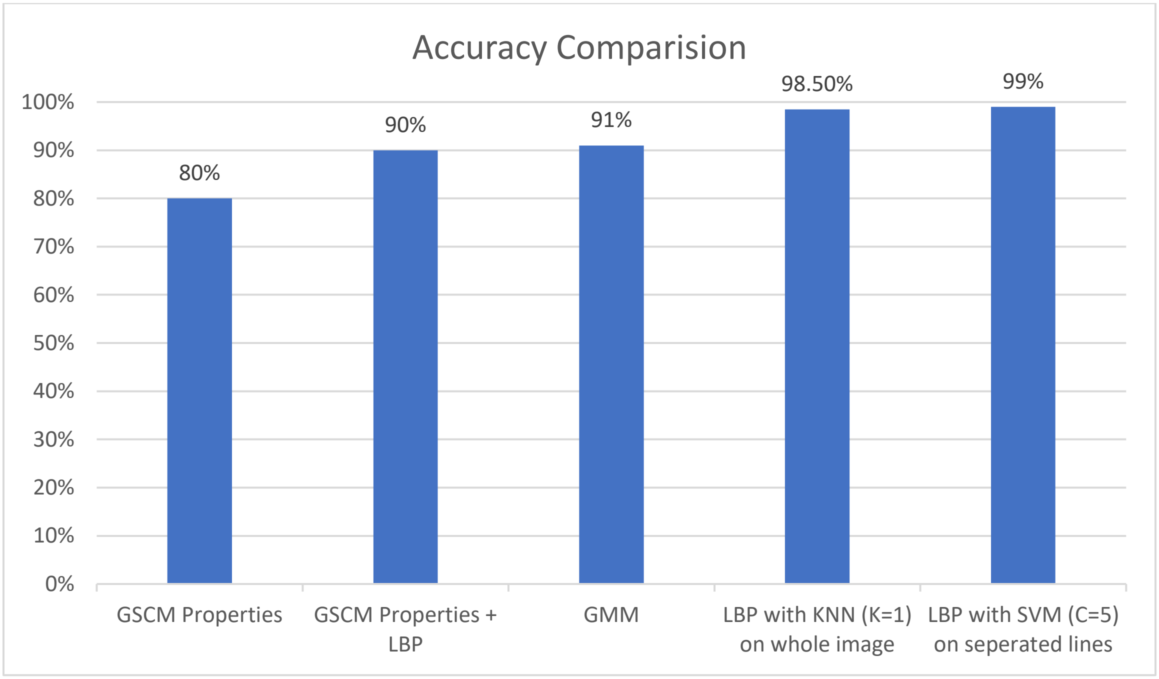
Performance Analysis & Results

We tested our system on more than 1500 randomly generated test cases, and our system managed to identify about 1485 of them correctly with an accuracy of about 99%.

$$accuracy = \frac{number\ of\ correctly\ classified\ test\ cases}{total\ number\ of\ test\ cases} \times 100\%$$

Each test case, as described above, consists of 3 different writers, where for each of those writers we are given 2 handwritten text images written by them, and finally, for each test case we are asked to detect or classify the identity of the writer of the given test handwritten text image. That is, each test case consists of 7 handwritten text images.

Even though we were constrained by using a slow programming language Python, we managed to finish identification of a test case on average time of about 1.1 seconds. That is, spending about 150 milliseconds to process an entire image of dimensions of about 2500×3500. We also managed to implement a function to calculate the LBP of an image of around $10x$ faster than that of the famous Skimage Python package.



Future Work

We are pretty sure that we reached a saturation level in classification accuracy using the LBP texture descriptor. This was so clear because adding any extra features with the LBP descriptor results in decreasing the overall accuracy.

We can completely change our approach and head to detect Morphological features which require an accurate character segmentation, which is not an easy mission.

Or instead, we can use deep neural networks, which will result in better accuracy.

Credits

Name	**Email**
Omar Wael	omar_bazaraa@hotmail.com
Ahmed Samir Hamed	asamirh.95@gmail.com
Ibrahim Radwan	i.radwan1996@gmail.com
Abdelrahman Eid	abdoeed23@gmail.com