Faculty of Engineering
Cairo University

Project report

Team '2' Members

-Sara Raafat Emil	**SEC:1**	**B.N:22**
-Abdelrahman Mohamed Badr	**SEC:1**	**B.N:32**
-Omar Salah	**SEC:2**	**B.N:1**

2020-2021

Contents

Writer identification system – overview:

In this report we will go through our Writer identification system showing the various approaches we used to achieve the best accuracy and execution time.

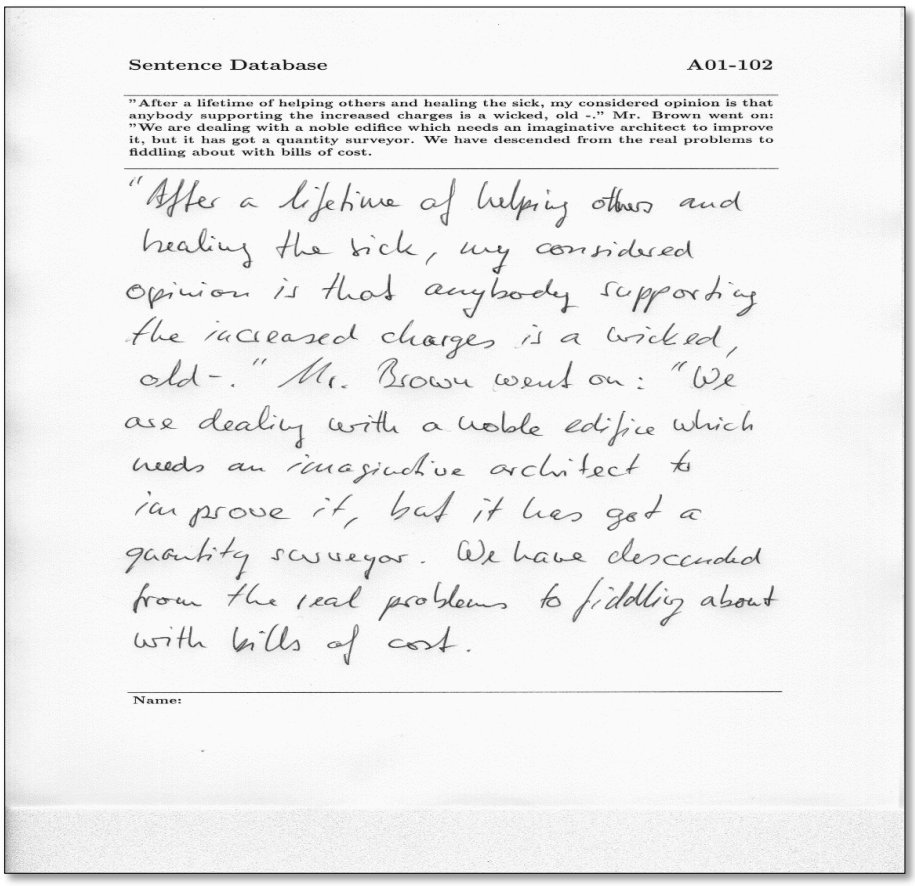
Project Pipeline

Preprocessing Module:

In this module the goal is to extract the handwritten paragraph only out of the image, then segment it into a number of separated lines, then separate each line into a number of words.

First goal: extract the handwritten paragraph:

Each image of IAM dataset has 3 long horizontal lines that separates different sections.

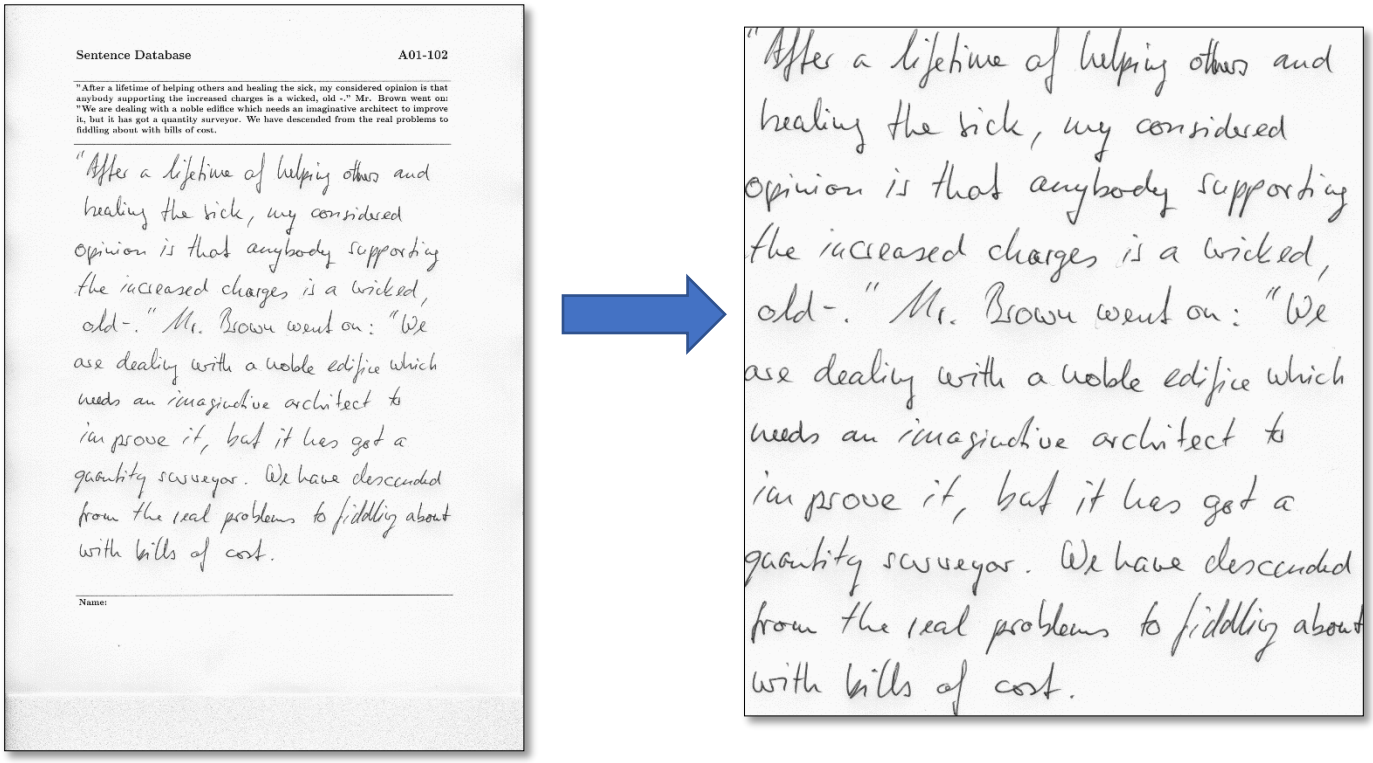

Sentence Database A01-102

"After a lifetime of helping others and healing the sick, my considered opinion is that anybody supporting the increased charges is a wicked, old -." Mr. Brown went on: "We are dealing with a noble edifice which needs an imaginative architect to improve it, but it has got a quantity surveyor. We have descended from the real problems to fiddling about with bills of cost.

"After a lifetime of helping others and healing the sick, my considered opinion is that anybody supporting the increased charges is a wicked, old -." Mr. Brown went on: "We are dealing with a noble edifice which needs an imaginative architect to improve it, but it has got a quantity surveyor. We have descended from the real problems to fiddling about with bills of cost.

Name:

The approach is to find the contours of the image, then draw a binding box around every contour, then filter them by the width of the box (we found that width of 1500 pixel is good enough), then we iteratively start determining the borders of the paragraph using histogram, until we get the borders right.



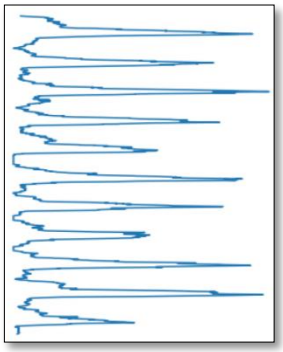
Second goal: segmentation

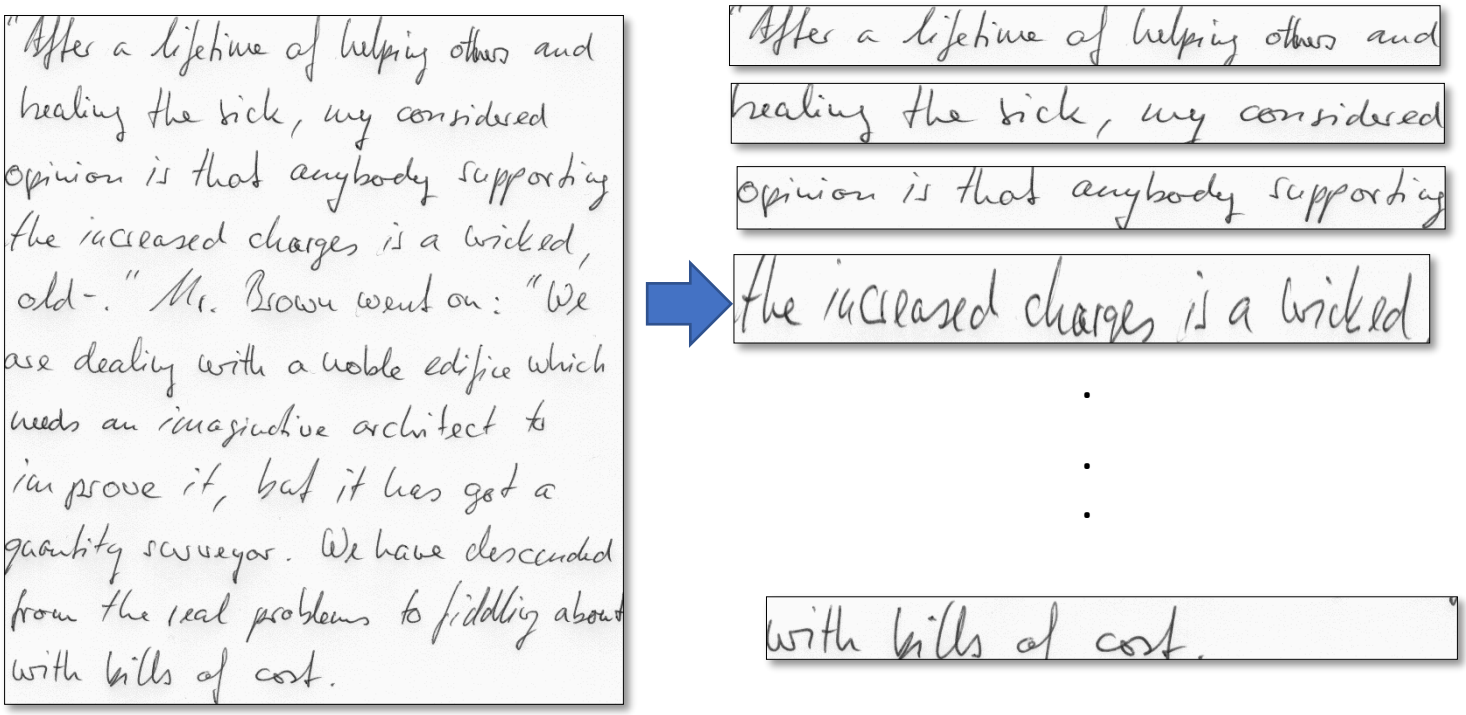
In order to extract the required features for the writer, we need to segment the paragraph into lines and words, for each type of segmentation we followed a different method:

- **Line segmentation**:

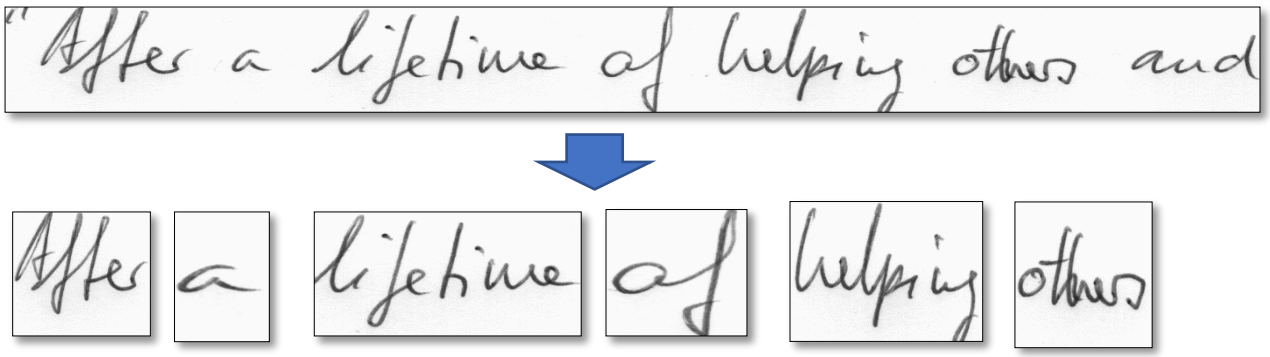
  First, we calculated the horizontal histogram of the paragraph, then detect the peaks and valleys of it to determine lines "peaks" and spaces "valleys".

  Then we loop between every 2 consecutive valleys to extract the line between them.





- **Word segmentation**:
  In order to explore which segmentation method results in more good features we also did segment each line into number of words to get more features per writer, segmenting words was directly made using vertical histogram, then defining a threshold (max/3) then segmenting each word according to this threshold.



Naturally some words might not be segmented right but it's ok since the average number of words per paragraph is about 60 word.

Feature Extraction Module

Good accuracy means good features. That is the approach we follow.

After reading some papers and surveys about this field we found two types of features:

- **Visible characteristics of the writing**: such as the height of the three main writing zones, the width of the characters, the slant of the writing, and the text's legibility. These features can also be observed and interpreted by humans.
- **Texture characteristics**: such as pattern of edges, corners and flat area.

After reading about both and see the final accuracy we decide to use texture approach because its result was better, seems to be faster and easier for implementation.

In a survey, it shows some techniques one of them is Local binary pattern (LBP).

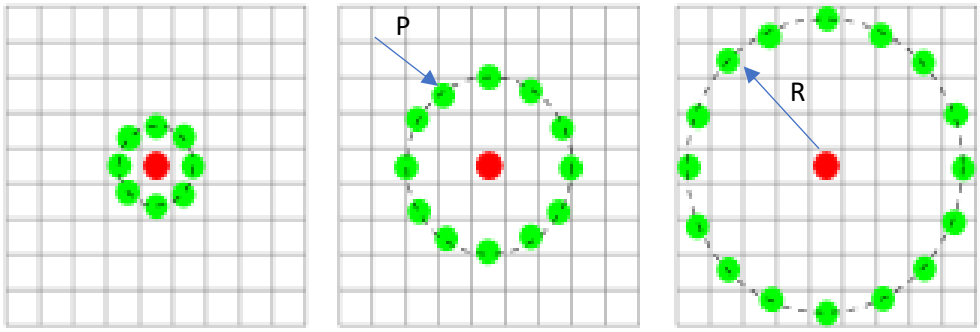
Local binary pattern (LBP):

LBP is a type of visual descriptor used for classification. It has since been found to be a powerful feature for texture classification. It has several improvements of the original LBP.

The original LBP:

The LBP feature vector is created in the following manner:

- For each pixel in a texture, compare the pixel to each of its **P** neighbors that Follow the pixel along a circle with radius **R**, i.e., clockwise or counter-clockwise.
- Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number, which is usually converted to decimal.
- It has 2 important parameter which are P and R. P is the number of neighbor pixels to be checked. R is the radius of a circle that define the distance between central pixel and neighbors



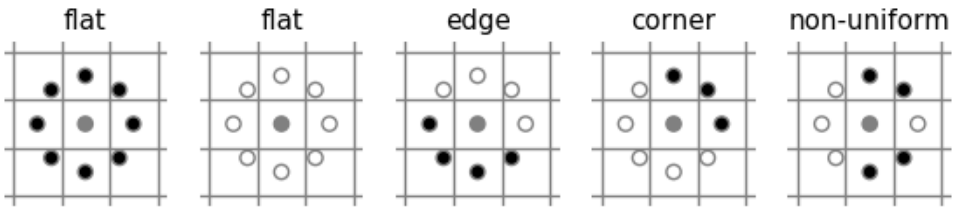
One of the useful extensions of the original operator is the so-called The LBP uniform.

The LBP uniform:

This idea is motivated by the fact that some binary patterns occur more commonly in texture images than others. A local binary pattern is called uniform if the binary pattern contains at most two 0-1 or 1-0 transitions. For example, **00010000 (2 transitions) is a uniform pattern, but 01010100 (6 transitions) is not**.

Using uniform patterns, the length of the feature vector for a single texture reduces from 256 to number P+1.

The result only includes patterns where all 1s are adjacent and all 0s are adjacent. *All* other combinations are labeled 'non-uniform' and added to the same bin.



In our work we use LBP uniform to reduce feature vector and get almost the same result in less time.

We tried different P and R put finally choose P = 8 and R =3, because increasing them doesn't enhance accuracy or average run time.

We tried different P and R values for the same test set and found that:

Using **P=8, R=1** resulted in approximately **94%** accuracy with average time **5.2** s.

Using **P=8, R=2** resulted in approximately **95%** accuracy with average time **5.1** s.

Using **P=8, R=3** resulted in approximately **95%** accuracy with average time **4.87** s.

Using **P=16, R=3** resulted in approximately **99%** accuracy with average time **7.28** s.

Model Training Module.

After feature extraction, the following step is the classification of the writer of the test images. We tried 3 different classifiers: Linear SVM, Non-linear SVM, and KNN.
For every image in dataset, after we extract its features, we append it to the training data and the labels are already extracted from the dataset. For every image in the test data, the output of the feature extraction is passed to the already trained model. Now we have a predicted writer for each word in the image ,we decide the id of the writer based on the most frequent id got on all the image words.

Linear SVM:

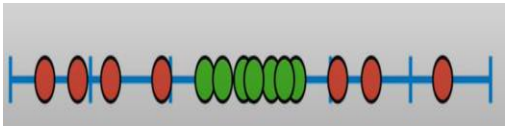
In general, the best SVM algorithm should maximize the distance between the hyperplane and the training data points.

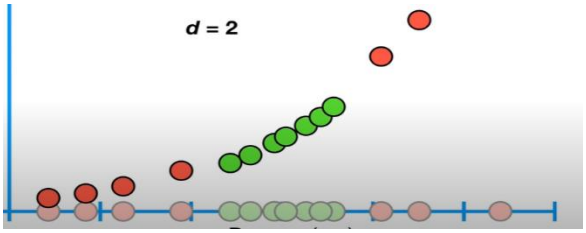
Data is classified with the help of hyperplane but The assumption of linear separability in data produced a less accurate model which was around 97% accuracy.

In the SVM function, we explore many parameters, one of them is *C* which is the penalty parameter of the error term, it controls the tradeoff between smooth decision boundary and classifying the training points correctly. As we increased the value of C, the model showed a better accuracy but a higher training time.

Another parameter is ***gamma*** where the higher the gamma value it tries to exactly fit the training data set, the default gamma showed a very bad accuracy 35 %. (The default of gamma is computed as 1 /n_features)

Non-Linear SVM:





We use Kernels to make non-linear space into a linear space, because data must be mapped into a higher dimensional space to get classified. We tried the sigmoid and poly kernels, the **poly kernel** showed much better results which was 99% accuracy for the model. As for the **degree** of the polynomial, we left it as default which is 3.

Parameters C and gamma discussed above also apply on the non-linear SVM.

KNN:

KNN takes K nearest neighbors to decide where the new data point with belong to. We tried different K's :1 and 5 and both showed around 60 % accuracy which is much worse than SVM. For the distance, we used the default which is the Euclidean distance (the square root of the sum of squared distance between two points).

Performance Analysis Module.

Using the IAM dataset, we generated a test set that contains 300 test set, each has 3 writers, each writer has 2 images to train the model.
We used 3 models to test them on the generated test set.

1st model: Linear SVM:

First, we fixed the segmentation mode to lines, P=8, R=3, we obtained the following results:

Model used	LBP parameters	Number of tests	Segmentation mode	accuracy	Avg time per test (seconds)
Linear SVM (C=100,max_iter=2000)	P=8 R=3	14	Lines	92.86%	4.68
Linear SVM (C=100,max_iter=2000)	P=8 R=3	**100**	Lines	90%	4.69
Linear SVM (C=100,max_iter=2000)	P=8 R=3	**300**	Lines	90.67	4.11

We found out that with a small number of test cases the accuracy wasn't that good, and it dropped after increasing the number of test cases.

Then we used Segmented words instead of lines, we obtained the following results:

Model used	LBP parameters	Number of tests	Segmentation mode	accuracy	Avg time per test (seconds)
Linear SVM (C=100,max_iter=2000)	P=8 R=3	14	words	100%	3.48
Linear SVM (C=100,max_iter=2000)	P=8 R=3	**100**	words	95%	3.89
Linear SVM (**C=5**,max_iter=2000)	P=8 R=3	100	words	89%	3.47
Linear SVM (**C=50**,max_iter=2000)	P=8 R=3	100	words	95%	4.3

Linear SVM (C=100,max_iter=2000)	P=8 **R=1**	100	words	94%	4.5
Linear SVM (C=100,max_iter=2000)	P=8 **R=2**	100	words	95%	4.3
Linear SVM (C=100,max_iter=2000)	**P=16** R=3	100	words	98%	8.14
Linear SVM (C=100,max_iter=2000)	**P=32** R=3	100	words	98	13.44
Linear SVM (C=100,max_iter=2000)	P=8 R=3	200	words	95%	3.93
Linear SVM (C=100,max_iter=2000)	P=8 R=3	300	Words	94.67	3.16

Here we also tried different P and R, we highlighted the best combinations which resulted in best results.

2nd model: non-linear SVM

Model used	LBP parameters	Number of tests	Segmentation mode	accuracy	Avg time per test (seconds)
Non-Linear SVM (C=100,kernel='sigmoid')	P=16 R=3	100	words	58%	5.89
Non-Linear SVM (C=100,**kernel='linear'**)	P=8 R=3	100	lines	93%	4.42
Non-Linear SVM (C=100,kernel='linear')	P=8 R=3	100	**words**	98%	3.73
Non-Linear SVM (C=100,kernel='linear')	**P=16** R=3	100	words	97%	5.57
Non-Linear SVM (C=100,**kernel='poly'**)	P=16 R=3	100	words	99%	6.3

We decided to go with this **model** as the main model to run the system.

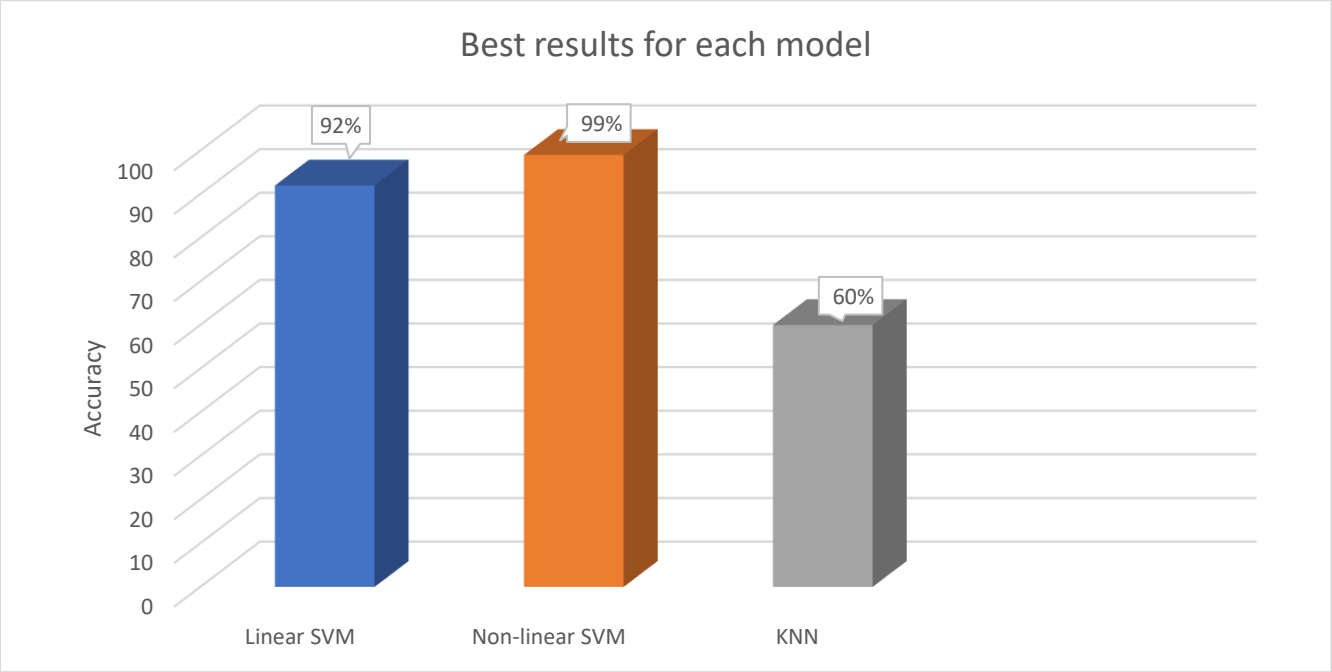
3rd model: K - nearest neighbor

Unsuccessful trial

Model used	LBP parameters	Number of tests	Segmentation mode	accuracy	Avg time per test (seconds)
Knn(n_neighbors=1)	P=8 R=3	100	words	60%	4.1
Knn(n_neighbors=5)	P=8 R=3	100	words	60%	5

The results speak for themselves 🤷.

The above results can be summed in this graph:



Other developed modules

In the process of developing our system we needed some utility modules to create test sets and to measure the performance of the system.

Creating test set module:

We developed a python script that creates any number of tests according to the following algorithm:

- Read the 'forms.txt' file which has the meta data for the IAM dataset
- Remove any writer that has less than 3 papers
- For each test set:
  - Select three random writers
  - For each writer:
    - Select 2 images to train the model
  - Finally select randomly one test image from one of the three writers 'to use the model to predict it belongs to which writer'
  - Add the writer id of the test image to 'validation.txt' file 'to be a reference to compute accuracy'

Complete implementation of this module can be found in file 'creating tst.py'

Run module

This is the module used to run the system on any given test set.

Complete implementation of this module can be found in file ‘run.py

Calculating the accuracy/AVG time module

In this module we work on three files

- Validation.txt: contains the correct id for the test image of each test set
- Results.txt: contains the predicted id for the test image of each test set
- Time.txt: contains the time taken to train the model and predict the id of each test set

By comparing the first 2 files we can obtain the accuracy of the system, and for the third file we can calculate the average time taken to finish 1 test set.

Complete implementation of this module can be found in file ‘calc_accuracy_time.py

Work distribution

Name	Work
Sara Raafat	• Training model module • Performance analysis module
Abdelrahman Mohamed Badr	• Feature extraction module • Calculating the accuracy/AVG time • Performance analysis module
Omar salah ali	• Preprocessing module • Performance analysis module • Creating test set module • Run module • Documentation