Building Classifier Models

by Edy Widihantoro

Submission date: 08-Oct-2024 11:45AM (UTC+0700)
Submission ID: 2478764420
File name: 2437_8_944E227.pdf (2.07M)
Word count: 8853
Character count: 46242

Building Classifier Models for on-off Javanese Character Recognition

Lucia D. Krisnawati
Universitas Kristen Duta Wacana
Jogjakarta, Indonesia
krisna@staff.ukdw.ac.id

Aditya W. Mahastama
Universitas Kristen Duta Wacana
Jogjakarta, Indonesia
mahas@staff.ukdw.ac.id

ABSTRACT

In this paper, we demostrated the building process of four classifier models as a part of an on-off character recognition system for Javanese characters. As Javanese character is no longer used in everyday writing and books, the dataset were collected by scanning the historical manuscripts and a reading lesson book. The rough dataset comprises 15.414 annotated characters and 633 classes. However, only 162 classes have sufficient data samples to be the training and testing one. Using this dataset, we measured the performance of four classifiers, namely k-NN, LDA, SVM, and Gaussian NB on the accuracy, micro-averaged precision, micro-averaged sensitivity and weighted-averaged precision and sensitivity metrices. The experiment shows that k-NN outperforms any other classifiers almost in most metrices, while SVM suffers the poorest performance. The research byproduct worth mentioning here is that it has identified 633 classes of distinct Javanese characters which comprise both common characters and compound characters found in modern Javanese writing as well as the archaic characters found in the literary works only.

CCS CONCEPTS

• **Applied computing** → **Optical character recognition**; *Document capture.*

KEYWORDS

Optical Character Recognition, Javanese Characters, Linear Discriminant Analysis, Support Vector Machine, k-NN

ACM Reference Format:
Lucia D. Krisnawati and Aditya W. Mahastama. 2019. Building Classifier Models for on-off Javanese Character Recognition . In *The 21st International Conference on Information Integration and Web-based Applications & Services (iiWAS2019), December 2–4, 2019, Munich, Germany.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3366030.3366050

1 INTRODUCTION

Optical Character Recognition (OCR) has been applied in various fields such as in writer identification for forensic research [7], indexing and searching the archival documents and manuscripts [16], or in recognizing the vehicle lisence plate [2]. Massively, OCR has been applied in digitization projects of recent and early printed books for the sake of building e-library [19, 20]. The projects of preserving the content of historical manuscripts, books, or newspapers implement character recognition on various writing systems such as Latin alphabets [17], Tangut [13], Thai, Bangla and Latin [22], Arabic alphabets [10], and Javanese characters [6, 8, 15] as well.

As an active area of research, character recognition inclines to be a part of OCR building blocks. OCR is a field of study which deals with converting digital images of text into editable documents that can be processed, edited, searched, saved, or copied using a computer application [3]. To achieve this broad and complex task, the pipeline of OCR system is conventionally decomposed into the following sequences of processing: image acquisition, preprocessing, segmentation, feature extraction, character recognition and post-processing [5, 9]. The digital images of text are acquired by either scanning or photographing, while the preprocessing phase covers the noise reduction, binarization, and layout analysis [20]. The so-called character recognition very often falls into a task of classification which assigns an inputted character into a predefined class of characters. The post-processing phase deals with the correction of the recognition errors in order to increase the accuracy rate.

The main goal of the OCR research is to gain the high rate of recognition accuracy which remains a challenging problem. In OCR researches, this problem is addressed differently since each step of OCR pipeline contributes to the success of the recognition process. Additionaly, the text genre such as historical manuscripts or modern well printed texts present different contrainsts. The historical texts tend to have a greatly degraded image quality [16]. Therefore, some researches addressed the preprocessing problems such as noise removal and geometric correction [4], problem of text, line, and character segmentation [14, 17], finding a robust feature extraction method for historical handwritten texts as found in [22], the problem of recognition by word matching [19] or classification models [8], or correcting the recognition error in the post-processing step as [1, 3].

In this study, we investigated the performance of several classifier algorithms to recognize Javanese characters. The use case of our OCR system is that it receives an input in the form of a text image online. In the experiment context, the text images were acquired by scanning text taken from *Serat Mangkunegaran IV*, which is a Javanese book written at the end of 19^{th} century. The system proceeds its task by gray scaling and binarizing the text image, segmenting line and characters, and extracting each of the inputted character features. The stream of character features become the

iiWAS2019, December 2–4, 2019, Munich, Germany

ACM ISBN 978-1-4503-7179-7/19/12…$15.00
https://doi.org/10.1145/3366030.3366050

input to the classifiers which should be able to predict to which class each of these characters belongs. The recognition outputs will be displayed online. The problem we faced is that we have very limited annotated character instances in each class of characters as training data. Therefore, we compared the performance of the following classification models: K-Nearest Neighbor (k-NN), Gaussian Naive Bayes (GNB), Linier Discriminant Analisys (LDA), and Support Vector machine (SVM).

The remaining part of this paper is organized as follows: Section II highlights the previous works related to this study, while Section III presents the brief overview on the Javanese characters. Section IV deals with the concepts of classification models applied in this study. Section V describes the pipeline of the proposed methods and its architecture. The strategy for building the data set, the experiment scenario and the evaluation on each classifier performance will be presented on Section VI. The last section deals with the conclusion of this study.

2 RELATED WORK

Based on the way a text is written, OCR researches focus on either printed or handwritten character recognition. This option is based on the more complex application which applies the OCR results. For example, digitizing projects for the sake of building e-library tend to focus more on the printed character recognition [20], while the projects on writer identification in historical texts incline to delve deeper on the handwritten one [7, 22].

Based on the way the character input is received, the OCR system could be categorized into an online character recognition (OLCR) or offline character recognition (OFCR). The significant difference between these systems is that OLCR processes temporal data using a list of points representing the track of writing [10], while OFCR processes an image as input which has been digitized, stored, and processed by the OCR system. This dichotomy gives no place to an OCR system accessible through Internet and receives text images as input online and delivers its output online but performs its recognition offline. Such system differs from the one that combines OLCR and OFCR as found in [10, 18]. From this point forward, we would like to address such system as an on-off character recognition system.

Related to OLCR and OFCR, Hamdani et. al. [10] combined the offline and online systems at the feature level. They applied beta-elliptic approach in capturing the spatio-temporal information in the feature extraction step for offline handwritten recognition process. The classifier was built using HMM-toolkit provided by Mathlab [10]. Surinta et. al. [22] built an offline handwritten character recognition which applied k-Nearest Neighbours (k-NN) and Support Vector Machine (SVM) as classifiers and highlighted the use of local gradient feature descriptor on the feature extraction phase. Focusing more on the automated layout analysis, Reul, et al. [10] built an offline OCR system using OCRopus library for recognizing early printed books from 15^{th} century.

In recognizing characters, the most common approach used is classification with its various learning models. However, classification is not the only option for recognizing character as shown in [16, 19]. Instead of classification, they utilized word spotting and word matching by means of Dynamic Time Warping (DTW) which aligns and compares sets of features extracted from two images of words [19]. Having sparse data training, Liu [13] implemented Principle Component Analysis (PCA) based classifier to recognize Tanggut characters. Meanwhile, Namsyl and Konya [17] combined deep learning (CNN-LSTM models) with synthetic training data generation and data augmentation technique to build a segmentation free OCR system. They claimed that their system architecture can be used to recognize printed, handwritten, and scene texts [17] as well.

Using classifiers for recognizing characters, most researches on OCR for Javanese characters tend to train their prototypes with the *incomplete* Javanese characters as dataset. Included in Abugida writing style, Javanese script has 20 basic consonants carrying inherent vowel 'a' called *Nglegena*. Other vowels and consonant cluster builders are written as diacritics which results in a compound character. Unfortunately, there has been no definite study stating the exact number of Javanese characters including the compound ones. A study in [24] assumed that the total number of Javanese characters is ca. 103 by excluding the compound characters. However, Unicode provides 91 code points including numbers and punctuation marks for Javanese characters.

In recognizing Javanese characters, Budhi and Adipranata [8] applied several ANN classifiers trained with 31 distinct characters along with 20 samples for each character class, while Dewa et. al. [6] applied CNN and Multilayer perceptron (MLP) to be tested with handwritten *Nglegena*. Widiarti and Wastu [24] developed an HMM-based classifier model for recognizing *Nglegena* too. As data set, they took 1000 characters consisting of 50 samples for each *Nglegena* character. Akin to Dewa et al., Karundeng et al. [11] made use handwritten *Nglegena* as much as 3379 samples to be trained and tested to their SVM-based classifier. In contrast to these Javanese OCR systems, this study made use of 162 classes of Javanese characters with 10 samples for each class. These classes comprise Nglegena, diacritics, numbers, punctuation marks, and some compound characters. In total, our dataset consists of 1620 characters.

3 JAVANESE CHARACTERS IN A NUTSHELL

Javanese is considered to be one of world's classical languages with literary tradition over a thousand years. Javanese has been written in Javanese script which is an Abugida type – a segmental writing system in which consonant-vowel sequences are written as a syllable unit [12]. In total, there are 20 basic consonant-vowel sequences as displayed in Figure 1. In Figure 1, the *Nglegena*, well knowkn also as *carakan*, are placed on the rows with the heading *aksara*, their consonant cluster builders are put on rows with the heading *Pasangan*. It displays their transliteration in Latin alphabet found under the heading *Nama*. The sign ©shows the position of *Carakan* and whether *Pasangan* should be written under *Carakan* functioning as diacritics or on the right side of *Carakan*.

Javanese texts are written from left to right and recognize no word boundary (*Scriptio Continua*). The presence of *pasangan* might indicate the first syllable of another word, yet it is not always the case. The main function of *Pasangan* is to start a succeeding syllable by making a *Carakan* preceeding it to be a consonant, i.e it nulifies the vowel of its preceeding syllable and thus make it a

Nama	Ha	Na	Ca	Ra	Ka
Aksara	ꦲ	ꦤ	ꦕ	ꦫ	ꦏ
Pasangan	◌ꦲ	◌ꦤ	◌ꦕ	◌ꦫ	◌ꦏ
Nama	Da	Ta	Sa	Wa	La
Aksara	ꦢ	ꦠ	ꦱ	ꦮ	ꦭ
Pasangan	◌ꦢ	◌ꦠ	◌ꦱ	◌ꦮ	◌ꦭ
Nama	Pa	Dha	Ja	Ya	Nya
Aksara	ꦥ	ꦝ	ꦗ	ꦪ	ꦚ
Pasangan	◌ꦥ	◌ꦝ	◌ꦗ	◌ꦪ	◌ꦚ
Nama	Ma	Ga	Ba	Tha	Nga
Aksara	ꦩ	ꦒ	ꦧ	ꦛ	ꦔ
Pasangan	◌ꦩ	◌ꦒ	◌ꦧ	◌ꦛ	◌ꦔ

Figure 1: The basic character sequences of Javanese (Nglegena) along with their consonant cluster builders (Pasangan)

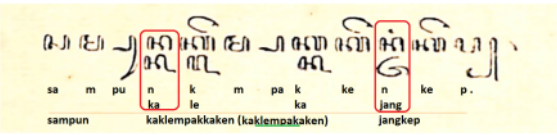

Figure 2: An example of writing 3 words at the end of a setence in Javanese characters

closed syllable. Figure 2 highlights *Pasangan* 'ka' and 'ja' in red boxes, which introduce the first syllables of the following words, while *pasangan* 'ka' in 'kka' starts its successive syllable only. The consonant cluster builder within a syllable is written by a group of diacritics called *Sandangan* which is written either on the left, right, above or continuously attached under the *Carakan (Nglegena)*. Figure 3 gives examples on how to write some consonant cluster builders within a syllable. Other vocals such as *o, i, u, è, ê* are treated as diacritics, but they will be written as special compound characters when they are used to indicate vowel-initial syllables or to write loanwords.

Interistingly, Javanese writing system has morphographemic rules to mark the boundary of a free morpheme when it is followed by a bound morpheme (suffixes) in the form of a vowel-initial syllable. The rule requires to write the suffix using the coda (last consonant) of its preceeding syllable as its onset. This results in the presence of twin consonants. To make it clear, the second word in Figure 2 depicts a morphographemic rule which is used to mark the boundary of a suffix to its root word. Literary transliterated, the complex word *kaklempakaken* with the base word *klempak* – which is underlined – has to be written *kaklempakkaken* in which the double graphemes 'kk' indicate the presence of a suffix *aken*.

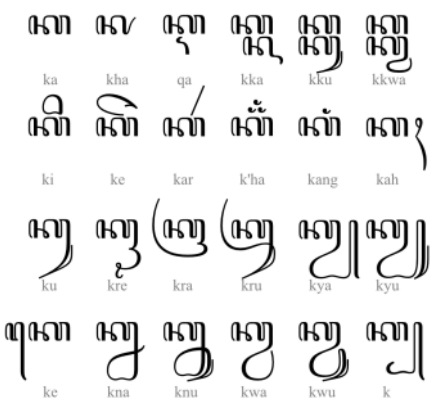

Figure 3: Some examples on characters functioning as consonant cluster builder within syllables and thus form inseparable compound characters

With such writing system, it remains a question left: how many distinctive characters the Javanese script has. To answer this question, we relied on the output of the segmentation process reported in [15]. The rationale is that there has been no study explicitly answering this question. The 91 blocks provided by unicode or a set of 103 characters mentioning in [24] list only character categories which are combinable to form compound characters. However many compound characters inseparable as shown in Figure 3. The significance of having the number of distinctive characters is to obtain the number of classes to classify. Using 72 scaned pages which resulted in 19.112 segments of characters, Mahastama and Krisnawati [15] identified that there are 633 distinctive characters which include numbers and punctuations as well.

4 CLASSIFICATION MODELS

This section presents some classification models that have been implemented in this study. It starts with brief description on k-Nearest Neighbors (k-NN), followed by Naive Bayes classifier and Linear Discriminant Analysis (LDA). The Support Vector Machine ends the discussion on the experimented classifier concepts.

4.1 K-Nearest Neigbors

Well known as a simple algorithm in Machine Learning (ML), k-NN is categorized as a nonparametic and instance-based learning method. It is nonparametic in the sense that it does not assume data distribution [5], the data points themselves become the parameters. As an instance-based method, k-NN's learning process is simply storing all instances of the training data. When a new instance or test data point is encountered, it will be matched to all instances in the training data.

In k-NN, the training phase is done by adding each sample instance to the list of *data samples*. Given an instance x_q, the k-NN classification algorithm will compute the distance of x_q to all data

samples and sort the distances in ascending order. Then, it discriminates k number of training instances, x_i, which are nearesst to x_q. distance metric applied is the Minkowski distance, which is a generalization of both Euclidean and Manhattan distances. The Equation 1 shows how to compute the Minkowski distance which is commonly calculated with p being 1 or 2 . If p equals to 1, then it corresponds to the use of Manhattan distance, while p equals to 2 refers to the use of Euclidean one.

$$d_{Mk} = \sqrt[p]{\sum_{i=1}^{d} |P_i - Q_i|^p} \tag{1}$$

where P denotes to parameters, and Q to queries.

The class prediction of x_q is done by voting and the class of an instance with the maximum votes in k neigbors is taken as the prediction. The computation of a class prediction is presented in Equation 2. It is clear that the function $\hat{f}(x_q)$ is assigned the value of $f(x_i)$ where x_i is the training instance nearest to x_q with the maximum voting, v, where $\delta(a, b) = 1$ if $a = b$ and 0 otherwise.

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i)) \tag{2}$$

With such algorithm, one of k-NN drawbacks is that it requires high memory for storing and load all training data in order ot make a prediction. The consequence is that testing k-NN becomes slow and computationally expensive. This means that it is inappropriate for real-time application [5] but many researches reported that it has fairly high classification performance. Therefore, k-NN functions as good benchmarks for evaluating other classification models.

4.2 Naive Bayes Classifier

Naive Bayes classifier is included in probabilistic models as it computes both prior and posterior probabilities of the classes when it is given an input data whose class needs to predict. It is also identified as a generative classifier which learns a model of the joint probability, $p(x, y)$ of the inputs x and the label y [21]. In making prediction, the Bayesian rules are used to calculate $p(y|x)$ in order to pick the most likely label y. To calculate a class's prior probability, the distribution of features and parameters need to be estimated. Depending on how these parameters are estimated, the Naive Bayes classifiers have several configurations such as Gaussian Naive Bayes (GNB), Bernoulli Naive Bayes, or Multinomial Naive Bayes. Since this study uses Gaussian distribution, the discussion will be focused to GNB.

As in other Bayes classifiers, the computation of posteriori probability in GNB uses Bayes formula as shown in Equation 3. It can be seen that in order to compute the posteriori probabilities, $P(c_i|x)$, it requires a prior probability $P(c_i)$ and the class-conditional probability distribution $p(x|c_i)$, where i=1, ..., N predefined classes. The class prior probability $p(c)$ is computed by dividing the number of instances of class c with the total number of instances in the dataset.

$$\begin{aligned} P(c_i|x) &= \frac{P(c_i)p(x|c_i)}{p(x)} \\ &= \frac{P(c_i)p(x|c_i)}{\sum_{j=1}^{M} P(c_j)p(x|c_j)} \end{aligned} \tag{3}$$

Then, the question is on how to compute the class-conditional probability. It is done by the help of Gaussian distribution as shown in equation 4.

$$P(x_i|c) = (2\pi\sigma^2_{x_i,c})^{\frac{1}{2}} * exp\left(-\frac{(x_i - \mu_{x_i,c})^2}{2\sigma^2_{x_i,c}}\right) \tag{4}$$

where μ refers to the mean, and σ signifies the training sample variance. The Equation 4 requires to compute the mean and variance first before computing the class-conditional probability.

4.3 Linear Discriminant Analysis

In spite of its name, Linear Discriminant Analysis (LDA) belongs to generative and parametric classifiers [5] instead of discriminative ones. Therefore, it considers the problem of predicting a class label y (prior probability) on the basis of a vector of features x_i, as in the Naive Bayes classifier. The generative approach assumes that $P(Y = 1) = P(Y = 0) = \frac{1}{2}$ and the conditional probability of X given Y is also computed by Gaussian distribution as in GNB [21]. Another assumption is that each feature or dimension has the same variance σ. The prediction of input data x is computed by applying Bayes' rule written as in Equation 5.

$$h_{Bayes}(x) = \underset{y \in \{0,1\}}{\operatorname{argmax}} P(Y = y)(P(X = x|Y = y)) \tag{5}$$

As Shwarzt and Ben-David wrote in [21], the log-likelihood is needed in order to predict $h_{Bayes}(x) = 1$. The Equation 6 shows how to compute this likelihood ratio.

$$llr = \frac{1}{2}(x - \mu_0)^T(x - \mu_0) - \frac{1}{2}(x - mu_1)^T(x - \mu_1) \tag{6}$$

Unlike the Gaussian NB, the main goal of LDA-based classifier model is to project the original dataset in d-dimensional onto a lower dimensional subspace, k-dimensional, where $k < d$. To achieve this goal, the computation of LDA is generally performed into the following steps:

(1) The first step is to compute the distance between the means of different classes from the dataset, which is called the *between-class matrix*
(2) The second step is to compute the within-class matrix which is a calculation of the distance between the mean and the samples of each class [23]. Both between-class and within-class matrices are addressed as a scatter matrix, W.
(3) The next step is to calculate the eigenvectors (V) along with their eigenvalues (λ) for matrix W.
(4) Then it proceeds with sorting the eigenvectors in descending order; the first k eigenvectors is used as a lower dimensional space (V_k) [23].
(5) The last step is to project all samples (X) onto the new subspace.

It is reported that LDA is a good model for dimensionality reduction and for the existence of some unlabeled dataset. However,

Tharwata et. al. [2] reported that LDA suffers from small sample problem. It easily fails to find the lower dimensional space if the dimensions are much higher than the number of samples in the data matrix.

4.4 Support Vector Machine-based Classifier

As a type of hyperplane classifier, Support Vector Machine (SVM) has been widely implemented for pattern recognition, included OCR, since the mid of 1990s has contributed to countless state-of-the-art performances [5]. The main goal of SVM algorithm is to compute the maximal margin, or put it in a simple way, to draw a margin of some width to the nearest data points. The margin, which is a gap between the two closest class points, is closedly related to a hyperplane which separates data points having different class. The so-called support vectors are the data points which are closest to the margin of a hyperplane.

For binary classification, the decision function is computed by Equation 7.

$$f(x) = w^T x + b \tag{7}$$

where w is a weight vector, and b is a bias (also called as threshold). The distance between a point x and the hyperplane is defined as $w^T x + b = 0$. The margin obtains its largest distance to the positive class if $w^T x + b = +1$ and to the negative ones by $w^T x + b = -1$. The search for the maximum marginal hyperplane in SVM is done in 2 steps:

(1) Generate hyperplanes which segregates the classes in the best way in an iterative way. The goal is to minimize the error.
(2) Select the hyperplane with the maximum margin

In practice, the SVM algorithm is implemented with a kernel which takes a low-dimensional input space and expands it into a higher dimensional space. This is done by using a kernel trick. There are several kernels in SVM, such as linear kernel, Radial Basis Function (RBF), or polynomial kernel. The linear kernel is good for binary classification while RBF-kernel projected to deal with the muti-class classification. The linear kernel function is defined in Equation 8, while the RBF-kernel is computed with the Equation 9.

$$k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i).\varphi(\vec{x}_j) \tag{8}$$

where $\varphi(\vec{x}_i)$ is the feature vector in the expanded feature space and may have infinite dimensionality [5].

$$k(\vec{x}_i, \vec{x}_j) = exp(-\gamma \|\vec{x}_i - \vec{x}_j\|^2) \tag{9}$$

where γ is a parameter ranging from 0-1. The higher value of γ will fit the data training, however it may cause an overfitting due to the increase of the number of support vectors [22].

5 THE PROPOSED METHOD

The highlight of this research is to build some classifier models and to experiment them on the data to get which model is the most suitable to our data. However, the recognition process could not stand alone but it should be done along other processes in OCR pipeline. Therefore, the flow of the processes in this research is described in Figure 4.

From this Figure, it can be seen that the system receives an input in the form of a scanned or photographed page of a book, text, or manuscript, or simply a text captured by the smartphone. This text image is then preprocessed. The segmentation process is performed on the level of lines and charaters. After having individual character per page, the process proceeds with extracting the features of each character. This process was applied to both training and testing process. The feature vectors of characters are then saved in a database as training data, while the feature vectors from the tested text image is used for doing a match in a classification way. After being trained, the classifiers would be validated and evaluated.

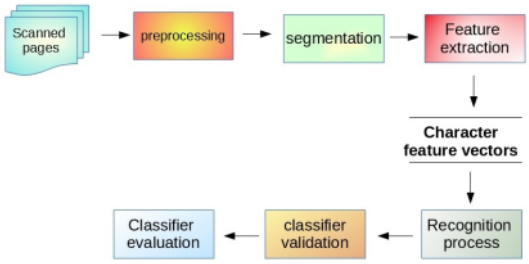

Figure 4: The architecture of the proposed system

5.1 Preprocessing Phase

Assuming that the input image has no geometric distortion and in relatively good condition, the preprocessing covers two tasks, i. e. the image gray-scaling and the background removal. The conversion of the input image into a grayscale one was implemented by using the function provided by opencv library in Python. In cv2, the function to load images in grayscale mode is callable through *cv2.IMREAD_GRAYSCALE*.

The background removal is a process which isolates the pixels forming the foreground, which is mainly text, from the pixels forming the rest of the image elements or the background. The foreground pixels are commonly marked with the integer 1, while the background pixels will be marked as 0. To remove the background, we simply applied the thresholding function provided by cv2 library in Python which is callable through *cv2.threshold()* syntax. We used the threshold $t = 160$ as it was suggested in [15]. However, we did an experimentation on 3 different threshold values, where $t = \{140, 150, 160\}$ to three different manuscripts printed on the end of 19^{th}, early 20^{th} centuries, and in 1970s. The nature of background color and print condition are totally different in these books. We decided to use $t = 160$ as it proves to give the best result to these three manuscripts, while $t = 140$ or $t = 150$ presents very poor and poor results to one of the manuscripts mentioned before.

5.2 Segmentation Process

In building our OCR system, we also applied the method of segmentation process suggested in [15] which concentrates more on solving the problem of line segmentation. The line segmentation for Javanese characters becomes a challenging task as it has upper and below elements of characters that are separated by the white spaces as shown in Figure 2. Applying the available segmentation methods

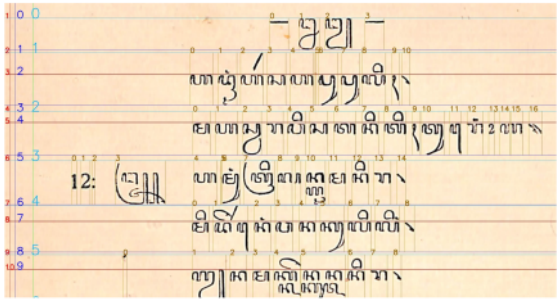

Figure 5: Three vertical lines on the left represent three passes on the line segmentation. Each line resulted from each pass is indexed and their difference on cutting the lines is marked by the index number.

will result in 2 or 3 separate lines which should be segmented into 1 line only. Therefore, the proposed method decomposes the line segmentation process into three stages or passes and one pass for the charater segmentation. Such segmentation strategy is aimed to adjust the nature of inputted text images having different height of line spaces. Figure 5 displays the line segmentation in three passes.

Briefly, the first pass of this three-stage line segmentation method is directed to find the line candidates by applying the Projection Profile Cutting (PPC) purely. In Figure 5, the PPC outputs are described with red lines and index numbers in red printed on the most left of the text. Having the line candidates, the task of the second pass is to remove falsely identified lines. This was done by removing any line candidades having the height value less than 5 pixels. The height threshold of 5 pixels came as a result of observation and experimentation applied on the segmentation of three manuscripts mentioned earlier. The ouput of the second pass is marked by the blue lines and index (see the middle vertical lines on the left on Figure 5).

The third pass is aimed to refine, split, or merge the falsely-identified line candidates by using the statistical technique to identify outliers of the vertical histogram of the text image. The information obtained from the statistical analysis was then used to regulate the thresholds and to decide whether to merge, split, or leave the lines as they were. The output of the third pass is displayed by the cyan horisontal lines, green vertical line (the third vertical line on the left of the text), and the cyan index numbers. Figure 5 shows that after a while, the third pass segmented the text into 6 lines shown by cyan index number 5, the second pass got 10 lines (index 9), and the pure PPC segmented the text into 11 lines (index 10).

After attaining lines of characters, the vertical segmentation was applied to obtain individual characters by analyzing the horizontal histogram of each line. This was done by applying Projection Profile Cutting technique. Figure 5 shows that mostly the characters are correctly segmented. The exception falls on those whose white space delimiter overlapped as shown by line index 5 and a set of characters which are indexed to number 5 also.

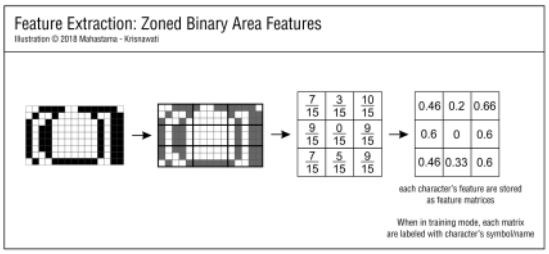

Figure 6: An illustration on how 9-area zoning works for extracting features of each character.

Table 1: An example of our feature extraction outputs in csv format.

f11	f12	f13	f21	f22	f33	f31	f32	f33	AS	class
.53	.25	.52	.57	.46	.37	.47	.35	.37	.31	na
.47	.58	.49	.13	.44	.41	0	.27	.39	.26	ra
.47	.07	.48	.51	.02	.48	.4	.33	.38	.26	pa
.54	.09	.23	.45	.17	.17	.61	.38	.33	.15	sa

5.3 Feature extraction Process

The feature of each character for both training and test data were extracted by applying a technique categorized as local feature extraction and a global one. The 9-area zoning was chosen to represent the local feature extraction methods. In this technique, the image divided into 9 equal window sizes. For each window, the ratio of the foreground pixels to the whole pixels of each window was calculated and then saved into a $3X3$ matrix. The process of 9-area zoning is illustrated on Figure 6. From this technique, we got 9 features for each character.

As for the global feature extraction method, we applied the aspect ratio technique (AS) which is obtained by dividing the height of a character by its width. This technique results in a single feature value for each character. Combined with the 9 features attained from 9-area zoning, there are in total 10 features for each character to train and to match in the classification process. The extracted features were then saved into a csv file format which made the classification process easier. Table 1 exemplifies the csv file format of our feature extraction outputs wherein the first nine features are named according to the raw and the column in the matrix, AS stands for *aspect ratio* and the classes stand for the character classes.

5.4 The classification Process

A classification is a process of assigning a class for a given data. In practice, the class is a label given to a group of data points sharing the same characteristics which are then computed into the data point's feature vectors. In general, the task of a classifier model is described by the Equation where y represents the output or the class of a given object x. Given a training set of labeled examples $(x_1, y_1), ...(x_n, y_n)$ and the object feature vector x, the prediction function f should be able to estimate to which class the object feature vector x belogs.

$$y = f(x) \tag{10}$$

In its implementation, the prediction function f takes form of a classifier. Each model of classifier attempts to predict the given object class using different approaches and techniques. We believe that there is no classifier inherently better than any other. For this reason, we built 4 different classifier models to experiment with our data. These classifiers are k-Nearest Neighbor, Gaussian Naive Bayes, Linear Discrimnant Analysis, and Support Vector Machine. The concept and algorithm of these classifiers have been explained in Section 4. We implemented the algorithms of these four classifiers using Scikit-Learn library in Python.

6 EXPERIMENT AND DISCUSSION

In order to do experiment and evaluate the prformance of these different classifier, we needed training as well as test data. Therefore, this section will present the strategy on how to get the data set, then it proceeds with the presentation of the experimental results and its evaluation.

6.1 Javanese Character Datasets

Since Latin alphabet was introduced by the colonial authority in 19^{th} century, the use of Javanese script in everyday life kept decreasing. Nowadays, it is no longer used but it can be found on the school lesson books and on the street signs in Yogyakarta and few cities in central Java. For this reason, our dataset was acquired by scanning 3 different books printed in different eras and which represent 3 different Javanese fonts. The first is the classical book *Serat Mangkunegaran IV* which was printed at the end of 19^{th} century, then the first 3 pages of book Matthew from the New Testament printed on early 20^{th} century and *Moelang Matja* which is a reading lesson book in Javanese script for elementary school and which was printed in 1970s. We selected randomly 72 pages out of these three books with one criteria only that these pages were free from scanning noises. However, they still contain light noises such as small blots or salt and pepper noise. Each page underwent the preprocessing and segmentation whose methods have been elucidated in the previous subsections. In average, one page comprises 250 character segments, the short page may consist ca. 180 segments and the longest one has ca. 320 segments. In total, we obtained 19.112 segments of characters excluding the undersegmentation or oversegmentation cases.

The data annotation was performed through two different methods: a crowdsourcing, annotation and evaluation workshops. Since not many Javanese could read and write Javanese script any more, the crowdsourcing was announced on the social media account of Wikimedia Indonesia (WMID). Thus, the annotator candidates are those who have interest on preserving Javanese script and language. The annotator selection process was conducted by administering 5 Javanese phrases with five different levels of difficulty. The task is to transliterate them in Latin alphabet and only those whose score is greater than or equal to 80 were accepted as annotators. From this process, we got 94 annotators.

Besides, we conducted annotation and evaluation workshop for the Javanese culture partisans living nearby Yogyakarta. The goal

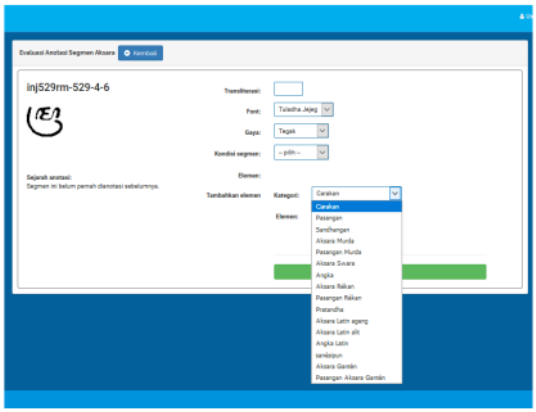

Figure 7: One of pages in annotation application Cakra developed for the prupose of dataset annotation and vaerification

was to to control the quality of annotation resulted from the crowdsourcing process. The evaluators were either volunteers or personnel of Javanese Wikipedia Community and could be considered as Javanese experts since most of them have degrees on Javanese literature or pedagogy. Through these techniques, we obtained 15.414 annotated characters and 633 classes whose correctness has been verified. The annotation and verification processes were enabled through a web-based annotation application *Cakra* that we developed for this specific purpose. One page of *Cakra* application is displayed in Figure 7.

6.2 Exprimental Results

In experimenting the four classifiers mentioned previously, we used 162 classes out of 633 available one. The reason is that more than half of these classes have less than or equals to 3 instances of annotated data. We considered that this is too small and decided to have minimally 10 instances of character samples in each class. The class in rank 162 is the cut-off for having minimally 10 annotated samples. These classes represent the frequently used characters which mostly are dominated by punctuations, *Carakan* and the most common compound characters. Naturally, these 162 classes excludes the archaic characters found only on the literary works such as *Serat Mangkunegaran*, and which in modern Javanese lesson books are no longer found.

For the experiments, we applied two phases of evaluation to our classifier models. The first is the model validation which evaluates our classifier models during the learning process. Based on the number of data, we opted 5-fold cross validation with a goal to get the most reliable model and which is able to accurately predict the classes in real use case. The second is the testing phase which assesses the real performance of the four models by the use of confusion matrix.

In validating our classifier models, we made use the class *KFold* provided by sklearn library in Python. We set $k = 5$ and splitted the dataset into a ratio of $80 : 20$, which did 5-fold cross

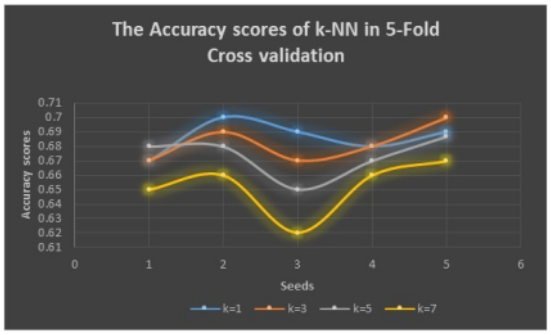

Figure 8: The accuracy scores of k-NN model in 5 seeds with $k = 1, 3, 5, 7$

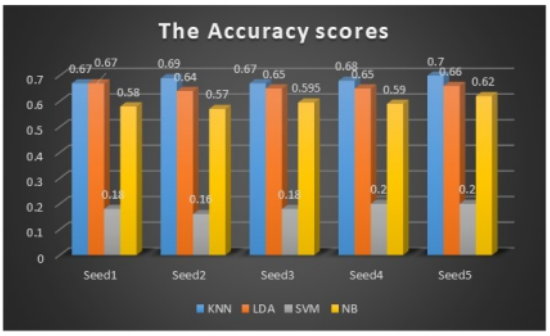

Figure 9: The accuracy scores of four experimented classifier models

validation by training each model 5 times on 80% of dataset and testing the rest, namely the 20%. The test data in each fold was guaranteed to be different. At first, we did some trial and error in determining the training-test data ratio and came to the best result on 80/20. In this case, we validated the models by computing their accuracy scores on each fold.

For the k-NN model, we tuned the leaf size parameter of the KDtree to 162 which corresponds to the number of classes. We decided to give the higher weight for closer neighbours which was realized by tuning the parameter weight to the *'distance'* value. The distance metric was set to 2 which refers to the Euclidean distance instead of using Manhattan distance metric. We experimented k-NN with $k = \{1, 3, 5, 7\}$. The accuracy scores of k-NN model is presented in Figure 8. Seed, in this chart, signifies the 20% data split to be the test data.

This experiment results showed that the $k = 1$ gained the highest average of the accuracy score by 0.69. However, the score difference is insignificant as the average of accuracy score remains on 0.6. It can be seen also that the k-parameter and the accuracy score are inversely proportional because the increase on k increases the possibillity of having bias and thus decreases the prediction accuracy. The chart in Figure 8 shows also that the accuracy scores of seed 5 are relatively higher than the other.

The accuracy scores of 4 experimented classifier models could be found in Figure 9. Tested on the limited number of data sample, k-NN proves to predict a bit more accurately than the other models. The most noticeable is the accuracy scores of SVM model. With the accuracy average of 0.18, SVM performance is quite far behind the other models. Our experiment found the contrast result to the SVM performance reported in some researches such as [6, 22], which stated that SVM showed great performance. In case that we applied a false kernel, we modified the kernel from using linear kernel into RBF kernel. However, the accuracy average of SVM using RBF kernel is much lower which achieves 0.02. To answer our curiosity to SVM's performance, we tested it with the Iris dataset provided on Github. The accuracy score of our SVM classifier tested on Iris dataset achieved 0.9 which is more than satisfying. We assumed that the underlying cause of poor performance of SVM is the insufficient number of training data. In this case, the number of features for each character sample is greater than the number of training data in each class. The ratio of the number of features to the number of training data in each class is proportional to 80% : 100%.

In implementing Linear Discriminant Analysis (LDA) model, we applied Singular Value Decomposition (SVD) for the solver parameter. The advantage of using SVD is that it does not compute the covariance matrix so that it saves the computation time. The prediction accuracy of LDA is competitive to k-NN in any seed (cf. Fig. 9) as the difference is less than 0.05. The Gaussian Naive Bayes (GNB) is on the third rank with the accuracy average on 0.59.

In testing phase, we evaluated the performance of the experimented models with the precision and sensitivity metrices. Unlike accuracy, the precision metric evaluates the prediction results on the basis of those predicted positive and hence precision is viewed as a finer metrics than accuracy in clasiffication case. In addition to precision, the proportion of actual positive cases that got predicted as positive (or true positive) was measured also using sensitivity metric. This metric is also well know as recall in the Information Retrieval concept. We applied micro average precision and sensitivity and compared them with the weighted ones. The Micro-Average Precision and Sensitivity are computed using the Equation 11 and Equation 12.

$$prec_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FP_i)} \tag{11}$$

$$sensi_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FN_i)} \tag{12}$$

where TP refers to True Positive prediction, FN stands for False Negative, and FP is for False Positive prediction.

The test data used in this phase were the same one used in the validation phase, which is approximately 20% of the dataset. Based on the confusion matrix, the precision and sensitivity of each tested data were able to compute. As presenting the whole matrix of evaluation results is impossible here, Table 2 displays a fraction of precision and sensitvity scores of character class from the evaluation outputs of LDA classifier. The character classes displayed in Table 2 were randomly chosen to represent different scores of precision and sensitivity.

Table 2: A fraction of precision and sensitivity scores on some character classes from evaluating the LDA classifier. The classes of characters are represented with the latin alphabets.

Classes	Precision	Sensitivity	Classes	Precision	Sensitivity
bi	1.00	0.50	dha	0.00	0.00
jar	1.00	1.00	mar	1.00	0.67
lê	0.00	0.00	ra	0.80	1.00
kir	0.67	0.50	kka	1.00	1.00
ma	0.00	0.00	mba	1.00	0.67
mê	0.33	0.50	ntu	0.67	1.00
pra	0.50	1.00	si	0.67	0.67
tri	1.00	1.00	wi	0.25	1.00

Table 3: The scores of micro-average precision, micro-average Sensitivity, and weighted-average precision and sensitivity of classifier models

	Micro-average		Weighted-averaged	
	Precision	Sensitivity	Precision	Sensitivity
KNN	0.69	0.69	0.74	0.69
LDA	0.66	0.66	0.71	0.66
SVM	0.18	0.18	0.18	0.18
GNB	0.62	0.62	0.67	0.62

The average Precision and Sensitivity were also weighted using the support values which resulted in the weighted average Precision and Sensitivity. Table 3 presents the measures on micro-average precision and sensitivity along with their weighted ones. This evaluation showed that the micro-average precision (MAP) scores of each model equal to their scores of micro-average sensitivity (MAS). Interestingly, the weighting technique has no influence at all on the weighted-average sensitivity (WAS) scores. However, it proves to be able to increase each model's precision score on the weighted-average precision (WAP). The exception falls to the SVM model whose MAP, MAS, WAS and WAP scores remain constant on 0.18. Experimented on our data, the SVM performance remains a mystery as we did not find a way to increase either its accuracy, MAP, or WAP. However, we assumed that the number of our annotated dataset is too small for training the SVM model.

As we scrutinized the confusion matrix, we found out that there are 6 characters in total which are unrecognizable by all classifier models and in every fold of 5-fold cross validation technique. Their precision rates are completely zero. Transliterated in Latin alphabets, this characters comprises *dha, ku, l, la, ma,* and the number *1* in Javanese. The failure to recognize number 1 in Javanese character is reasonable since its form and stroke are almost similar to the character *ga*, while there is no reasonable explanation why the other characters are always unrecognizable. There are also some characters whose precision and sensitivity scores are very low and their scores are no greater than 0.20 or 0 in most experiment folds. These characters are *2, lu, mu* and *wê*. We assumed that this low recognition rate for the number 2 in Javanese is that it has stroke which is almost similar to character *lê*.

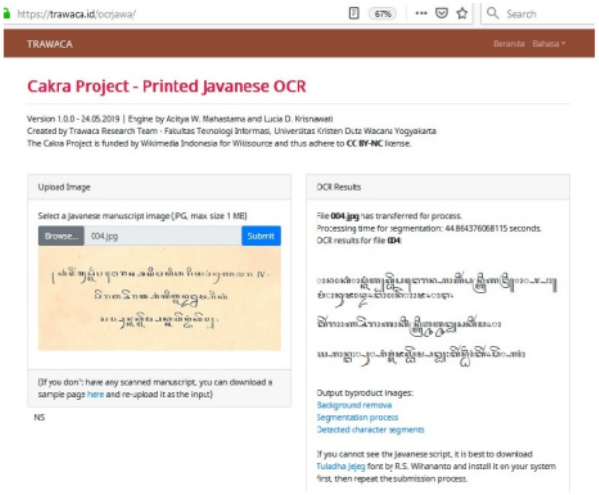

Figure 10: The capture of the on-off OCR system for Javanese characters

Though the accuracy, precision, and sensitivity scores of the experimented classifier models are still far from satisfying, the OCR system for Javanese character has been deployed and uploaded to the Trawaca website. It is accessible on the following link https://trawaca.id/ocrjawa. Processing the data online, it needs ca. 2-5 minutes to process the whole page of a Javanese text image containing ca. 250-300 characters. A sample page is provided also in case the reader has no scanned Javanese text. The byproducts of the processing stages such as background removal, segmentation and detected character segments could be seen by clicking the provided links. The capture of the on-off OCR system could be seen in Figure 10.

7 CONCLUSION

In this paper, we have presented the building process of four classifier models in a pipeline of an on-off printed OCR system for recognizing Javanese characters. These classifier models have been validated using 5-fold cross validation and assessed using the accuracy, precision and sensitivity metrices. Trained with characters taken from 3 books from different eras, the k-NN-based classifier outperforms any other classifier models. Its weighted-averaged precision score reaches 0.71 and its weighted-averaged sensitivity reaches 0.69.

Interestingly, the performance of SVM is very disappointing as its accuracy, precision, and sensitivity scores are averagely less than 0.2. In general, we assumed that the insufficient number of training data may cause the poor performance of SVM and prevents the other classifier models to attain the maximal performance on making a class prediction. This is proved by validating and testing our four classifiers to Iris dataset which resulted in weighted-averaged precision higher than 0.9. However, this research has contributed on the finding the total number of distinct Javanese characters which achieves 633 classes. Further, if it is compared to the researches on

the same field and objects (cf. the last 2 paragraphs of of section 2), this project has moved a step forward by using 162 character classes and deploying the system for real use instead of building a prototype trained with the basic Javanese characters only (*Nglegena*).

For future work, we need to improve the method of dataset annotation so that most classes have a well balanced number of character samples. The disadvantage of the previous method is that it took all characters from 72 pages to annotate and thus created a gap on the character frequency or the number of samples. The most commonly used characters have more annotated data, while more than half of the classes have less than or equals to 3 characters samples only. We projected that on the future annotation scheme, we would like to make a selection on the basis of lines containing compound and rare characters. For the classifier models, we plan to tune up more parameters and use some additional features to improve their performance, especially the Support Vector Machine.

8 ACKNOWLEDGMENTS

We would like to give our deepest gratitude to Wikimedia Indonesia (WMID) which has funded this research and partially funded the conference participation. We also give our thanks to Javanese Wikipedia community for their contribution on evaluating the character segments. We would also like to show our gratitude to Informatics Department, Duta Wacana Christian University which partially funded our participation in this conference. We thank to the head of Javanese Wikipedia Indonesia for his assistance in coordinating the workshops for evaluating our datasets. Last but least, we thank to everyone who was involved in the data annotation through the crowdsourcing process.

REFERENCES

[1] Bea Alex, Claire Grover, Ewan Klein, and Richard Tobin. 2012. Digitised Historical Text: Does it have to be mediOCRe?. In *Proceedings of KONVENS 2012 (LThist 2012 workshop)*. Vie 401–409.

[2] Muhammad R. Asif, Qi Chun, Sajid Hussain, Muhammed S. Fareeq, and Subhan Khan. 2017. Multinational vehicle license plate detection in complex backgrounds. *al of Visual Communication and Image Representation* 6, C (2017), 176–186. ://doi.org/10.1016/j.jvcir.2017.03.020

[3] Youssef Bassil and Mohammad Alwani. 2012. OCR Post-processing Error Correction Algorithm Using Google's Spelling Suggestion. *Journal of Emerging Trends mputing and Information S* 3, 1 (Jan. 2012). http://arxiv.org/abs/1204.0191

[4] Wojciech Bieniecki, Szymon Grabowski, and Wojciech Rozenberg. 2007. Image Preprocessing for Improving OCR Accuracy. In *MEMSTEC'07*. Lviv-Polyana, 0.

[5] Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu, and Ching Y. Suen. 2007. *Character Recognition Systems: A Guide for Students and Practiotioners* (1st. ed.). Wiley, New y.

[6] Chandra K. Dewa, Amanda L. Fadhilah, and Afiahayati. 2018. Convolutional Neural Networks for Handwritten Javanese Character Recogniti *ndonesian Journal of Computing and Cybernetics Systems* 12, 1 (Jan 2018), 83–94. https: //doi.org/10.22146/ijccs 4

[7] Maruf A. Dhali, Seng He, Mladen Popović, Eibert Tigschelaar, and Lambert Schomaker. 2017. A Digital Palaeographic Approach towards Writer Identification in the Dead Sea Scrolls. In *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, Vol. 1. Porto, Portugal, 693–702. https://doi.org/10.5220/0006249706930 8

[8] Rudy Adipranata Gregorius S. Budhi. 2015. Handwritten Javanese Characters Recognition Using Several Artificial Neural Network Methods. *Journal of ICT Research Applications* 8, 3 (2015), 195–212.

[9] Karez A. Hamad and Mehmet Kaya. 2016. A Detailed Analysis of Optical Character Recognition Technology. *International Journal of Applied Mathematics, ctronics and Computers* 4 (2016), 244–249.

[10] Mahdi Hamdani, Haikal El Abed, Monji Kherallah, and Adel M. Alimi. 2009. Combining Multiple HMMs Using On-line and Off-line Features for Off-line Arabic Handwriting Recognition. In *10th International Conference on Document Analysis and Recognition*. IEEE Computer Society, 201–205.

[11] Brian Karundeng, Kho I. Eng, and Anto S. Nugroho. 2009. An Evaluation of Feature Extraction Aalgorithm for Automatic Language Transcri n System for Acient Handwriting Javanese Manuscripts. In *Proceeding of d^{rd} International Seminar on Industrial Engineering and Manage* . Bali, G59–G65.

[12] Lucia D. Krisnawati and Aditya W. Mahastama. 2018. A Javanese Syllabifier Based on its Orthographic Forms. In *2018 International rence on Asian Language Processing, IALP*. Bandung, Indonesia, 244–249. https://doi.org/10.1109/IALP. 8629173

[13] Changqing Liu. 2012. On Tangut Historical Documents Recognition. In *International Conference on Medical Physics and Biomedical Engineering (Pysics Procedia)*, Vol. 3. Elsevier, 1212–1216.

[14] Jean-Marc Ogier Made W. A. Kesiman, Jean-Christophe Burie. 2016. A New Scheme for Text Line and Character Segmentation from Gray Scale Images of Palm Leaf Manuscript. In *15th InternationalConference on Frontiers in Handwriting Recogn 2016*. Shenzhen,China, 32–330.

[15] Aditya W. Mahastama and Lucia D. Krisnawati. 2019. Improving Projection Profile for Segmenting Characters from Javanese Manuscripts. In *International rence on Intermedia Arts and Creative Technology (CreativeArts)*. Yogyakarta.

[16] R. Manmatha and Toni M. Rath. 2003. *Indexing of Handwritten Historical Documents - Recent Progress*. Retrieved May 18, 2019 from https://pdfs.semanticscholar. org/47a3/de4595eb1d486b9283e415356 22b462.pdf

[17] Marcin Namysl and Iuliu Konya. 2019. Efficient, Lexicon-Free OCR using Deep ning. *CoRR* abs/1906.01969 (2019). http://arxiv.org/abs/1906.01969

[18] Nazaruddin and Sayed Muchallil. 2017. Comparison Online to Offline Handwritten Jawi Character Recognition Application. *Indian Journal of Science and Tech y* 10, 12 (2017), 1–5. www.indjst.org

[19] Toni M. Rath and R. Manmatha. 2003. Features for Word Spotting in Historical Manuscripts. In *Proceedings of the Seventh International Conference on Document Analysis and R nition (ICDAR '03)*. IEEE Computer Society, 218–222.

[20] Christian Reul, Marco Dittrich, and Martin Gruner. 2017. Case Study of a highly automated Layout Analysis and OCR of an incunabulum: 'Der Heiligen Leben' (1488). In *Proceedings of the 2nd International rence on Digital Access to Textual Cultural Heritage (DATeCH2017)*. ACM, G ttingen, Germany, 155–160. https://doi.org/10.1145/3078081.3078098

[21] Shai Shalev-Shwartz and Shai Ben-David. 2014. *On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes*. Cambridge ersity Press, New York.

[22] Olarik Surinta, Mahir F. Karaaba, Lambert R.B. Schomaker, and Marco A. Wiering. 2015. Recognition of Handwritten Characters Using Local Gradient Feature Descriptors. *En ring ApplicationsofArtificial Intelligence* 45 (2015), 405–414.

[23] Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahimd, and Aboul E. Hassanien. 2017. Linear Discriminant Analysis: A Detailed Tutorial. *AI Communications* 30, 2 (2017), 169–190. https://doi.org/10 /AIC-170729

[24] Anastasia R. Widiarti and Phalita N Wastu. 2009. Javanese Character Recognition Using Hidden Markov Model. *International Journal of Computer, Electrical, Automation, Control and Information Engineering* 3, 9 (2009), 2201–2204.

Building Classifier Models

ORIGINALITY REPORT

18%	14%	14%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	wiese.free.fr Internet Source	2%
2	www.ijcttjournal.org Internet Source	1%
3	www.coursehero.com Internet Source	1%
4	Lucia D. Krisnawati, Aditya W. Mahastama. "A Javanese Syllabifier Based on its Orthographic System", 2018 International Conference on Asian Language Processing (IALP), 2018 Publication	<1%
5	Fetty Tri Anggraeny, Yisti Vita Via, Retno Mumpuni. "Image preprocessing analysis in handwritten Javanese character recognition", Bulletin of Electrical Engineering and Informatics, 2023 Publication	<1%
6	Baligh M. Al-Helali, Sabri A. Mahmoud. "Arabic Online Handwriting Recognition (AOHR)", ACM Computing Surveys, 2017 Publication	<1%

7	Submitted to Panjab University Student Paper	<1 %
8	creativearts.isi.ac.id Internet Source	<1 %
9	1filedownload.com Internet Source	<1 %
10	metatron.scholasticahq.com Internet Source	<1 %
11	Raji Ghawi, Jürgen Pfeffer. "Movie Genres Classification using Collaborative Filtering", Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services, 2019 Publication	<1 %
12	Hongxi Wei, Hui Zhang, Guanglai Gao. "Word Image Representation Based on Visual Embeddings and Spatial Constraints for Keyword Spotting on Historical Documents", 2018 24th International Conference on Pattern Recognition (ICPR), 2018 Publication	<1 %
13	Submitted to Universidad de Valladolid Student Paper	<1 %
14	dokumen.pub Internet Source	<1 %
	www.pure.ed.ac.uk	

15	Internet Source	<1%
16	Dona Valy, Michel Verleysen, Sophea Chhun, Jean-Christophe Burie. "A New Khmer Palm Leaf Manuscript Dataset for Document Analysis and Recognition", Proceedings of the 4th International Workshop on Historical Document Imaging and Processing - HIP2017, 2017 Publication	<1%
17	www.yumpu.com Internet Source	<1%
18	www.mitpressjournals.org Internet Source	<1%
19	Elizabeth Nurmiyati Tamatjita, Rouly Doharma Sihite, Aditya Wikan Mahastama. "A Lightweight Chinese Character Recognition Model for Elementary Level Hanzi Learning Application", 2019 International Congress on Applied Information Technology (AIT), 2019 Publication	<1%
20	Submitted to Higher Education Commission Pakistan Student Paper	<1%
21	Submitted to Asia Pacific Instutute of Information Technology Student Paper	<1%

22	publikationen.bibliothek.kit.edu Internet Source	<1 %
23	www.ijareeie.com Internet Source	<1 %
24	G. Abdul Robby, Antonia Tandra, Imelda Susanto, Jeklin Harefa, Andry Chowanda. "Implementation of Optical Character Recognition using Tesseract with the Javanese Script Target in Android Application", Procedia Computer Science, 2019 Publication	<1 %
25	Olarik Surinta, Mahir F. Karaaba, Lambert R.B. Schomaker, Marco A. Wiering. "Recognition of handwritten characters using local gradient feature descriptors", Engineering Applications of Artificial Intelligence, 2015 Publication	<1 %
26	repository.tudelft.nl Internet Source	<1 %
27	Mahdi Hamdani, Haikal El Abed, Monji Kherallah, Adel M. Alimi. "Combining Multiple HMMs Using On-line and Off-line Features for Off-line Arabic Handwriting Recognition", 2009 10th International Conference on Document Analysis and Recognition, 2009 Publication	<1 %
	moam.info	

28	Internet Source	<1%
29	Submitted to Somaiya Vidyavihar Student Paper	<1%
30	etd.ohiolink.edu Internet Source	<1%
31	kth.diva-portal.org Internet Source	<1%
32	Submitted to University of Edinburgh Student Paper	<1%
33	Submitted to University of Melbourne Student Paper	<1%
34	Submitted to University of Sussex Student Paper	<1%
35	www.mdpi.com Internet Source	<1%
36	Submitted to Mepco Schlenk Engineering college Student Paper	<1%
37	Stephen Marsland. "Machine Learning - An Algorithmic Perspective", Chapman and Hall/CRC, 2019 Publication	<1%
38	Submitted to University of Queensland Student Paper	<1%

39	jurnal.ugm.ac.id Internet Source	<1 %
40	Submitted to University of Petroleum and Energy Studies Student Paper	<1 %
41	dzone.com Internet Source	<1 %
42	Marcin Namysl, Iuliu Konya. "Efficient, Lexicon-Free OCR using Deep Learning", 2019 International Conference on Document Analysis and Recognition (ICDAR), 2019 Publication	<1 %
43	Mikhail Kanevski. "Machine Learning for Spatial Environmental Data - Theory, Applications, and Software", EPFL Press, 2019 Publication	<1 %
44	Submitted to University of Warwick Student Paper	<1 %
45	ipfs.io Internet Source	<1 %
46	web.cse.ohio-state.edu Internet Source	<1 %
47	www.researchgate.net Internet Source	<1 %
48	Guide to OCR for Arabic Scripts, 2012. Publication	<1 %

49	export.arxiv.org Internet Source	<1%
50	Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim, Aboul Ella Hassanien. "Linear discriminant analysis: A detailed tutorial", AI Communications, 2017 Publication	<1%
51	Submitted to University College London Student Paper	<1%
52	d.docksci.com Internet Source	<1%
53	pdfs.semanticscholar.org Internet Source	<1%
54	www.fmph.uniba.sk Internet Source	<1%
55	Jianshuang Xu, Johannes Klein, Jörn Jochims, Niklas Weissner, Rüdiger Kays. "A reliable and unobtrusive approach to display area detection for imperceptible display camera communication", Journal of Visual Communication and Image Representation, 2022 Publication	<1%
56	www.slideshare.net Internet Source	<1%

57	Submitted to Mansoura University Student Paper	<1 %
58	Submitted to The American College of Greece Libraries Student Paper	<1 %
59	etasr.com Internet Source	<1 %
60	Scott N. Romaniuk, Mary Manjikian. "Routledge Companion to Global Cyber-Security Strategy", Routledge, 2021 Publication	<1 %
61	meu.edu.jo Internet Source	<1 %
62	preview.hindawi.com Internet Source	<1 %
63	repositori.unsil.ac.id Internet Source	<1 %
64	usir.salford.ac.uk Internet Source	<1 %
65	Keerthi Varadhi, Chinta Someswara Rao, GNVG Sirisha, Butchi Raju katari. "Recognizing human activities using light-weight and effective machine learning methodologies", F1000Research, 2024 Publication	<1 %

66	Matthew Turk, Alex Pentland. "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, 1991 Publication	<1 %
67	T. Coen. "Optimizing the tuning parameters of least squares support vector machines regression for NIR spectra", Journal of Chemometrics, 05/2006 Publication	<1 %
68	arxiv.org Internet Source	<1 %
69	docslib.org Internet Source	<1 %
70	ebin.pub Internet Source	<1 %
71	kobra.uni-kassel.de Internet Source	<1 %
72	link.springer.com Internet Source	<1 %
73	repository.petra.ac.id Internet Source	<1 %
74	sites.labic.icmc.usp.br Internet Source	<1 %
75	A. Staiano, R. Tagliaferri, G. Longon, P. Benvenuti. "Committee of spherical	<1 %

probabilistic principal surfaces", 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), 2004
Publication

---

76 Gulshan Shrivastava, Sheng-Lung Peng, Himani Bansal, Kavita Sharma, Meenakshi Sharma. "New Age Analytics - Transforming the Internet through Machine Learning, IoT, and Trust Modeling", Apple Academic Press, 2020 <1%
Publication

---

77 Lecture Notes in Computer Science, 2002. <1%
Publication

---

78 Regina Cernicka, Andreas Mladenow, Christine Strauss. "The Impact of Updates in Social Crowd Projects", Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services, 2019 <1%
Publication

Exclude quotes	Off	Exclude matches	< 3 words
Exclude bibliography	Off		