Machine Learning Methods for Optical Character Recognition

Ivan Dervisevic

December 18, 2006

Abstract

Success of optical character recognition depends on a number of factors, two of which are feature extraction and classification algorithms. In this paper we look at the results of the application of a set of classifiers to datasets obtained through various basic feature extraction methods.

Contents

Chapter 1

Introduction

Optical character recognition (OCR), an area of computer science that started developing as early as 1950, currently encompasses two previously distinct areas – pure optical character recognition, using optical techniques such as mirrors and lenses and digital character recognition, using scanners and computer algorithms. The field has come far, from applications that required font-specific training, to current, intelligent applications that can recognize most fonts with a high degree of accuracy. Most of these intelligent applications use some form of machine learning to adapt and learn to recognize new fonts. Current OCR software can achieve an accuracy of more than 97% in cases of typewritten text [2], 80-90% for clearly handwritten text on clear paper, while recognition of cursive text is an active area of research.

The intent of this paper is to focus on recognition of single typewritten characters, by viewing it as a data classification problem. Analysis of the results should give us an idea on the viability of these algorithms for use in a complete OCR application. Due to long computing times demanded by large datasets, we will work only with a small sample of capital letters from a dataset consisting of images of single characters from over 1000 True Type fonts. The results of this smaller-scale research should show what steps need to be taken to speed up the processing of a larger dataset and the preprocessing required to get the best results.

After evaluating the results of the application of classification algorithms on plain, unaltered data, we will take a look at some feature extraction functions and the results of the classification of images altered by the application of these functions on the original images used.

The final step is a detailed examination of results achieved by each of these algorithms, which will allow us to find problem areas for each and point in the direction of further research.

A brief overview of the material presented in this paper follows:

- Chapter two will focus on discussing basic Data Mining and Machine Learning ideas and describe the classification methods in more detail.
- Chapter three gives a brief introduction to the entire process of OCR and the steps that are neccessary for a successfull OCR application.
- Chapter four is a presentation and analysis of the results.

- Chapter five offers a view on the possibilities for future improvements on this research.

Chapter 2

Machine Learning and Data Mining

2.1 Introduction to Basic Data Mining and Machine Learning Concepts

As a broad sub-field of artificial intelligence, machine learning is concerned with the development of algorithms and techniques that allow computers to "learn". At a general level, there are two types of learning: inductive, and deductive. Inductive machine learning methods create computer programs by extracting rules and patterns out of massive data sets.

Data mining is defined as the process of discovering patterns in data. The process must be automatic or (more usually) semiautomatic. The patterns discovered must be meaningful in that they lead to some advantage. The data is invariably present in substantial quantities. Useful patterns allow us to make nontrivial predictions on new data. These useful patterns are called structural patterns and the techniques for finding and describing these structural patterns in data constitute a machine learning problem.

Four basically different styles of learning appear in data mining applications. In *classification,* the learning scheme is presented with a set of classified examples from which it is expected to learn a way of classifying unseen examples. In *association learning*, any association among features is sought, not just ones that predict a particular class value. In *clustering*, groups of examples that belong together are sought. In *numeric prediction*, the outcome to be predicted is not a discrete class but a numeric quantity.

We have chosen to view OCR as a classification problem, but will be using some clustering techniques to help improve our results. Classification is sometimes called supervised learning because, in a sense, the method operates under supervision by being provided with the actual outcome for each of the training examples. This outcome is called the class of the example. The success of classification can be judged by trying out the concept description that is learned on an independent set of test data for which the true classifications are known but not made available to the machine during the learning process. The success rate on test data gives an objective measure of how well the concept has been

learned.

The input to a machine learning scheme is a set of instances. These instances are the things that are to be classified, associated, or clustered. Each instance is an individual, independent example of the concept to be learned. In addition, each one is characterized by the values of a set of predetermined attributes. Each dataset is represented as a matrix of instances versus attributes, which in database terms is a single relation, or a flat file.

2.2 Classifiers Used

Four classifiers that were used in this research will be discussed as examples: Naïve Bayes [1], SMO – a classifier that implements John Platt's sequential minimal optimization for training a support vector classifier [1] and a classifier generating a pruned C4.5 decision tree [3].

NaïveBayes classifier is based on Bayes' rule of conditional probability and "naïvely" assumes independence. The assumption that attributes are independent (given the class) may sound simplistic but, as [4] has shown, the number of distributions for which the additional penalty term is large goes down exponentially fast (that is, almost all distributions are very close to the distribution assuming conditional Independence).

ComplementNaïveBayes is a variation of the multi-nominal naïve Bayes classifier, optimized to work with text.

SMO trains a support vector classifier which uses linear models to implement nonlinear class boundaries by transforming the input using a nonlinear mapping (transforming the instance space into a new space).

C4.5 creates a decision tree using a divide-and-conquer algorithm. The tree is pruned to remove unnecessary subtrees, make the searches faster and avoid overfitting. The complexity of training a decision tree of this type is $O(mn \log n) + O(n (\log n)^2)$.

For more information, see [1].

2.3 Classifier Evaluation

To evaluate the performance of a classifier, some standards need to be set. Each classifier needs to be trained on one dataset and tested on another (called, respectively, the training set and the test set). Because of the limited sizes of datasets in research, one of the most commonly used methods "n runs of k-fold crossvalidation", which divides the dataset into k random pieces of the same size, then takes each piece to be the training set and tests on the remaining $k-1$ pieces. This process is repeated $n$times to obtain an average result.

A number of measures are used to evaluate the success of a classifier [1]:

Precision is he proportion of correctly classified instances to all the instances classifier:

$precision = \frac{|R \cap D|}{|D|}$

where R is the set of correctly classified instances, while D is the complete set of classified instances.

Recall is the proportion of correctly classified instances, out of all classified instances:

$recall = \frac{|R \cap D|}{|R|}$

where R and D are again the set of correctly classified instances and the complete of instances classified.

The F-measure is the weighted harmonic mean of precision and recall, the traditional F-measure or balanced F-score is:

$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

This is also known as the F_1 measure, because recall and precision are evenly weighted.

The general formula for non-negative real α is:

$F = (1 + \alpha) \cdot \frac{precision \cdot recall}{\alpha \cdot precision + recall}$

Two other commonly used F measures are the F_2 measure, which weights recall twice as much as precision, and the $F_{0.5}$ measure, which weights precision twice as much as recall.

Another measure that can be observed is the measure of correlation between results of the classification and the dataset, called the $\kappa - statistic$. A $\kappa - statstic$ value over 0.7 shows a statistically significant correlation.

Finally, we can view the results of the classification as a confusion matrix – which shows us all possible classes for the classification and how the instances of each class were classified.

Chapter 3

Optical Character Recognition

As mentioned in the introduction, OCR translates images of handwritten or typewritten text (usually captured by a scanner) into machine-editable text, or images of characters into a standard encoding scheme representing them (e.g. ASCII or Unicode). Initially the goal was to recognize machine printed text formed from characters drawn from a single font and of a single size. With success in recognizing single-font, single-size text the goal was expanded to reading multi-font text in which a range of character sizes may be present. Success bred more challenging problems such as hand-printed character recognition, and, of late, the recognition of cursive scripts, both printed and handwritten.

Early techniques exploited the regularity of the spatial patterns. Techniques like template matching used the shape of single-font characters to locate them in textual images. More recent techniques that are used to recognise handwritten text do not rely solely on the spatial patterns but instead characterise the structure of characters based on the strokes used to generate them. These newer techniques are now also applied to the recognition of machine printed characters, improving recognition rates and increasing the robustness of the techniques against image defects such as text alignment, imaging noise, and contrast deficiencies.

The recognition rates obtained on scanned images of machine printed text are excellent and sufficiently high (in excess of 99%) to transform OCR into a viable commercial process. The recognition of constrained handwritten material, such as addresses on postal items is also a commercial reality that is used by most of the major postal services. However, the recognition of cursive handwritten material is not yet developed sufficiently for it to be a useful technique for reading general handwritten textual material [2].

The major OCR applications are able to take advantage of the fact that the text is presented on a uniform background that has sufficiently high contrast between text and background. When the background is not uniform OCR recognition rates are substantially lower, and at present, are not commercially viable.

Although modern OCR algorithms exhibit considerable variety in the techniques employed, the major stages in the OCR process can still be elaborated.

No one algorithm uses all the techniques, nor all the stages, and different algorithms use the stages in different orders. However, all algorithms do have to confront the issues raised in each of the stages. Following is a description of those stages. [9]

Step 1: Extraction of Character Regions from an Image

Extraction of character regions from an image is based on using ancillary information known about the image to select an image property (or properties) that is (are) sufficiently different for the text regions and the background regions to be a basis of separating one from the other.

If it is known that the text is printed on a white or light background, we can use the detection of white space as a means of segmenting the image into regions that contain lines of text. We can reapply the process to segment the line regions into smaller regions that contain words. A Horizontal Projection Operator generates a vertical histogram of the image and white space has a distinctive signature (e.g. zero if white is represented by a grey level of 0 and black by a grey level of 255 for an 8-bit image). We use this white space's signature to segment the histogram and hence the lines of text. We now repeat the operation for each line of text using the Vertical Projection Operator to segment the words in each line of text.

Alternatively, we might know the expected character size of the text, or other spatial arrangements of the text, and use Connect Component Analysis or Mathematical Morphology to identify areas of the image that have character-like spatial arrangements. We can select a mask so that morphology operations will grow the characters to join them together into word regions without bridging the space between words or between lines. Another option is to use connected component methods to identify image regions that contain blobs of the right size.

Further options that are available are based on the differing spatial frequencies associated with characters as opposed to a uniform-like background. Local Fourier masks or texture masks can be used to identify areas of high spatial frequencies and hence segment textual regions[1].

Step 2: Segmentation of the Image into Text and Background

While some OCR algorithms work with grey-scale images, many convert their input into binary images during the early stages of processing. Although this is the case most try to extract non-textual regions, such as graphics before they segment text from the background. Given that we have image regions that contain text, whether single word regions or whole slabs of text, the goal of this stage is to identify image pixels that belong to text and those that belong to the background.

The most common technique used is thresholding of the grey-level image. The threshold value may be chosen using ancillary knowledge about the image,

[1] It should be noted that a large number of algorithms reverse stages one and two preferring to generate a binary image in which text and background are separated before extracting textual regions. None-the-less, the textual regions can be extracted from the grey-level image before the image is converted into a binary image.

using statistical techniques to select a global threshold that 'best' divides the image into two classes, or by using measures calculated in the neighbourhood of each pixel to determine the 'best' local threshold. In practice local adaptive thresholds seem to produce the best segmentation results.

Step 3: Conditioning of the Image

Whatever techniques are employed to extract text from its background it is inevitable that the resultant image segments will contain some pixels identified as belonging to the wrong group. Conditioning the image refers to the techniques used to 'clean up' the image, to delete noise.

Morphological operators and neighbourhood operations are the most popular for identifying noise and deleting it. Generally isolated pixels are easily removed while regions of noise adjacent to text or background are more difficult to identify and hence remove.

Step 4: Segmentation of Characters

Historically, OCR algorithms segmented their input image into regions that contained individual characters. While this is often possible for machine-printed text, it is more difficult for handwritten material and is perhaps the major task encountered in reading cursive scripts. Many newer algorithms, even for machine-printed text, avoid this stage and move into character recognition without prior character segmentation. In fact character recognition occurs before character segmentation.

Morphology operations, Connected Component Analysis, and vertical projections are used to segment characters. Most of the techniques used to extract character regions from an image can be used to segment characters. However, most algorithms that employ this stage assume that some joined characters will be segmented as one character and some characters will be segmented into more than one piece. Later stages of processing may need to attempt to split a region or join one or more to form a single character.

Character segmentation may be alternated with character recognition, recognition of easy characters being used to segment the remaining space into character regions.

Step 5: Normalisation of Character Size

After the image is segmented into characters, it is usual to adjust the size of the character region so that the following stages can assume a 'standard' character size. Of course, characters that have the same height vary in width, so the aspect ratio of the character region is important in the normalisation of character size.

It is important to note that size normalisation is usually only required when the techniques in the following stage depend on character size. Some character features such as topological ones are independent of size and consequently size normalisation is not a pre-requisite.

If the size of the character region is N rows by M columns and the 'standard' size is P rows by Q columns then size normalisation can be achieved by forming an expanded character region of PN rows by QM columns by pixel replication and then by sampling the expanded character region to obtain a P rows by Q

columns normalised character region. Note that Q may need to change for each character region so that the aspect ration of the character region P by Q is the same as the aspect ratio of the N by M character region.

Step 6: Feature Detection

The stages preceding Feature Detection are often described as preliminary processing. Feature detection and classification are the heart of OCR. Over the history of OCR many different feature detection techniques have been employed. Initially template matching was used to find the whole character as a feature, while later, subfeatures of the character were sought. Algorithm found the boundary outlines, the character skeleton or medial axis, the Fourier or Wavelet coefficients of the spatial pattern, various moments both spatial and grey-level, and topological properties such as the number of holes in a pattern. All have been used as features for classification of the character regions.

In selecting character features to be detected, algorithm designers were conscious of the need to detect features that characterised the characters being read independently of actual font and size. In addition, they were attracted to features that were invariant to image variations, such as translation, rotation, contrast, and colour as well as being representative of the characters when parts may be missing or have embellishments (particularly for handwritten text).

Of course, not all the desired properties of a feature can be found in any single measurement. Consequently, OCR algorithms usually detect a range of features, the total of which is sufficient to identify each character irrespective of the particular instantiation used in the text.

Just as the range of features detected is quite wide, so have been the techniques used to find them. Statistical correlation, boundary following, region thinning, mathematical morphology, various transforms such as the Fourier transform, texture masks, spatial pattern operators, and topological algorithms have all been used to detect features. It would be fair to say that those character recognition methodologies that segment the image into character regions before feature detection generally employ feature detectors that seek parts of letters and digits such as sharp corners, holes, concavities, and the like. Feature detection techniques that are applied before character segmentation usually attempt to find evidence for a specific character being present rather than part features. There is a growing research interest in investigating techniques for finding characters in text without prior character segmentation. This is driven by the problems encountered in cursive scripts where segmentation without recognition may be a more difficult problem than recognition itself.

Irrespective of the particular features detected or the techniques used to identify them, the output of the feature detection stage is a feature vector that contains the measured properties of the character region.

Step 7: Classification

The role of classification is to assign to a character region the character whose properties best match the properties stored in the feature vector of the region. Initially, OCR classifiers tended to be structural classifiers, that is, the designer devises a set of tests based on whether particular features exist in specific positions within the character region. For example, one might test for a sharp corner

in the lower left of a character region as a way of distinguishing between a 'B' and an '8'. The tests used were based on the designer's understanding of character formation and were a function of his training. The structural approach has more success with machine printed text than it does with handwritten text where detailed spatial features are less characteristic of the text than in machine printed form.

More recent classifiers have taken a statistical rather than structural approach to character recognition. The designer uses a set of training examples, that is, a set of character regions where the character present in the region is known, and then uses statistical techniques to build a classifier that matches the feature vector of a yet to be identified character to the feature vectors in the training set. The classifier assigns the character whose training vectors best match the feature vector of the unknown character. This approach can be embodied in Bayes Decision Rule techniques, Nearest Neighbour Matching techniques, Decision Tree approaches, and in Neural Networks. Basically all use the training set to build a set of decision rules that are then used to classify. In a technology like neural networks the training set determine the weights that encode the decision rules of the network.

Recent statistical classifiers have achieved excellent recognition rates. However, while the recognition rates are in the high 90% range, most have not achieved rates in excess of 99%. Generally, this is because a single classifier can be trained to get near perfect result in limited circumstances, say with a single font, but they lose their near perfect behaviour when they are trained with a wide range of characters. Consequently, modern classifiers are in fact conglomerates of classifiers coupled with a mechanism to return the best classification from the best classifier (or set of classifiers).

The arrangement for majority decision has itself taken two approaches, one being a parallel approach where all classifiers provide a classification and then the final output is selected from their results, and the other approach being a hierarchical one in which the first classifier selects a set of possible candidate characters and subsequent stages narrow the selection down into a final classification. In practice, a combination of the two approaches produces excellent classification rates and is currently the basis of state of the art technology.

A potentially interesting approach might be to couple these classification algorithms with a probabilistic algorithm based on a dictionary of words and word n-tuplets, where the decisions of the classifiers would be weighted by the probabilities provededprovided by the second algorithm.

Step 8: Verification

The final stage in OCR is verification. In this stage, knowledge about the expected result is used to check if the recognised text is consistent with what is expected. Such verification may include confirming that recognised words are indeed words found in a dictionary, or that vehicle licence plates are listed in the database of issued plates.

Verification cannot guarantee that the recognised text is correct but often it can detect errors. Errors can be handled by redoing the OCR but this time getting the second best classification, and repeating the verification. Alternatively, the classifier might provide an ordered list of possible classifications when the

certainty of correct classification is below some threshold. The verifier can use its information to determine the final classification.

Verification, while an important stage, is very much application dependent and often implemented independently of the OCR engine.

Chapter 4

Results and Analysis

4.1 Terminology Used

To avoid confusion and lengthy explanations in the analysis of the results and errors, all important terminology used in this chapter sections will be explained in this short glossary.

Due to the nature of the images used, a sparse vector was chosen as the best representation. Sparse vectors are vectors where, instead of memorizing every element of the vector, we only memorize the non-zero ones and their positions, saving both space and time.

An *eigenvector* of a transformation is a non-null vector whose direction is unchanged by that transformation. The factor by which the magnitude is scaled is called the *eigenvalue* of that vector

Structure tensors are a matrix representation of partial derivative information. In the field of image processing and computer vision, it is typically used to represent the gradient or "edge" information. It also has a more powerful description of local patterns as opposed to the directional derivative through its coherence measure. In the 2d case, the structure tensor matrix is formed as:

$$S = \begin{bmatrix} I_x^2 & I_X I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Eigen-decomposition is then applied to the structure tensor matrix S to form the eigenvalues and eigenvectors (λ_1, λ_2) and (e_1, e_2) respectively. These new gradient features allow a more precise description of the local gradient characteristics [7].

The Hessian matrix is the square matrix of second partial derivatives of a scalar-valued function. Given the real-valued function $f(x_1, x_2, \ldots, x_n)$, if all partial second derivatives of f exist, then the Hessian matrix of f is the matrix $H(f)_{ij}(x) = D_i D_j f(x)$, where $x = (x_1, x_2, \ldots, x_n)$. That is,

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

To evaluate the results of classification, we will observe the measures mentioned in section 2.3, specifically precision, recall, the $F_2 - measure$ and the $\kappa - statistic$, as well as the percentage of the correctly classified instances. We will also observe a number of confusion matrices to illustrate some common errors.

4.2 Datasets

The original dataset consisted of one 40x40 pixel image of each character (62 in total) in 20-point size from over one thousand True Type fonts, all extracted using a simple php script. Due to huge memory requirements of this dataset, the dataset used for this research was reduced – we look only at capital letters and have significantly reduced the number of fonts used to 47 fonts for the first series of experiments and 238 fonts for the subsequent experiment.

To extract the features from this dataset, a small Java application was written, that accesses the raster of the image and finds the needed data. A very modifiable image manipulation application, ImageJ (also written in Java), was used to apply various feature extraction algorithms to the dataset, using the ImageScience and FeatureJ plugins (for more info, see http://rsb.info.nih.gov/ij/).

All of the datasets were constructed in the same way: observing the image as a bi-level image, a sparse array was created, where the only relevant data was the ordered location of the black or white pixels (sparse vector) from the raster of the image.

This preprocessing (using ImageJ and FeatureJ) was applied to three different sets of images:

1. DATASET 1 : Plain, unaltered images. Elements of the sparse array were black pixels.

2. DATASET 2 : The same set of images, which was modified by calculating the structure tensor for each element and then constructing a new image from the lowest eigenvalues of the eigenvectors of this transformation.

3. DATASET 3 : The final set of images was constructed from the smallest eigenvalues of the Hessian of the original images. The Hessian transformation is used for discriminating locally between plate-like, line-like, and blob-like image structures.

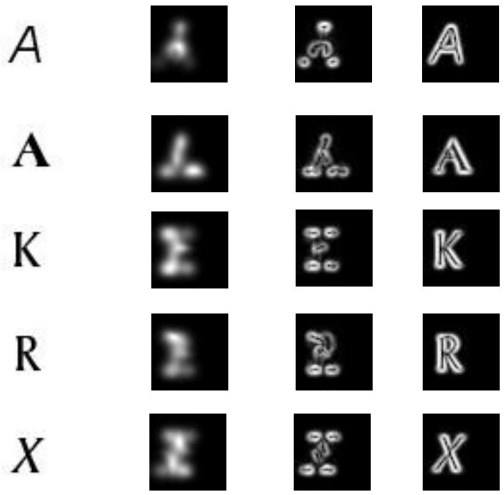

Figure 4.1: Examples of images from the four datasets. The first column are plain, unaltered images, second are the smallest eigenvalues of the structure tensors, the third are the smallest eigenvalues of the Hessian, while, in the last column we have images with the edge detection algorithm applied.

After this preprocessing, all of the datasets were processed in the Weka data mining environment, using the already mentioned algorithms (see section 2.2).

4.3 Results and Errors

To analyze the results of the research, we can look at a number of things: a general comparison of the algorithms used, analysis of the results of each specific algorithm across the different datasets, as well as a closer look at some of the most frequent errors and confusion points. The criteria for comparison are discussed in some depth in section 4.1 and 2.3.

4.3.1 Comparison of Algorithms

After the classification process, the algorithm that gave the best results was the support vector classifier - SMO, with 70% correctly classified instances as the worst result and a high of 88% correctly classified instances. The worst classifier, by far, was complement naïve Bayes, with 40%-70% correctly classified. Naïve Bayes and the J48 algorithm for training a pruned decision C4.5 tree stayed in the middle with 60-70% correctly classified instances for every dataset.

A detailed look at the performance of these classifiers for every dataset follows.

Dataset 1

Classification of the plain rasters of images from the smaller dataset shows the following:

J48 performed poorly here, with 67.18% correctly classified, with a κ-statistic of 0.6587.

Naïve Bayes and complement naïve Bayes gave almost identical results - 71.19% correctly classified and a κ-statistic of 0.7004 for naïve Bayes and just under 0.7 for complement naïve Bayes.

SMO classifies 83.55% of the instances correctly, with a κ-statistic of 0.8289

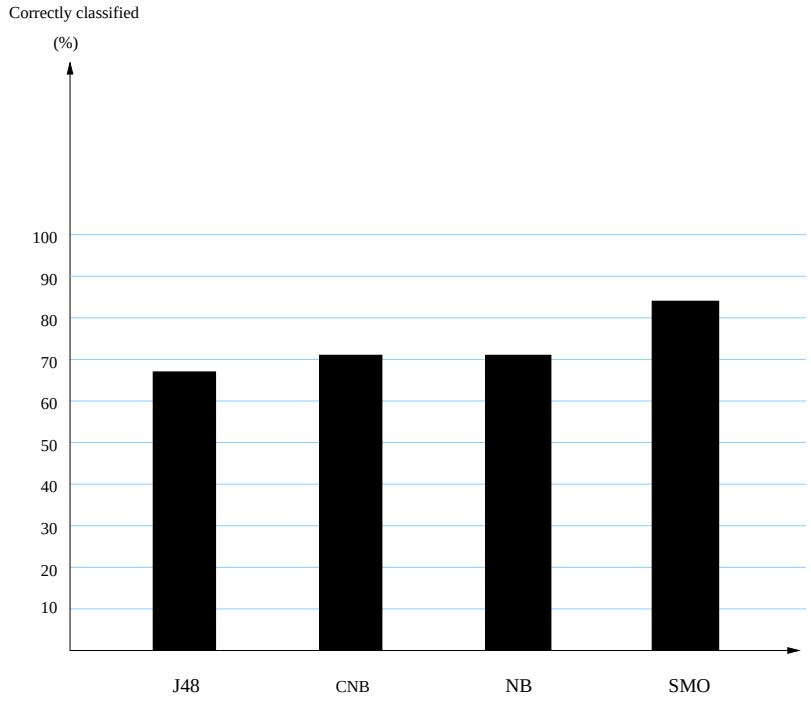

Figure 4.2: A graphical comparison of the results of the classifiers on DATASET 1

Common errors: naïve Bayes and complement naïve Bayes showed a number of common misclassifications, while SMO and J48 did not show grouped misclassifications. Naïve Bayes common misclassifications: "H" classified as "R", "D" as "C", and at least a few instances of every letter misclassified as "I". For a more detailed discussion, see section 4.3.2.

	J48	NB	CNB	SMO
correctly classified	67.18%	71.19%	71.05%	83.55%
$\kappa - statistic$	0.6587	0.7004	0.6990	0.8289

Table 4.1: Compiled results of the classifiers on DATASET 1

DATASET 2

Classification of the smallest eigenvalues of eigenvectors of the structure tensors shows similar, but slightly worse results:

Complement naïve Bayes was the worst classifier here, with only 40.42% correctly classified instances and a κ-statistic of 0.3804.

J48 and naïve Bayes also gave significantly weaker results than when working with just a plain raster - 62.77% for J48 and 68.08% for NB, with respective κ-statistics 0.6128 and 0.6681.

SMO again outperformed all other algorithms, with 84.04% correctly classified instances and $\kappa = 0.843$

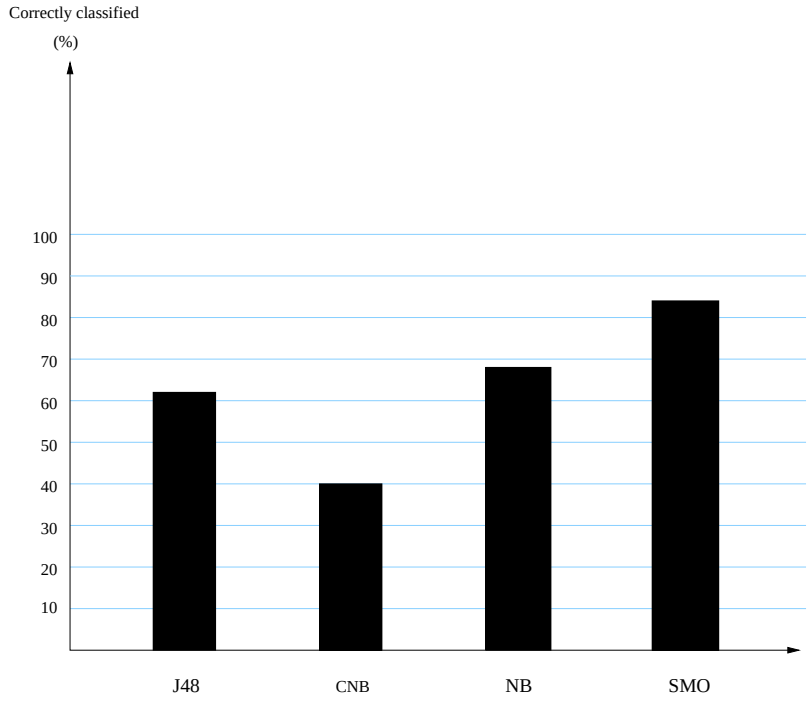

Figure 4.3: Graphical comparison of the results of the classifiers on DATASET 2

Common errors: complement naïve Bayes shows huge errors here, with more than 10% of all instances classified as an "I" and a significant grouping of misclassifications are "O" classified as "G", "H" as "M", "F" as "P" and "Y" as "V". J48 again shows no significant groupings of errors, except for a large number of instances of "P" classified as "F" and naïve Bayes shows similar results. The only significant misclassifications that SMO showed (more than three grouped incorrectly classified instances) are "H" as "K" and "O" as "C".

	J48	NB	CNB	SMO
correctly classified	62.77%	68.08%	40.42%	84.04%
$\kappa - statistic$	0.6128	0.6681	0.6990	0.843

Table 4.2: Compiled results of the classifiers on DATASET 2

Dataset 3

Even though visually very distinctive, this dataset proved as the hardest one for the algorithms.

Complement naïve Bayes and J48 both gave bad results - 41% for CNB and 46.56% for J48, with a κ-statistic below 0.45 for both.

Naïve Bayes showed slight improvements - 59% and $\kappa = 0.5736$.

SMO again gave the best results, although it performed significantly worse than on other datasets - 70.21% correctly classified instances and $\kappa = 0.6902$.

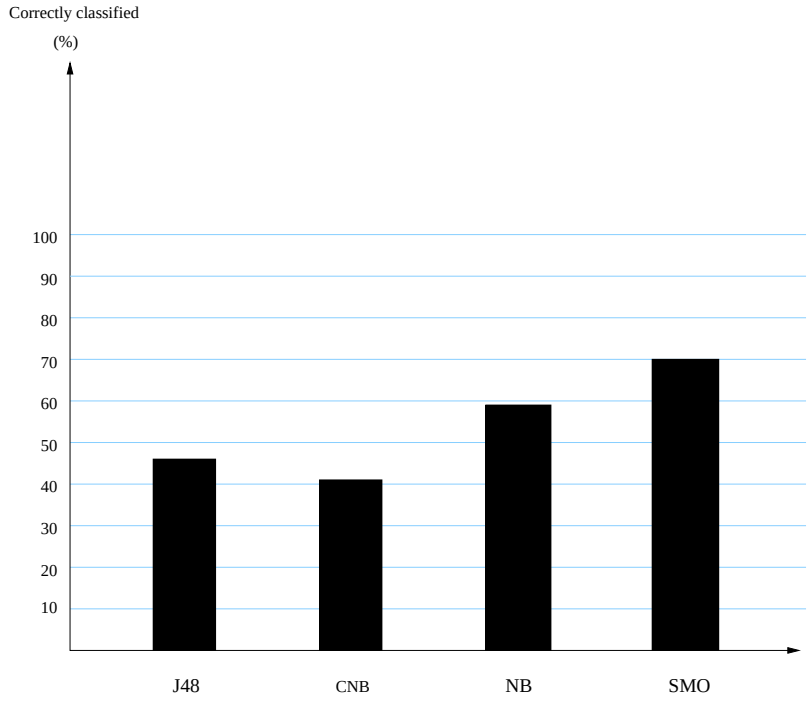

Figure 4.4: Graphical comparison of the results of the classifiers on Dataset 3

	J48	NB	CNB	SMO
correctly classified	46.56%	59%	41%	70.21%
$\kappa - statistic$	<0.45	0.5736	<0.45	0.6902

Table 4.3: Compiled results of the classifiers on Dataset 3

Dataset 1, large

Due to the size of this dataset and the time that classification takes, only three classifiers were trained and tested on it. Complement naïve Bayes, as the weakest algorithm for the previous three datasets, was left out.

The increase in size of the dataset helped improve each of the algorithms used:

Naïve Bayes showed the worst results here, with only 74.55% correctly classified instances, a small improvement over the smaller dataset. It's kappa statistic was 0.7353

J48 was improved the most with the increase in dataset size, with 78.47% correctly classified instances and $\kappa = 0.7761$

SMO again outperformed the other algorithms, with 88.96% correctly classified instances and κ=0.8852.

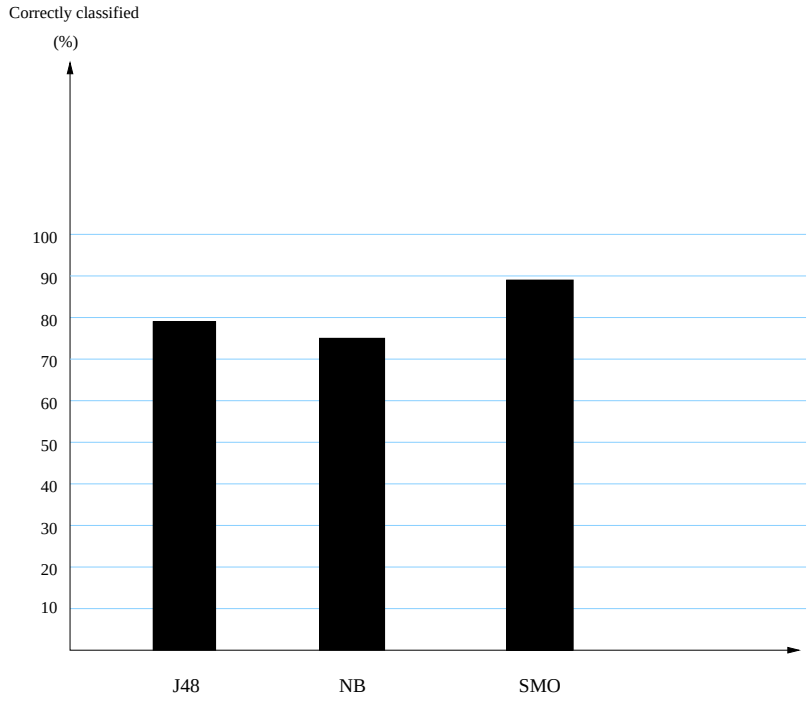

Figure 4.5: Graphical comparison of the results of the classifiers on a larger version of DATASET 1

Common errors: As the dataset is several times larger than the smaller one we worked with before, misclassification groupings are more obvious. Again, the most common error is classification of "P" as "F" and "F" as "P", which happens even with SMO. Other larger groups of misclassified instances are: "G" classified as "C", "H" as "N", "K" as "R" and "Y" as "V" for both J48 and naïve Bayes, as well as a large number of other misclassifications for NB (for a more detailed look, see section 4.3.2)

	J48	NB	SMO
correctly classified	78.47%	74.55%	88.96%
$\kappa - statistic$	0.7761	0.7353	0.8852

Table 4.4: Compiled results of the classifiers on a larger version of DATASET 1

4.3.2 A Detailed Look at the Performance of the Algorithms

Complement Naïve Bayes

Complement naïve Bayes showed the weakest results overall, it's kappa-statistic showing insignificant correlation between classified results and real groups in all three datasets where it was used. The average F-measure per class was never larger than 0.65, with values as low as 0.041 for some classes. Precision and recall did not show better results, with averages from 0.5 to 0.7.

J48

J48 also showed disappointing results in most datasets, with the exclusion of the DATASET 1, LARGE. Unlike the naïve Bayes algorithms, J48 did not show any significant grouping of classification errors, spreading them out instead. For the smaller datasets, the F-measure averages were between 0.4 and 0.65, with a low of 0.208, with similar results for precision and recall.

When applied to the larger dataset, the results changed significantly, with F-measure, recall and precision all larger than 0.785.

Some common classification errors for this algorithm are classifications of "P" as "F", "F" as "P", "V" as "Y", "Y" as "V" and "C" as "G" and "G" as "C", most of which are only visible on the larger dataset.

Naïve Bayes

Naïve Bayes took the middle ground between the weaker CNB and J48 and the much stronger SMO. Like the two previous algorithms, it showed insignificant correlation when used to classify the smaller datasets, with the exception of DATASET 1, where the correlation was statistically significant ($\kappa = 0.7004$). For the smaller datasets, it achieved F-measures between 0.5 (Hessian) and 0.7 (plain raster), with recall and precision moving within similar bounds. For the larger dataset, it showed slightly weaker results than J48, with all three measures between 0.75 and 0.77.

SMO

Of all the classification algorithms tested, SMO gave the best results, significantly outclassing all the other algorithms, with the weakest percentage of correctly classified instances of 70% and a maximum of 88.96%. Due to the large number of correctly classified instances, there were no visible groupings of errors, with the exception of the "F" as "P" and "P" as "F" misclassifications. The lowest average F-measure was 0.704, while the highest was 0.89 (a full 0.114 better than the best achieved by any other algorithm). The highest average values for recall and precision were, similarly, just below 0.9.

4.4 Conclusions

For this research, a dataset generated from over 1000 True Type fonts was trimmed down to a more manageable size. From that, smaller set of images, three different datasets were created through various basic feature extraction

methods and another, fourth dataset was created out of a larger subset of the original set of images.

These, smaller, datasets were then classfied using four different classification algorithms implemented in the WEKA library - naïve Bayes, complement naïve Bayes, J48 and SMO. The best three of these algorithms were also used to classify the larger dataset.

Of all the algorithms, CNB and J48 gave the worst results on the smaller datasets, while naïve Bayes gave better, but weak results. SMO, a classifier which trains a support vector machine, showed significantly better classification results than the others, with every clasification having a significant correlation to the dataset. This was somewhat offset by the much longer time for training the SMO classifier.

On the larger dataset, there was a significant improvement for all the algorithms applied, with all of the classifications showing a significant correlation to the dataset. J48 showed a huge improvement, while SMO classified almost 90% of the instances correctly. This leads to the conclusion that the original size of the dataset (47 fonts) was insufficient for continued research and brings up the question of when the gain from increasing the size of the dataset will stop being significant.

Common errors occurred in all classifications, leading to the conclusion that the feature extraction algorithms used were too simple. A better algorithm, or a combination of algorithms needs to be found to eliminate such errors.

Chapter 5

Possibilities for Improvement and Future Projects

As the feature extraction methods used were quite basic, different, more complex feature extraction algorithms should lead to an increase in correctly classified instances, as the common causes of errors can be eliminated through carefule combination features used.

It is obvious that the increase in the size of the dataset positively affects the number of correctly classified instances. It would be interesting to look into this connection carefully so that the right size for the training and test sets can be chosen, where the efficiency of training a classifier will not be significantly hurt by the large size of the dataset. Alternatively, it would be interesting to find out what is the point where the increase in the number of correctly classified instances becomes insignificant or levels off.

Increasing the number of algorithms used, including at least one neural network[8], should give somewhat more complete results, as well as allow comparison between using support vector machines and neural networks for classification in OCR, as neural networks are becoming increasingly popular in the field of OCR.

Bibliography

[1] Ian H. Witten and Eibe Frank, "Data Mining: Practical Machine Learning Tools and Techniques, second edition", Morgan Kaufmann Publishers, Elsevier, San Francisco, CA, 2005.

[2] Stephen V. Rice, Junichi Kanai and Thomas A. Nartker, "A Report on the Accuracy of OCR Devices", Information Science Research Institute, University of Nevada, Las Vegas, 1993.

[3] Ross Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[4] N. Sebe, Ira Cohen, Ashutosh Garg, Thomas S. Huang, "Machine Learning in Computer Vision", Springer, Netherlands, Dordrecht, 2005.

[5] J. Canny, "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 6, 1986, pp. 679-698.

[6] M. Sonka, V. Hlavac, R. Boyle, "Image Processing, Analysis, and Machine Vision, 2nd ed.", PWS Publishing, Pacific Grove, CA, 1999.

[7] G. Medioni, M. Lee and C. Tang, "A Computational Framework for Feature Extraction and Segmentation," Elsevier Science, Mar. 2000.

[8] Alexander J. Faaborg, "Using Neural Networks to Create an Adaptive Character Recognition System", Cornell University, Ithaca NY, 2000.

[9] G. Smith, "Optical Character Recognition", CSIRO Manufacturing and Infrastructure Technology, Australia, 2003.