Using Gradients and Pixel Information Assisted by Statistical Linguistic Post-Processing in an Automatic Handwriting Recognition System

Sybren Jansen, s1710184, sybren.jansen@rug.nl,

University of Groningen, Department of Artificial Intelligence

July 10, 2012

Abstract

Automatic handwriting recognition is still a hard problem in the field of AI. However, a lot of progress has been made in recent years. In this paper, a new automatic handwriting recognition system is proposed using an ensemble of classifiers. One classifier using radial zones and density histograms as feature set, the other using the histogram of oriented gradients (HOG). The ensemble was assisted by a top-down linguistic post-processor using modified interpolated Kneser-Ney smoothing. Results showed that the ensemble did outperform radial zones and density histograms, but did not outperform the HOG classifier. However, the ensemble showed more correct candidates in its candidate list making it more suitable for post-processing. Adding post-processing and using only a few candidates from the ensemble, resulted in a performance boost, reaching about 67.5% correct of known words.

Keywords: Handwriting recognition, Histogram of Oriented Gradients, Radial Zones, Linguistic post-processing, Kneser-Ney Smoothing, Viterbi

1 Introduction

Automatic handwriting recognition is still a hard problem in the field of Artificial Intelligence. Many databases contain handwritten text which need to be digitalized. Handwriting recognition is a difficult problem as it suffers from noisy, broken, multi-stroke, incomplete and ambiguous characters. Also, there is the problem of different writing styles between writers.

In recent years, though, a lot of progress has been made. Plamondon (2000) and Rehman and Saba (2011) give an extensive overview of current advances in off-line (and on-line) cursive script recognition. According to these surveys, many successful recognition techniques are based on statistical classifiers such as template matching, Bayesian classifier, Polynomial discriminate classifier, Fuzzy logic/rules, and k-Nearest-Neighbor (k-NN). However, some of these methods require all training samples be stored and compared for the classification process, which in some cases can be impractical. Hidden Markov Models-based classifiers (HMM), popular in speech recognition, remained highly successful for numeric recognition, achieving recognition rates above 98% (Britto Jr et al., 2004; Cai and Liu, 1999). For global word recognition the number of HMM-based techniques is increasing rapidly (Günter and Bunke, 2005). Recently, a number of studies have tried using Support Vector Machines (SVM) for numeral/character classification and got promising results above 99%. Gatos et al. (2006) successfully used SVMs for classification of words (88% recognition rate). Also, ensembles of classifiers were introduced in the fields of machine learning in 2003. Some have used it to improve both accuracy and reliability of the handwriting recognition systems (Günter and Bunke, 2003).

A shown, there are a lot of different setups that can be used for the task of handwriting recognition. Aside from the used classifier, different feature sets are possible. Among them are moments, histograms, zoning, contours and end points. Another one is the Histogram of Oriented Gradients (HOG), which was originally used for the task of human detection. However, good results have been found using local HOG features in handwritten word spotting tasks (Rodrìguez and Perronnin, 2008; Terasawa and Tanaka, 2009). We hypothesize that this might also perform well in our application of handwriting recognition.

HOG has some points in common with Lowe's SIFT key point descriptor (Lowe, 1999), but it differs in the fact that HOG normalizes the histograms in overlapping blocks and makes, as a result, a redundant expression. Also, HOG is computed without scale/orientation normalization and is computed everywhere in the image instead of only around the detected key points.

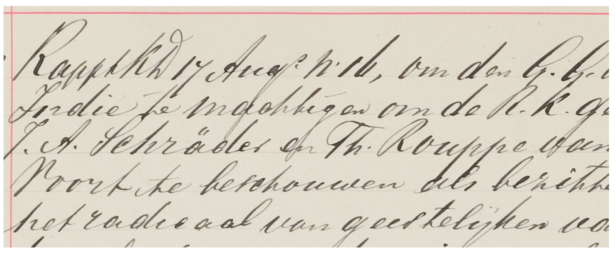
Next to the bottom-up approach used in every handwriting recognition system, Plamondon (2000) argues that language models are essential in recovering strings of words after bottom-up classification. Next to string matching between candidate words and a lexicon, dictionary statistics are used to improve results. The performance improvement derives from selection of lower-rank words from word recognition output when the surrounding context indicates such selection makes the entire sentence more probable. A popular method for this is using n-grams (Bunke, 2003). Using n-grams, the probability distributions over possible word sequences can be computed. However, our language allows for an infinite number of n-grams, whereas any training set allows for just a finite number of n-grams. Therefore, solutions like smoothing (Chen and Goodman, 1999; Katz, 1987) and backing-off (Rosenfield, 2000) can be useful.

In this paper a complete handwriting recognition system for handwritten text is proposed. Our system makes use of an ensemble of classifiers, in order to improve recognition results, and a top-down statistical linguistic post-processor, which uses word transition probabilities using n-grams, to be able to find the most likely word sequence. One feature set will focus on the pixel counts, whereas the other will focus on the gradients in that image (HOG). It is hypothesized that HOG, being a more sophisticated method, will outperform the simple pixel count features. Also, we hypothesized that both using an ensemble and using a top-down linguistic post-processor should improve performance. Our system also competed in a small competition held at the University of Groningen, The Netherlands, where three teams tested their system on a secret test set.

Each phase of the system will be discussed, starting with the preprocessing in Section 2.2, feature extraction in Section 2.3, bottom-up classification in Section 2.4 and the top-down selection in Section 2.5. In Section 3 the conducted experiments are described. Section 4 shows the results of our system, and Section 5 will discuss the implications of this research.

2 Methods

To built a robust handwriting recognition system, both bottom-up and top-down information was used. This section is divided into two parts covering both aspects of this process.



Rappt K.D 17 Aug. N. 16, om den G. G. v

Indie te machtigen om de R. K. ge

J. A. Schräder en Th. Rouppe van

voort te beschouwen als berichte

het radicaal van geestelijken va

Figure 1: A selection of a page from the 'Queen's Cabinet'.

2.1 Dataset

The handwriting material, which was used in the experiments, is historical handwriting from the "Queen's Cabinet" (*Kabinet der Koningin*, stored at the Dutch National Archive, *Nationaal Archief*, Den Haag) as shown in Figure 1. Accompanying every page are the predefined segmentation coordinates with the manually annotated words. Therefore, segmenting pages is no longer a problem. However, some annotations are not perfect and cover multiple words or even skip words. Therefore, the training and test set can contain some errors.

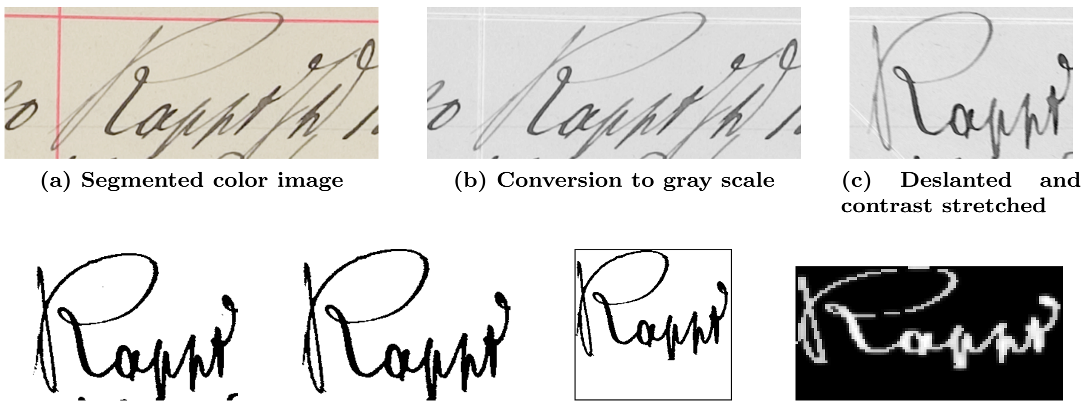
The entire set consists of 66 pages and is made up out of 2323 unique words, 6097 unique bigrams and 8653 unique trigrams. Having so little data to train on results in a sparse data set with 77.4% of the words occuring only one or twice and 95.6% of the words occurring less than 20 times. To obtain a more robust training set, distortions were applied to word images that do not occur frequently to create more training samples of these words. Varga and Bunke (2003) showed that generating synthetic training data can substantially improve the performance of a handwriting recognition system. For every instance that occurred less than 20 times, four additional instances were created. The used distortions were gray scale *erosion*, *dilation*, *opening*, and *closing*.

An extra line transcription file without accompanying word images was available for training the post-processor. Therefore, the linguistic model consists of 10703 unique words, 37100 unique bigrams and 61696 unique trigrams.

2.2 Preprocessing

To deal with differing illumination and to get robust features out of an image, binarization and normalization were applied to the word images extracted from the pages. In this section, the processes involved in binarization and normalization are discussed.

Conversion to gray scale A necessary step for some binarization techniques is to first convert the im-


(a) Segmented color image (b) Conversion to gray scale (c) Deslanted and contrast stretched

(d) Binarized image using Sauvola's method (e) Connected components removal (f) Centered to center of gravity (g) Inverted normalized image used for HOG

Figure 2: The different stages of preprocessing the word image containing the Dutch abbreviation 'Rappt'. The border around image (g) is for clarification reasons.

ages to gray scale. As shown in Figure 1, there are some red lines on the page which should be removed. To get rid of these lines and convert it to gray scale in one step, the green and blue channel of the RGB image were thrown away. The red lines, as a result, will become white and because there is only one channel left ranging from 0-255, the gray scale image is obtained. The result is shown in Figure 2(b).

Deslanting All words on the pages have a slant angle of 45°. In order to extract robust features from the image, it is useful to upright the images such that the characters are better separable. To upright the images, a mathematical transformation on the x-coordinate of every pixel was applied, as shown in Equation 2.1:

$$x' = x - (y - height) * -tan(\alpha) \qquad (2.1)$$

with $\alpha = 45$. After this transformation, there are still some parts of the next word in the image. This part of the image was cropped away, leaving only the pixels that are part of the word. The result is shown in Figure 2(c).

Binarization To extract the pixel count features, binarization is performed on the word images. Ntirogiannis et al. (2008) compared different binarization techniques which can be used to obtain a binarized image. For our proposed method, Sauvola's specialized text binarization method was used (Sauvola and Pietikainen, 2000), which showed very good results. It can separate text components from background in bad conditions, caused by uneven illumination or noise. The threshold $t_{(x,y)}$ is computed using the local mean, $m_{(x,y)}$, and local standard deviation, $s_{(x,y)}$, of the pixel intensities in a window centered around the pixel (x, y) (Equation 2.2).

$$t_{(x,y)} = m_{(x,y)} * (1 + k * (\frac{s_{(x,y)}}{R} - 1)) \qquad (2.2)$$

The whole word image was used as window, this is because all the word images are cut from the page beforehand and they are, as a result, already small. Also, the variation in illumination per word is very limited. Two parameters must be defined here: R, which represents the maximum value of the standard deviation (normally set to 128 for a gray scale document), and k, which is a parameter which takes positive values in the range [0.0, 0.5] and controls the value of the threshold in the local window such that the higher the value of k, the lower the threshold from the local mean. The local mean $m_{(x,y)}$ and standard deviation $s_{(x,y)}$ adapt the value of the threshold according to the contrast in the local neighborhood of the pixel. To assist Sauvola in this way, contrast stretching was used (Figure 2(c)). Because gray scale images are used, R is set to 128. Even with contrast stretching there is very little contrast in some parts of the words, therefore, k had to be set to a low value. After some experimentation, a value of $k = 0.05$ seemed to be the most optimal. The result is shown in Figure 2(d).

Unwanted connected components As shown in Figure 2(d), in some cases there are some unwanted connected components on the bottom or top of the image. Connected components were considered as unwanted when they touch the bottom or the top of the image and do not cross the vertical center of gravity of the image. The center of gravity, as in physics, can be seen as the point where the mass of the object is equally balanced. In binarized images, this corresponds to the point where there are as many black pixels to the one side as to the other side of the image.

Next to these unwanted connected components, also connected components of size smaller than 10 were considered as unwanted connected components (despeckling). The result is shown in Figure 2(e).

Centering and resizing For classification it is very important to have normalized features. To achieve this, some transformations were performed.

The first transformation was cropping away a certain percentage of the ink on the borders leaving only the core of the image containing the most information. Preliminary experiments resulted in getting the best performance with cropping away 10% of the ink.

The second transformations involved centering the image to the center of gravity. Some words contain ascenders, descenders, contain both, or none at all. The distribution of ink can vary a lot between words. Because most ink situates between the baseline and the corpus line of the words, centering the images to the center of gravity will result in the baseline and corpus line being in the middle of the image. With this, invariance to the position of a word is obtained. The result is shown in Figure 2(f). For clarification, a small border is placed around the image to show how the image is centered.

The last transformation involved rescaling the word images to a standard size. Every word on a page has a different width and height, and it is difficult to compute features from these images with the same meaning across all word images. Therefore, the images were rescaled, using bilinear interpolation, to a fixed resolution of 100×50 pixels (*width* $\times$ *height*).

2.3 Feature extraction

According to Suen (1986), there are two types of features: statistical and structural features. Statistical features are derived from statistical distribution of every point in the image such as moments, histograms and zoning. Structural features are based on geometric and topological features of characters such as contours and end points. To use all these features can lead to dimensionality problems commonly known as the curse of dimensionality. Therefore, to overcome this problem, a selection of features must take place.

A few feature sets were constructed to test which features work well and which features do not. The feature sets were constructed with the features mentioned below.

2.3.1 Density histograms

Density histograms give a compact representation of the binary image. Density histograms are constructed by the horizontal and vertical projection method (AL-Shatnawi and Omar, 2008). The horizontal and vertical projection method work by reducing the 2D array of data to 1D based on the pixels of the image:

$$P(x) = \sum_{y} I(x, y) \qquad (2.3)$$

where $P(x)$, in this example, is the horizontal projection of the image for row x, and $I(x, y)$ is the pixel intensity in the image at position (x, y), $I(x, y) \in \{0, 1\}$.

The horizontal (or vertical) projection method can not only be used to calculate the center of gravity, it can also be used as a feature for our classifier. Both the horizontal as the vertical projection of the image were used.

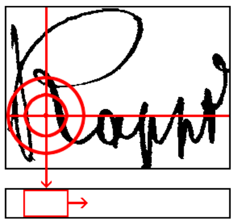


Figure 3: Radial zones method as a feature for the classifier. A sliding window slides horizontally over the input image and the number of black pixels in the 8 different segments are stored.

2.3.2 Radial zones

Radial zones work with the sliding window method (Fig. 3). The window slides on a fixed height from left to right. In our method the window slides on the vertical center of gravity, because this is where the information density is the highest. The window is shaped as a circle and divided into two major components, the outer ring and the inner circle. Both components are further divided into four segments: top right, bottom right, bottom left, and top left. In every segment, the total amount of black pixels is counted and stored. This gives us 8 features per window.

Preliminary experiments showed that a suitable measure for the diameter of the inner circle is defined as the height of the image divided by 8, whereas the diameter of the outer ring is determined as the height of the image divided by 4. The inner circle will cover the core of the letters, surrounding the vertical center of gravity. The outer ring covers the top and bottom of the lowercase letters and pieces of the ascenders and descenders.

Density histograms and radial zones formed the first feature set, focused on the pixels of the image.

2.3.3 Histogram of oriented gradients

The second feature set does not focus on the pixels of the image in particular, but on the orientations of the gradients in the image.

The image $I(x, y)$ was divided into $M \times N$ cells, with each cell having height H and width W. The cells do not overlap. Then, for each cell the HOG was computed. The gradients were assigned to the closest orientation given a number T of regularly spaced orientations. One can choose to use the entire range (2π) or use only half (π).

Computing the gradient histogram HOG can be computed on both binarized or gray scale images. Some preliminary experiments were conducted and better performance was observed using the gray scale images.

For using gray scale images, the background pixels had to be removed. The background of the image does not contain any relevant information and can thus be subtracted without any loss of information. The gray scale image, J, was obtained by leaving only the pixels in the original gray scale image which were detected as foreground pixels by Sauvola's Text Binarization Method.

To facilitate the succeeding feature extraction method, the image was inverted such that the background has intensity 0 and foreground has a new intensity of 255 minus the original intensity.

The obtained gray scale images have variable foreground gray levels over different samples. Because HOG is dependent on these gray levels, each image was smoothed and normalized such that the foreground pixels of each image had a standard mean of 210 and a standard deviation of 20, as was done in Liu and Suen (2008). The result is shown in Figure 2(g).

The gradient can be calculated using different operators: Roberts, Sobel, and more. Roberts and Sobel operators calculate two gradient components, g_x and g_y, in orthogonal directions. A study by Liu and Suen (2008) showed that Sobel slightly outperforms the Roberts operator. Therefore, a 3×3 Sobel mask was used to calculate the gradient.

At a pixel $J(x, y)$, the Sobal gradient $\mathbf{g} = (\mathrm{g}_x, \mathrm{g}_y)^{\mathrm{T}}$ is calculated by

$$\begin{aligned} \mathrm{g}_x(x,y) &= J(x+1,y-1) + 2J(x+1,y) \\ &\quad +J(x+1,y+1) - J(x-1,y-1) \\ &\quad -2J(x-1,y) - J(x-1,y+1), \end{aligned} \tag{2.4}$$

$$\begin{aligned} \mathrm{g}_y(x,y) &= J(x-1,y+1) + 2J(x,y+1) \\ &\quad +J(x+1,y+1) - J(x-1,y-1) \\ &\quad -2J(x,y-1) - J(x+1,y-1) \end{aligned} \tag{2.5}$$

The gradient magnitude m and direction θ were then obtained for a pixel at position (x, y) as

$$m(x,y) = \sqrt{\mathrm{g}_x^2 + \mathrm{g}_y^2} \tag{2.6}$$

and

$$\theta(x,y) = \angle(\mathrm{g}_x, \mathrm{g}_y), \tag{2.7}$$

where $\angle$ is a function that returns the direction of the Sobel gradient $\mathbf{g}$ in the range $[-\pi, \pi]$.

Next, the orientation bin which is the closest to $\theta(x, y)$ was computed and $m(x, y)$ was added to the corresponding bin. As assigning gradients to the closest orientation may result in aliasing noise, the gradient magnitude of a pixel can be shared between the two closest bins, as determined by a linear interpolation in the angle domain. Let α denote the angle between the pixel and the closest bin, then the distance to the second closest bin becomes $\frac{2\pi}{T} - \alpha$. Then the contribution of this pixel to the closest and second closest bin becomes respectively:

$$m(x,y)[1 - \frac{T\alpha}{2\pi}], \text{ and } m(x,y)\frac{T\alpha}{2\pi}. \tag{2.8}$$

To avoid boundary effects, it was assumed that outside the image the pixel values are 0.

Block normalization After obtaining the histograms for each cell, normalization was performed in overlapping blocks to get a redundant expression. A block was defined as a group of $h \times w$ cells. The block slides inside the window image, that means that $(N - h + 1) \times (M - w + 1)$ unique blocks exist. Each block is normalized such that their components sum to 1. The HOG descriptor is a concatenation of the normalized block descriptors. Each block has a dimensionality of hwT, resulting in HOG having $(N - h + 1)(M - w + 1)hwT$ dimensionalities.

Parameter estimation In the above algorithm, some parameters remained undecided. The number of cells ($M \times N$), the number of orientations T, and the number of cells in a block ($h \times w$).

For estimating optimal parameters, some preliminary experiments were conducted. Experiments with the number T and the range resulted in the best settings for our system being $T = 9$ and a range of 2π.

Using the study of Terasawa and Tanaka (2009) as a guideline, preliminary experiments were conducted with the number of cells and the number of cells in a block. The obtained settings were $M = 5$ and $N = 20$, which corresponds to a cell with dimensions 10×5 pixels ($height \times width$). For the blocks, 2×2 cells were used, which was also found to be the most optimal in Terasawa and Tanaka (2009).

2.4 Bottom-up classification

For both feature sets, k-nearest neighbor (k-NN) was used for classification. Both the classifier and how distances can be mapped to probabilities, which is needed for the linguistic post-processor, are described next.

k-NN The k-NN classifier assigns the input to the class having most examples among the k neighbors of input. All neighbors have equal vote, and the class having the maximum number of voters among the k neighbors is chosen. Ties are broken arbitrarily or a weighted vote is taken.

Simple nearest neighbor ($k = 1$) was used for our testing method. This means that the input is assigned to the class of the nearest pattern. As distance metric Euclidean distance was used.

Before classification, the mean word length and mean standard deviation were used to prune word classes that are much shorter or longer than the word that is currently being classified. For example, when it is known that the current instance has a word length of 300 pixels and the standard deviation over all training samples is 50, all word classes smaller than, say, 150 and larger than, say, 450 pixels are very unlikely to be correct. The word length of every training instance was extracted just before resizing and centering the images to the center of gravity, which gives us per class a mean word length and standard deviation. For classes containing only one instance, the mean standard deviation of all classes containing more than one instance was used as standard deviation. A class was considered a possible candidate if the instance to be classified fell in between the bottom and upper limit given by:

$$len(c) - 3 * sd(c) < len(i) < len(c) + 3 * sd(c), \quad (2.9)$$

where $len(c)$ denotes the mean word length of the current class c, $len(i)$ the word length of instance i to be classified, and $sd(c)$ denotes the word length standard deviation of class c. The word length of the instance thus had to fall inside three times the word length standard deviation of the class, which covers roughly 99% of all instances in that class. This way outliers are removed.

A list of N candidates was constructed for the post-processor. The post-processor will then decide from these candidates what the final output for this word instance will be (Section 2.5). When both sets of feature vectors are combined (one set for radial zone and density histogram features and one set for the HOG features), two simple nearest neighbor classifiers are used. In this case, both of these classifiers will propose a list of the N most likely candidate(s) for each word.

Statistical probabilities The linguistic post-processor, which will be described next, combines contextual probabilities with the probabilities from the bottom-up classifier. But, because the classifier only returns distances, these distances need to be transformed to statistical probabilities.

To do this, two frequency maps were constructed of the possible distances, one for same class samples, and one for different class samples. These maps were constructed by calculating the distance, d, of every training sample to every other training sample. If the two training samples had the same class, the same class frequency map for distance d was incremented. If, on the other hand, the two training samples had different classes, the different class frequency map for the distance d was incremented. Combining both maps and given a distance, the number of samples that are probably going to be classified correctly (or incorrectly) can be calculated.

The problem of having a small training set is that both frequency maps have to be interpolated on missing distances. For example, when a distance of, say, $d = 10$ and $d = 12$ both occur 100 times, and $d = 11$ does not occur, the problem arises of giving wrong probabilities to this last distance. To avoid this, smoothing could be applied, but with larger gaps in the data this problem is not solved. Instead, the frequencies were accumulated. The same class frequency map was accumulated from right to left (largest distance to smallest distance), the different class frequency map from left to right. As a result, two sigmoid graphs were obtained. The probability of classifying an instance correctly, given a distance d, becomes

$$P(d) = \frac{F_{same}(d)}{F_{same}(d) + F_{diff}(d)}, \quad (2.10)$$

where F_{same} is the accumulated frequency of the same class map and F_{diff} is the accumulated frequency of the different class map.

2.5 Top-down selection

Although the bottom-up pipeline already proposes the most likely word for each word-image, only the pixel information of the words is used to come to this conclusion. This process does not take into account any linguistic knowledge such as the appearance of previous words. Therefore, our method also proposes a top-down post-processor, specifically focusing on linguistic information and clues which can be derived from the interplay between words.

Instead of only receiving the most likely word for each segment in the text, each classifier provides the post-processor with N possible candidates, each with its own probability. From this list of candidates the post-processor then selects the most likely word using n-gram probabilities.

In the case where both classifiers are combined and they both suggest the same candidate, only the highest probability for this candidate is used. However, the classifier that proposes the lowest probability is not allowed to provide a new replacement candidate, leaving this classifier with only $N-1$ candidates to suggest. This way, the candidate that is proposed by both classifiers has an even higher probability of being selected due to lesser competition through other candidates.

Probabilities of n-grams All words, bigrams and trigrams are extracted from the manually annotated training data. Typically, the probability P of a unigram w_i is approximated by dividing the count C of w_i by the count of all words w in the training data, so $P_{uni}(w_i) \approx \frac{C(w_i)}{C(w)}$. The probability of an n-gram in the training data is calculated by dividing the number of times a specific n-gram occurs by the times the preceding word(s) occur. For trigrams, when $P(w_i)$ is approximated given the previous two words w_{i-2} and w_{i-1}, the result will be $P_{tri}(w_i|w_{i-2}w_{i-1}) \approx \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})}$.

Note that this way, due to the non-exhaustive nature of the training data, many possible word combinations will not be assigned a probability at all. For this reason, when the probability of a candidate is calculated, an interpolated model is used where the trigram, bigram and single word probabilities are combined. Consequently, when running into a novel word sequence where the highest order n-gram will result in a zero probability, information from lower order n-grams is also taken into account.

However, this does not solve all problems. For instance, the word sequence 'trouwen in Rotterdam' might not be in the training vocabulary while the sequence 'trouwen in Amsterdam' is. Therefore, it would be senseless to give this first sequence a probability of zero at the trigram level and rely only on lower order n-grams. Also, if a sequence like 'trouwen in Amsterdam' occurred only once in our small set of training data, it is very likely that it just showed up by chance and that its probability will be highly overestimated. Smoothing techniques try to overcome these difficulties by adding a small probability to word sequences that have not been encountered before and removing a bit of probability (a discount factor, D) from sequences that have. Different smoothing techniques have been proposed, such as Katz smoothing (Katz, 1987) and Jelinek and Mercer (Jelinek and Mercer, 1980). However, as Chen and Goodman (1999) found, a modified version of interpolated Kneser-Ney outperforms all previous techniques.

Modified interpolated Kneser-Ney smoothing The basic idea behind Kneser-Ney smoothing (Ney et al., 1994) is that some words appear in less contexts than others and should therefore be assigned a smaller probability in a new context. For instance, in our training data the word 'KD' occurs frequently, but only directly after the word 'Rappt'. Therefore without smoothing, the probability $\frac{C(\text{'KD'})}{C(w)}$ will be very high when running into a new context. Kneser-Ney smoothing on the other hand uses a modified probability distribution for lower-order n-grams, based on the number of contexts each word occurs in. This way, 'KD' will receive a relatively low probability when encountered in a novel word sequence where it does not follow the word 'Rappt'. The full set of formulae used to make up interpolated Kneser-Ney smoothing is shown in equations 2.11 until 2.13.

$$P_{KN}(w_i|w_{i-2}w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i) - D_{tri}}{C(w_{i-2}w_{i-1})} + \lambda(w_{i-2}w_{i-1})P_{KN-bi}(w_i|w_{i-1}) \tag{2.11}$$

$$P_{KN-bi}(w_i|w_{i-1}) = \frac{|\{v|C(vw_{i-1}w_i) > 0\}| - D_{bi}}{\textit{unique-trigrams}} + \lambda(w_{i-1})P_{KN-uni}(w_i) \tag{2.12}$$

$$P_{KN-uni}(w_i) = \frac{|\{v|C(vw_i) > 0\}| - D_{uni}}{\textit{unique-bigrams}} + \lambda\frac{1}{C(w)} \tag{2.13}$$

Here, *unique-trigrams* and *unique-bigrams* represent the total number of unique trigrams and bigrams in the vocabulary respectively. $|\{v|C(vw_i) > 0\}|$ gives the total number of unique word types that precede w_i in the training data. The same holds for $|\{v|C(vw_{i-1}w_i) > 0\}|$, which denotes the total number of word types that precede the bigram $w_{i-1}w_i$. λ is a normalization constant such that the probabilities sum to one.

Each n-gram order uses a different discount constant D. The modification to interpolated Kneser-Ney that Chen and Goodman (1999) propose is that a different discount constant should also be estimated for word (sequences) that occur once, twice or more than twice. This is done according to formulae 2.14 until 2.17 where C_k stands for the number of n-grams that occur k times.

$$Y = \frac{C_1}{C_1 + 2C_2} \tag{2.14}$$

$$D_{1count} = 1 - 2Y\frac{C_2}{C_1} \tag{2.15}$$

$$D_{2count} = 2 - 3Y\frac{C_3}{C_2} \tag{2.16}$$

$$D_{2+count} = 3 - 4Y\frac{C_4}{C_3} \tag{2.17}$$

Notice that this full set of discount constants is thus estimated for each of the three used n-gram orders.

Viterbi's algorithm Using the modified version of interpolated Kneser-Ney smoothing on trigrams, the probability of each candidate proposed by the classifiers can be estimated taking into account all other candidates that were proposed earlier. However, this would lead to an exponential time complexity of $O(N^T)$ per classifier, where N stands for the number of candidates proposed for one word and T represents the total number of words in the text. To reduce this complexity, Viterbi's algorithm (Forney, 1973) is used for finding the most likely sequence of words given the sets of possible candidates. This method has successfully been used before in the field of handwriting recognition (Hu et al., 1996; Senior, 1998). In order to obtain more flexible trigram probabilities, Vitberi's algorithm was implemented on a trigram scale in which an extra depth of candidates is being tracked. This increases the time complexity of the algorithm from $O(N^2 * T)$ to $O(N^3 * T)$ which is acceptable and as preliminary experiments showed, it increased performance. The formulae that make up Viterbi's algorithm on this third order level are given by:

$$V_{1,k} = P(w_{1,k}|I_1) * \pi_k \tag{2.18}$$

$$V_{2,l,k} = P(w_{2,k}|I_2) * P_{KN}(k|l) * V_{1,l} \tag{2.19}$$

$$V_{t,l,k} = P(w_{t,k}|I_t) * \max_{x \in S_{t-2}} (P_{KN}(k|x,l) * V_{t-1,x,l}) \tag{2.20}$$

Here, π_k states the initial probability of a candidate k. A set of candidates at word position t is contained in state S_t. Therefore, in $w_{t,k}$, t denotes the position of a word w and k denotes a candidate at position t, so $w_{2,3}$ indicates candidate three at word position two in a text. A candidate at position $t-1$ is denoted by l. $P(w_{t,k}|I_t)$ is the probability that the classifier gave to a candidate, given the word image at the corresponding position. Finally, $V_{t,l,k}$ is the probability of the most likely path (the 'Viterbi path') through the sequence of candidate lists, ending at the bigram $w_{t-1,l}, w_{t,k}$. By using back pointers that remember which candidate is used in the second equation, the Viterbi path $x_1, ..., x_T$ can be found:

$$x_{T-1}, x_T = \underset{l \in S_{T-1}, k \in S_T}{\arg\max} (V_{T,l,k}) \tag{2.21}$$

$$x_{t-1} = Tr(t, x_t) \tag{2.22}$$

$Tr(t,x)$ is a traceback function that returns the value of x which was used to compute $V_{t,l,k}$ in equation 2.20 if $t > 1$. When $t = 1$, the function simply returns the candidate k which maximized the probability of the Viterbi path.

3 Experiments

To test the performance of the handwriting recognizer, two experiments were conducted. In the first experiment, three different experimental setups were used to test how the different types of features perform separately and combined. In the first setting only radial zone and density histogram features were extracted, while in the second setting solely HOG features were used. These setups shall be referred to as PC and HOG respectively. In the third setup (ALL), both feature sets were combined. In order to decide which proposed candidate to use, simply the candidate with the highest probability from any of the two classifiers was selected. Although no linguistic post-processing followed the classification in any of these settings, the number of proposed candidates N was set to 10 in order to explore the difference between proposed words.

In the second experiment, the ALL-setup was compared against a setup in which linguistic post-processing was added. This setting will be referred to as ALL_PP. In this setting, a number of different configurations for N were tested in order to explore the effects of the post-processor. The configurations that were chosen are $N = 10$, $N = 2$, $N = 1$.

k-fold cross-validation was used to find a reliable estimation of the predictive performance of the different setups. Due to the size of the data set which consists of 66 pages, k was set to 11. From each fold, we obtained the classification results of the six pages within that fold.

4 Results

The results of the two experiments are described below. For clarity, the results of all experimental conditions are put together in Table 1.

4.1 Experiment 1

One-way ANOVA was conducted to compare the effect of the type of feature on the percentage of correctly classified words on a page in HOG, PC and ALL conditions. There was a significant effect of the feature set on classification performance at the $p < 0.05$ level for the three conditions ($F(2, 195) = 4.32$, $p = .01$). A Tukey's post-hoc test revealed that the percentage of correctly classified words was significantly higher in the HOG condition (M = 55.94, 95% CI[54.43, 57.46], $p < .01$) and the ALL condition (M = 55.51, 95% CI[53.99, 57.03], $p < .05$) compared to the PC condition (M = 52.99, 95% CI[51.47, 54.50]). There were no statistically significant differences between HOG and ALL ($p = .94$). These results indicate that both a single classifier using solely HOG features and the combination of two classifiers using both types of features outperform a single classifier which uses solely radial zone and density histogram features. However, there is no indication that HOG or the combination of features perform better than one another.

Table 1: Results experiments 1 and 2

	% Correct	% Correct in c-list	% Correct known words	% Correct known words in c-list
$\text{HOG}_{\text{N=10}}$	55.94	70.99	64.46	81.72
$\text{PC}_{\text{N=10}}$	52.99	69.35	61.04	79.88
$\text{ALL}_{\text{N=10}}$	55.51	**73.99**	64.30	**85.63**
$\text{ALL\_PP}_{\text{N=10}}$	51.30	**73.99**	59.15	**85.63**
$\text{ALL\_PP}_{\text{N=2}}$	57.07	67.57	65.99	78.20
$\text{ALL\_PP}_{\text{N=1}}$	**58.80**	62.26	**67.49**	72.10

Results of the different experimental setups put together. The first two columns show the overall classification performance and the percentage of correct words occurring in the candidate list. The third column shows the percentage of correct words when only the words that could have been possibly known are taken into account, i.e., words that were not encountered in the training data are excluded from the calculation. The last column displays the percentage of previously encountered correct words in the candidate list. The highest scoring setup(s) are displayed in bold.

To analyze the possibilities that each condition offers for the post-processor, it is also necessary to look at the percentage of correct words that can be found in the candidate lists. This percentage gives the upper-limit that the post-processor could eventually reach, because words that are not in the candidate list will never be found otherwise. A one-way ANOVA compared the effect of the feature type on the percentage of correctly classified words present in the candidate lists in *HOG*, *PC* and *ALL* conditions. There was a significant effect of the feature set on this percentage at the $p < .05$ level for the three conditions ($F(2, 195) = 7.92$, $p < .001$). A Tukey's post-hoc test showed that the percentage of correct words in the proposed candidate lists was statistically significantly higher in the *ALL* condition (M = 73.99, 95% CI[72.34, 75.63]) when compared to the *HOG* condition (M = 70.99, 95% CI[69.34, 72.68], $p = .01$) and the *PC* condition (M = 69.35, 95% CI[67.70, 71.00], $p < .001$). There were no significant differences between *HOG* and *PC* ($p = .17$). These results show that the combination of the candidates proposed by each classifier contains significantly more correct words than the lists of candidates proposed by each classifier individually. In other words, the candidates proposed using HOG features differ from the candidates proposed using radial zone and density histogram features, although both feature sets propose about the same number of correct instances. By combining these two candidate lists, the possibility of having the correct word in the final candidate list increases. For this reason, the *ALL* setting seems to offer more possibilities for the post-processor.

4.2 Experiment 2

Experiment 1 showed that the *ALL* condition proposed significantly more correct words in its candidate list when $N = 10$ than the other two conditions. We therefore compared the *ALL* condition to the *ALL_PP* condition with $N = 10$ in order to see how post-processing would affect the number of correctly classified words. Barlett's test showed a violation of homogeneity of variances ($X^1 = 9.3$, $p < .05$), therefore a two-sample t-test assuming unequal variance was used to compare the effect of post-processing on the percentage of correctly classified words on a page in *ALL* and *ALL_PP* conditions with $N = 10$. There was a significant difference in classification performance for post-processing with 10 candidates (M = 51.30, SD = 9.06) and no post-processing (M = 55.51, SD = 6.16) conditions; $t(130) = 3.12$, $p < 0.01$. This implies that linguistic post-processing makes performance of our classifier significantly worse when the number of candidates the post-processor may choose from is set to 10. These results did not seem very promising, however two more tests were conducted in which N was set to 1 and 2 respectively.

Bartlett's test showed that variances of the percentage of correct words in *ALL*, $ALL\_PP_{N=2}$ and $ALL\_PP_{N=1}$ conditions were homogeneous ($X^1 = 3.3$, $p = .19$). Therefore a one-way ANOVA was carried out which indicated a significant effect at the $p < 0.05$ level for the three conditions ($F(2, 195) = 3.85$, $p < .01$). A post-hoc Tukey's test revealed a significantly higher percentage of correct words in the $ALL\_PP_{N=1}$ condition (M = 58.80, 95% CI[57.15, 60.46], $p < .01$) when compared to the *ALL* condition (M = 55.51, 95% CI[53.85, 57.17]). No significant effect was found between the $ALL\_PP_{N=2}$ condition (M = 57.07, 95% CI[55.41, 58.72]) compared to the $ALL\_PP_{N=1}$ condition ($p = .14$)or to the *ALL* condition ($p = .19$). These results indicate that post-processing can significantly improve the performance of the bottom-up classification, but only when the number of candidates it may choose from is set to at the most 1 per classifier.

Table 2: Results of the competition

Team	Avg. edit dist.	% Correct
Team 2	1.77	65.62
Our team	2.01	55.35
Team 1	2.14	51.47

4.3 Competition

In a small competition at the University of Groningen, our automatic handwriting recognizer using the $ALL\_PP_{N=1}$ setup competed against two other student teams on a secret test set of 71 pages. These results are shown in Table 2. As can be seen, our recognizer outperforms Team 1 which made use of more holistic features such as the number of ascenders and descenders in a word. This team also focused on the contour of a word using Fourier descriptors. However, our system is outperformed by the recognizer of Team 2 in which HOG features with different parameter values were combined with multi zoning, a different method for pixel count.

4.4 Error analysis

To better understand the errors the proposed system makes, an analysis was performed on the output of the system using the ALL_PP setting with $N = 1$. The number of encountered training instances for correct and incorrect words were recorded and analyzed. Only the incorrect words that could have been correct, words that occur in the training data, were taken into consideration. The results are shown in Figure 4.

A data point is considered a potential outlier, indicated by a circle, if the point exceeds $< Q1 - 1.5 * IQR$, or $> Q3 + 1.5 * IQR$, where $Q1$ is the lower quartile, $Q3$ the upper quartile, and IQR is the inter-quartile range. Extreme outliers are highlighted with a solid dot ($< Q1 - 3 * IQR$, or $> Q3 + 3 * IQR$).

The original scores were ranked ordered and a Mann-Whitney U-test was used to compare the ranks for the $n = 7923$ words that were correctly classified and the $n = 3816$ words that were incorrectly classified and occur in the training set. The results indicate a significant difference between these two types of words regarding the number of instances per word, $U = 5953565$, $p < 0.001$, with the sum of the ranks equal to 55671529 for correctly classified words and 13236401 for incorrectly classified words. This indicates that, in general, the incorrectly classified words (median = 11) have a different number of encountered training instances than the correctly classified words (median = 159). This number is generally lower than the number for the correctly classified words, as shown in Figure 4.

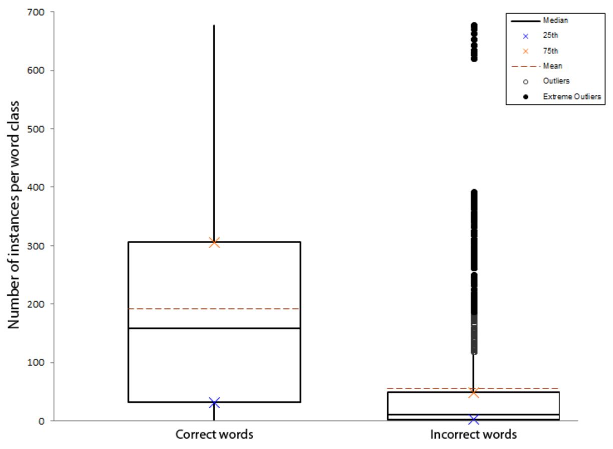


Figure 4: The number of encountered training instances for correct and incorrect words. Only the words that could have been correct are shown here.

5 Discussion

In this paper, a new handwriting recognition system was proposed using an ensemble of classifiers and a statistical linguistic post-processor. On the one hand, radial zones together with density histograms was used for one classifier (PC), focusing on the pixel information in the image. On the other hand, histogram of oriented gradients was used for the second classifier (HOG). HOG outperformed PC, confirming our first hypothesis that the more advanced feature method outperforms the more simple pixel based feature. The ALL condition, where both classifiers were combined, performed significantly better than the PC condition, although it did not perform better than the HOG condition. However, ALL did have more correct candidates in its candidate list, making it more appropriate for the linguistic post-processor. Therefore, partially confirming the second hypothesis that an ensemble of classifiers can perform better than single classifiers.

An explanation for this difference in performance could be that HOG does not focus on the pixels itself, but on the orientations these pixels have, providing a more holistic view of the image. After block normalization, the HOG-descriptor makes a redundant expression of the image. Also, HOG uses gray scale images for feature extraction. Radial zones, on the other hand, focuses on the number of black and white pixels in a specific radius. Density histograms do the same but in less regions. When ensembling both classifiers, both the good and bad properties are combined. Both classifiers provide correct and incorrect words with high probabilities. Therefore, there is a good chance that the performance will not be higher than a single classifier, as was observed with HOG.

Using n-grams together with modified interpolated Kneser-Ney smoothing resulted in slightly worse performance when the number of candidates was set to 10. However, giving the post-processor less candidates to choose from resulted in significantly better performance, with the best results obtained in the $N = 1$ condition (58.80% correct, 67.49% correct of known words). This confirms our third hypothesis, being that linguistic post-processing can indeed improve performance.

One explanation for the drop in performance, when $N = 10$, could be that the post-processor has to much influence compared to the bottom-up classifier. For example, in the situation where there are two candidates, say 'A' and 'B', with respectively 90% and 50% probability, and 'B' has a statistical very high probability, while 'A' is a word that only occurred once in the training data, the post-processor will probably favor 'B' over 'A', while the bottom-up classifier is quite certain about the word being 'A'. If this would be the case, a solution would be to only use post-processing when the bottom-up classifier is not certain of which word it is. A problem would then be to set the right threshold for the post-processor to kick in.

Another explanation could be that the bottom-up classifier uses incorrect probabilities. In our proposed system, statistical probabilities are calculated comparing every instance to every other instance and saving everything in just two frequency map. After that, the probability of a word being correct is calculated as the number of times a word was classified correctly, given the computed distance, divided by the total number of words having that distance. However, there could be situations where this is not the most optimal solution. Suppose that there is a very small word, a one digit number for example, that shows very little variance in writing. When comparing an instance of that word to other instances of the same word, there will probably be very little variance in distance. Whereas when you take a very long word, of, say, 12 characters, there could be very large variance in calculated distances. Therefore, the one digit number will have 99% probability when distance is, say, 0-2, and have 20% probability when distance is 4. The long word will have 99% probability when distance is 0-10, and have 20% probability at distance 16. Combining these results for all words could lead to assigning wrong probabilities to distances for some words. A solution would be to compute for each word a probability map. For word classes having only one instance for training, one could still use the average map. Memory would probably cause no problems, as the probability maps are of small size.

As our error analysis showed, the incorrect words occurred significantly less in the training data than the correct words occur. This is to be expected. When you have more instances of a particular word to train on, the chance of classifying that word correctly increases. A logical solution for this problem would be to obtain more training data. Distortions were already added for words occurring less than 20 times. Additional distortions could be added to increase performance even more. Some possible distortions could be: horizontal or vertical scaling, baseline bending, connected component level distortions, or thinning and thickening.

An additional point of improvement could be to use a different distance metric than Euclidean distance. A lot of distance metrics exists, like Manhattan and Mahalanobis distance, which takes into account the correlations of the data set and is scale-invariant.

To conclude, our proposed handwriting recognition system performs reasonably well reaching about 67.5% correct of words that could have been correctly classified. The ensemble of classifiers, combining radial zones and density histograms together with histogram of oriented gradients resulted in more correct words in the candidate list. Combining the ensemble with linguistic post-processing gave the system a significant boost in performance. However, there are still some improvements that could be made.

References

A. AL-Shatnawi and K. Omar. Methods of arabic language baseline detection the state of art. *IJCSNS International Journal*, 8(10):137–143, 2008.

A. Britto Jr, R. Sabourin, F. Bortolozzi, and Suen C.Y. Foreground and background information in an hmm-based method for recognition of isolated characters and numeral strings. In *Proceedings of the 9th internation workshop on frontiers in handwriting recognition*, pages 371–376, 2004.

H. Bunke. Recognition of cursive roman handwriting past, present and futurey. *Seventh Int. Conf. on Document Analysis and Recognition*, page pages 44846, 2003.

J. Cai and Z-Q Liu. Integration of structural and statistical information for unconstrained handwritten numeral recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 21(3):263–270, 1999.

S.F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, 1999.

G.D. Forney. The viterbi algorithmn. *Proc. IEEEn*, 61:268278, 1973.

B. Gatos, I. Pratikakis, A.L. Kesidis, and S.J. Perantonis. Efficient off-line cursive handwriting word recognition. In *Proceedings of the 10th international workshop on frontiers in handwriting recognition*, 2006.

S. Günter and H. Bunke. Ensembles of classifiers for handwritten word recognition. *International Journal on Document Analysis Recognition*, 5:224–232, 2003.

S. Günter and H. Bunke. Off-line cursive handwriting recognition using multiple classifier systems. on the influence of vocabulary, ensemble, and training set size. *Optics Lasers Eng*, 43(3-5):437–454, 2005.

J. Hu, M.K. Brown, and W. Turin. Hmm based on-line handwriting recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(10):1039–1044, 1996.

F. Jelinek and R.L. Mercer. Interpolated estimation of markov source parameters from sparse data. *In Proceedings of the Workshop on Pattern Recognition in Practice*, 1980.

S.M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401, 1987.

C.L. Liu and C.Y. Suen. A new benchmark on the recognition of handwritten bangla and farshi numeral characters. *Pattern Recognition*, 42(12):3287–3295, 2008.

D.G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, Corfu, Greece, 1999.

H. Ney, U. Essen, and R. Kneser. Interpolated estimation of markov source parameters from sparse data. *Computer, Speech and Language*, 8:138, 1994.

K. Ntirogiannis, B. Gatos, and I. Pratikakis. An objective evaluation methodology for handwritten image document binarization techniques. In *Proc. of the 8th International Workshop on Document Analysis Systems (DAS'08)*, pages 217–224, 2008.

R. Plamondon. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1): 63–84, 2000.

A. Rehman and T. Saba. Off-line cursive script recognition: current advances, comparisons and remaining problems. *Artif Intell Rev*, 37:261–288, 2011.

J.A. Rodrìguez and F. Perronnin. Local gradient histogram features for word spotting in unconstrained handwritten documents. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2008.

R. Rosenfield. Two decades of statistical language modeling: Where do we go from here? *Proc. of the IEEE*, 22:12701278, 2000.

J. Sauvola and M. Pietikainen. Adaptive document image binarization. *Pattern Recognition*, 33:225–236, 2000.

A.W. Senior. An off-line cursive handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):309 – 321, 1998.

C.Y. Suen. Character recognition by computer and applications in handbook of pattern recognition and image processing. In *Young TY, Fu K-S (eds)*, pages 569–586. Academic Press Inc., San Diego, 1986.

K. Terasawa and Y. Tanaka. Slit style hog feature for document image word spotting. In *Proceedings of the 10th International Conference on Document Analysis & Recognition*, pages 116–120, 2009.

T. Varga and H. Bunke. Generation of synthetic training data for an hmm-based handwriting recognition system. In *7th Int. Conference on Document Analysis and Recognition*, 2003.