AUTOMATIC ESTIMATION OF THE READABILITY OF HANDWRITTEN TEXT

Andreas Schlapbach and Frank Wettstein and Horst Bunke

Institute of Computer Science and Applied Mathematics
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{schlpbch, wettstei, bunke}@iam.unibe.ch

ABSTRACT

In this paper, the problem of estimating the readability of handwritten text is addressed. The estimation problem is posed as a two class classification problem where a text is classified as either *readable* or *unreadable*. A classifier is trained on this two class classification problem. In the training phase, for each text a number of features are extracted. At the same time the recognition rate achieved on the text is determined. Based on the recognition rate, each feature vector is labelled, i.e., assigned to one of the two classes. The labelled data is then used to train a classifier. The *k*-Nearest Neighbour (*k*-NN) and the Support Vector Machine (SVM) classifier are evaluated in this work. Both classifiers show promising results on a test set of 715 text lines from 20 writers.

1. INTRODUCTION

In handwritten text recognition, false recognition is expensive. An example is an address reading system where a letter with a falsely recognized address is directed to a wrong destination, possibly returned, and then needs to be reprocessed. Moreover, false recognition is expensive in the sense that a needless recognition attempt has been made. Therefore, it would be interesting to filter out unreadable text before it is fed into a text recognition system. The text that has been filtered out can then be processed by a human or a specialised recognition system, thus decreasing the overall cost.

The readability problem can be posed as a classification problem [10]. A text is classified as either *readable* or *unreadable*. A classifier can be trained on this two class classification problem. The training data consists of a set of feature vectors extracted from the text together with the label of the class the text belongs to. In the classification phase, feature vectors are first extracted from the text and then classified. If they are classified as *readable* they are passed on to the text recognition system. Otherwise, if they are classified as *unreadable*, they are filtered out. Fig. 1 gives a schematic overview of readability classification.

In the present paper, the problem of estimating the readability of handwritten text is studied. Only very few works on

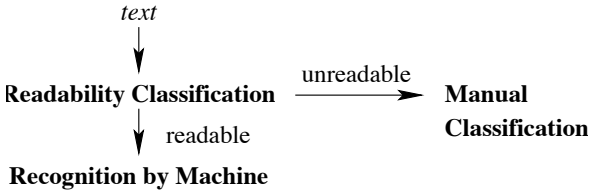

Figure 1: Readability Classification

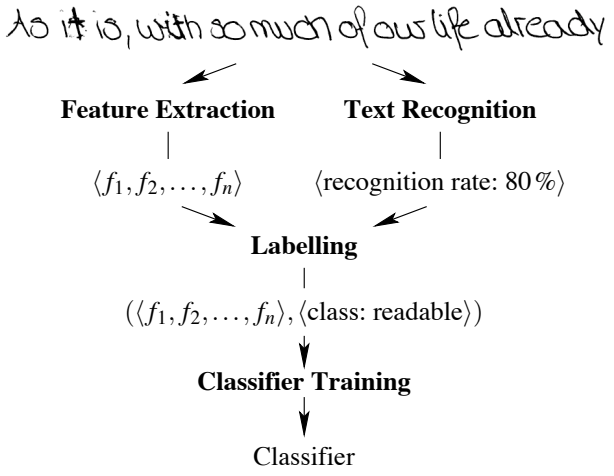

Figure 2: Training of the Readability Estimation System

readability estimation exists. The problem of predicting the accuracy of an Optical Character Recognition (OCR) system for machine printed text has been studied in [2, 12]. Other, loosely related works deal with writer identification (for an overview of recent work see [20]), handwriting style classification [15], and sub-category classification analysis of handwriting [5]. However, to the best of the authors' knowledge, the particular problem of estimating the readability of handwritten text has never been addressed in the literature before.

The rest of this paper is structured as follows. In the next section, the proposed readability estimation system is introduced. Section 3 describes the data and the experimental setup. The results are presented and discussed in Section 4. Finally, Section 5 concludes the paper and proposes future work.

2. SYSTEM DESCRIPTION

This section describes the readability estimation system. In the training phase, each text is transformed into a feature vector and the text recognition rate achieved on the text is determined. Next, the feature vectors are labelled, i.e., assigned to one of the classes *readable* or *unreadable* depending on the recognition rate. The labelled data is then used to train a classifier. A *k*-NN and a SVM classifier are evaluated in this work. A systematic overview of the training procedure is shown in Fig. 2. In the classification phase, the same features as the ones used for training are extracted from a text and then passed on to the trained classifier, which assigns the text to the class *readable* or *unreadable*.

The rest of this section is structured as follows. Fea-

ture extraction and text recognition are presented in Subsections 2.1 and 2.2, respectively. Labelling is defined in Subsection 2.3 and the classifiers are described in Subsection 2.4. Finally, Subsection 2.5 discusses methods to find a good subset of the original feature set for classification.

2.1 Feature Extraction

This paper considers individual text lines as the basic units. The features extracted from a text line image have initially been defined for writer identification and have shown very good results on large sets of writers [8, 20]. These features are able to distinguish writers with diverse writing styles. Therefore, it is reasonable to apply them to the readability classification task as well. The readability classification task can thus be viewed as the problem of distinguishing two writers, one with readable and one with unreadable handwriting.

Before feature extraction, a text line image is normalised. The normalisation operations are designed to improve the quality of the features. Only a short description of the normalisation operations is given here; for more details see [8]. In the first step, the grey scale text line images are binarised using Otsu's binarisation algorithm [18]. Next, the text line images are clipped. Finally, Hilditch's thinning algorithm is applied which iteratively refines a text line image until a stable image is achieved [9].

The 100 extracted features can be divided into five groups. Only a short overview of the features is given here; a detailed description can be found in [8]. The first group of *basic features* describe basic properties of a text line, such as the skew, the slant angle, the middle and the lower region of a text line, the transition width, and the average character width.

The second group of *component features* describe the writing style of a writer with respect to its connectedness, i.e., whether a word is written in one single stroke or in multiple strokes. Each connected component in a text line image is described by its bounding box. From the connected components of a text line, various measures are calculated. They include the average distance of two successive bounding boxes, the average distance of two consecutive words, or the average within-word distance of connected components.

The basic idea behind the third group of *fractal features* is to measure how the area of a handwritten text grows when a dilation operation is applied on the image [22]. The writing is dilated using circular and ellipsoidal kernels of various size resulting in an *evolution graph*. The evolution graph is divided into three more or less linear segments. The slope of each of the three straight line segments is used as a feature to characterise the handwriting.

The lower (upper) contour of a text line is defined as the sequence of pixels obtained if the lower-most (upper-most) pixel is considered. The characteristic contour is calculated by deleting the gaps that occur between the contours of single components and by concatenating these components, resulting in one connected line. From the resulting characteristic lower and upper contour the *features of the characteristic contour* are extracted.

The last group are the *features of the enclosed regions* which are defined on the closed loops occurring in a handwritten text from which features are obtained. Each loop defines a blob, i.e., a region enclosed by the loop from which the features are extracted.

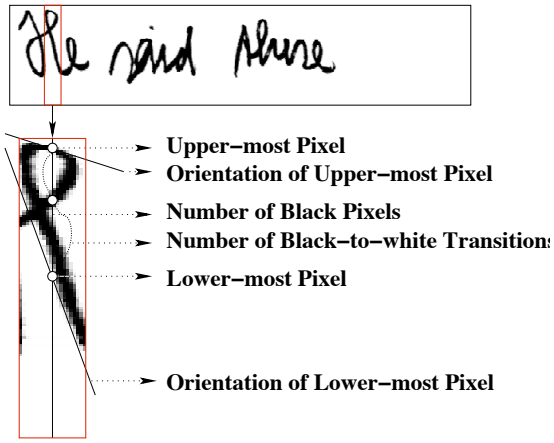

Figure 3: Local Features Extracted by the Sliding Window

In summary, 100 individual features are obtained from one handwritten text line. These features are arranged in a feature vector that serves as an abstract representation of the text line. As the extracted features can have very different numerical ranges, each feature is normalised with respect to its mean and standard deviation.

2.2 Text Recognition

Hidden Markov Models (HMMs) are a powerful statistical tool for the modelling of a sequence of observations. Due to their expressive mathematical structure they have been successfully applied to a wide range of tasks in pattern recognition, i.e., handwriting recognition [16] or speech recognition [19]. For both isolated word and general text recognition, HMMs have become the predominant approach. HMM-based recognisers have a number of advantages over other approaches [3]. Firstly, they are resistant to noise and can cope with shape variations. Secondly, HMM-based recognisers are able to implicitly segment a text line into words and characters, a task that is difficult to perform explicitly [23]. Thirdly, standard algorithms for training and classification exist [19].

The text recognition system uses HMMs to model the handwriting. Only a short description of the system is given here; for a detailed presentation we refer to [17]. Before feature extraction, skew, slant, and baseline position of each text line are normalised. These normalisation steps are necessary to reduce the impact of the different writing styles. After preprocessing, a handwritten text line is transformed into a sequence of feature vectors. For this purpose, a sliding window is used. The window has a width of one pixel and moves from left to right, one pixel per step, over the image. At each position of the window, nine geometrical features are extracted. The first three features are of global nature and describe the distribution of the black pixels in the window. The remaining six local features describe specific points in the window, such as the position of the upper and the lower contour, respectively. See Fig. 3 for an illustration of the local features.

For each upper and lower case character an individual HMM is built. Additionally, frequent punctuation marks, such as full stop, colon and space are modelled. Other, infrequent punctuation marks are mapped to a special garbage HMM. Each character HMM consists of 16 states. The states

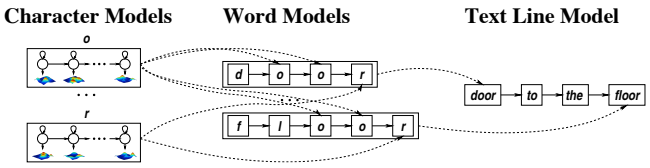

Figure 4: HMM Modelling of a Text Line

are connected in a linear topology, i.e. for each state only two transitions exist: a transition to itself and a transition to the next state. The character models are concatenated to word models which in turn are concatenated to model a complete text line. The HMM modelling of a text line is illustrated in Fig. 4.

The system is trained by applying the Baum-Welch algorithm which iteratively maximises the probability for a given sequence of observations [19]. Recognition is performed by the Viterbi algorithm using dynamic programming to recursively maximises the likelihood of the state sequence [19]. The recognition is supported by a statistical bi-gram language model which defines the probability that a word w_j follows a word w_i [24]. The bi-gram language model is obtained from the LOB corpus [11].

2.3 Labelling

The readability estimation problem can be stated as a classification problem [10]: given an input feature vector X_t extracted from a text t, determine whether X_t belongs to class c_1 or to class c_2. Class c_1 indicates that the text is *readable*, while class c_2 indicates that the text is *unreadable*. Thus

$$X_t \in \begin{cases} c_1, & \text{if} \quad r(X_t) \geq \vartheta \\ c_2, & \text{otherwise} \end{cases} \tag{1}$$

where $r(X_t) \in [0,1]$ denotes the recognition rate on text t achieved by the text recognition system described in Section 2.2. The threshold $\vartheta \in [0,1]$ controls the classification. Clearly, if ϑ is set to a high value, more text is classified as *unreadable*.

2.4 Classifier Training

To address the readability classification problem, two classifiers are evaluated. The first classifier is the k-Nearest-Neighbour (k-NN) classifier. The k-NN classifier determines the k nearest neighbours to each input feature vector and opts for the class that is most often represented. In case of a tie, the class with the smallest sum of distances is chosen. The Euclidean distance measure is used in the experiments described subsequently. The optimal number k of nearest neighbours is determined on a validation set. The advantages of this classifier are its conceptual simplicity and the fact that no classifier training is needed.

The second, more complex classifier is the Support Vector Machine (SVM). The idea of a SVM is to separate two different classes of patterns by a maximum margin hyperplane [4]. The maximum margin hyperplane is the hyperplane for which the distance to the closest pattern of either class is maximal. Generally, two classes are not linearly separable without error because of outliers and noisy objects. In order to handle such errors, so called slack variables ξ are introduced. These slack variables measure the error in terms of distance to the class boundary. In order to control whether the maximization of the margin or the minimization of the error is more important, a weighting parameter C is defined.

If two classes are not separable in the original input space, the so-called *kernel trick* is applied. It maps the data into a high dimensional feature space and constructs a separating hyperplane with maximum margin there. This hyperplane is equivalent to a non-linear decision boundary in the original input space. Using a kernel function, it is possible to compute the separating hyperplane without explicitly carrying out the mapping into the feature space. A kernel function fulfils the condition:

$$\kappa(\mathbf{x},\mathbf{y}) = \langle \sigma(\mathbf{x}), \sigma(\mathbf{y}) \rangle$$

where $\mathbf{x}$ and $\mathbf{y}$ are feature vectors and $\sigma : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is a function with $n, m \in \mathbb{N}$ and $m \geq n$, mapping objects from the input to a higher dimensional feature space. Different kernel functions exist. The *radial basis function (RBF)* kernel function

$$\kappa(\mathbf{x},\mathbf{y}) = \exp(-\gamma \|\mathbf{x}-\mathbf{y}\|^2), \quad \gamma > 0.$$

was chosen in this work because it contains only one meta-parameter. Furthermore, the simpler linear kernel with no meta-parameter is a special case of the RBF kernel [13]. In addition, RBF kernels are most widely used and have been extensively studied [21]. The parameter γ of the kernel function as well as the weighting parameter C need to be optimized on a validation set. The SVM is implemented using the LIBSVM library [6].

2.5 Feature Set Transformation

It is an open question whether the 100 extracted features are optimal or near-optimal for the task under consideration. Actually, some features may not be independent of each other or even be redundant and therefore provide only little information. Moreover, there may be features that do not provide any useful information at all. Subsets of features may exist that perform as well as, or even better than, the original set of features. Furthermore, using a smaller set of features results in a more efficient classifier with respect to both computation times and memory requirements.

Feature extraction is the process of deriving a subset of the original set of features in order to increase classifier efficiency and to allow for higher classification accuracy [14]. There are two different approaches to obtaining a subset of features: *feature selection* and *feature transformation*. In feature selection, a subset of the original set of features is selected while in feature transformation, the features of the original feature set are combined and then projected onto a space of lower dimensionality.

In this paper, the classical PCA method to find an effective transformation is applied. PCA seeks a projection that best represents the data [7]. This method has been chosen because of the fast computation times compared to other feature selection methods.

3. EXPERIMENTAL SETUP

The text lines used in the experiments are part of the IAM handwriting database [17][1].This database currently contains

[1]The IAM handwriting database is publicly available at: http://www.iam.unibe.ch/fki/databases/iam-handwriting-database

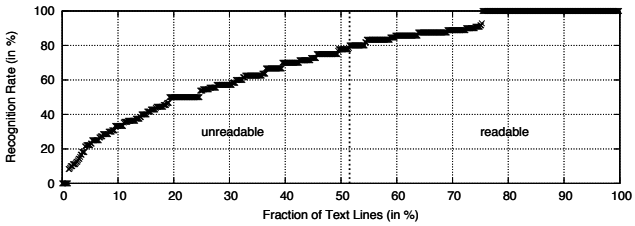

Figure 5: Distribution of the Recognition Rates

over 1,500 pages of handwritten text from over 650 writers. In total, 3,308 text lines from 347 writers are used in the experiments described in this paper. The text lines are divided into two disjoint sets. The first set is used to train and validate the text recognition system. The second set is used to train, validate, and test the readability estimation system.

The first set of 1,478 text lines of 297 writers is split into a training and a validation set. The text recognition system is trained using 1,333 text lines from 268 writers. The meta-parameters are optimized using the remaining 145 text lines from 29 writers. The text lines of each writer appear only in one data set, thus the sets are writer independent.

The second set of 1,830 text line from 50 writers is split into three sets: a training, a validation and a test set. This data set is the same set as the one that was used for writer identification in [8]. A total of 780 text lines from 20 writers are used as training set, 335 text lines from another 10 writers are used as validation set, and the remaining 715 text lines from 20 writers form the test set. Again, this is a writer independent setup.

During training of the handwriting recognition system, the number of Gaussian mixture components of the HMM-based recognizer is increased from 6 to 18 components in steps of 3. Optimal performance on the validation set is achieved using 15 Gaussian mixture components resulting in a word accuracy rate of 69.78%.

Next, the data of the readability estimation system is labelled. The labelling is controlled by the parameter ϑ (see Eq. 1). This parameter was set so as to have approximately the same amount of readable as well as unreadable text lines in the training set. This partitioning prevents the SVM from classifying all data as coming from one class and nevertheless achieve good classification rates. The parameter was set to $\vartheta = 0.8$, which results in 48.46 % of the text lines being readable and 51.54 % unreadable. The distribution of the recognition rates for the text lines of the training set are shown in Fig. 5. The vertical dashed line shows the partitioning of the data into the classes *readable* and *unreadable*.

4. RESULTS AND DISCUSSION

First of all, the meta-parameters of the two classifiers are optimized on the validation set. In case of the k-NN classifier, the parameter k needs to be determined. It is varied from 1 to 780 in steps of 2 (780 denotes the total number of text lines in the training set). The highest classification rate of 72.54 % is achieved with $k = 651$ neighbours. In case of the SVM-classifier, the meta-parameters $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$ need to be optimized. The highest classification rate of 71.04 % is achieved with $C = 2^1$ and $\gamma = 2^{-7}$. In the subsequent experiments, the classifiers are trained using the meta-parameters thus obtained.

Feature Set	k-NN		SVM	
	# Feat.	Cla. Rate	# Feat.	Cla. Rate
Original	100	65.17 %	100	66.15 %
PCA	36	64.20 %	45	67.41 %

Table 1: Classification Rates on the Test Set

Classifier's Decision	Actual Class of the Text Data	
	readable	*unreadable*
readable	13.43 %	*9.65 %*
unreadable	*22.94 %*	53.99 %

Table 2: *Type 1* and *Type 2* Errors

The results on the test set are given in Table 1. In the first row, the classification rates for the original feature set consisting of 100 features are shown. The k-NN based readability estimation system achieves a classification rate of 65.17 %. The SVM-based readability estimation system yields a slightly better classification rate of 66.15 %. However, the classification rate is not significantly higher at the statistical significance level of 99 %.

For the PCA algorithm, the optimal dimension d of the transformed feature subspace is determined as follows. For each dimension $d \in [1, 100]$, the classification rate for a given dimension is calculated on the validation set. The dimension which produces the highest writer identification rate is selected. Using this dimension, the final classification rate on the test set is calculated.

The results of applying PCA to the original feature sets are shown in the second row of Table 1. The k-NN-based system achieves a classification rate of 64.20 % using 36 features. The SVM-based systems returns a classification rate of 67.41 % for 45 features. Neither of the results is statistically significantly different compared to the results obtained by the original feature set of 100 features.

A readability estimation system can make two types of errors. It can falsely label a readable text as *unreadable* (*Type 1* error) or it can label an unreadable text as *readable* (*Type 2* error) [1]. The two types of errors as well as the cases where the system correctly classifies readable text as *readable* and unreadable text as *unreadable*, are shown in Table 2. Table 2 is calculated on the test set consisting of 715 text lines using PCA and the SVM classifier. A correct classification rate of 67.41 % is achieved. The *Type 1* error is 9.65 % and the *Type 2* error equals 22.94 %.

The following conclusions can be drawn from the results presented in Table 1. Firstly, the SVM-based system does not perform significantly better than the simpler k-NN based system. Secondly, the feature sets found by PCA are half or two-thirds times smaller then the original features set and lead to classification rates that are not significantly different from those obtained with the full set of features. Thirdly, in over two-thirds of all cases, the readability of a text line is correctly estimated. This result is very promising.

5. CONCLUSION AND FUTURE WORK

In this paper, a new approach to estimating the readability of a handwritten text is presented. It allows one to filter out

unreadable data prior to recognition and thus helps avoiding needless recognition attempts. The estimation problem is posed as a two class classification problem where a text is classified as either *readable* or as *unreadable*. A classifier is trained on this classification problem. In the training phase, for each text, a vector of features is extracted and the text recognition rate achieved on the text is determined. Next, the feature vectors are labelled and the labelled data is used to train two classifiers, a k-NN and SVM classifier, respectively. Both classifiers show promising results on a test set of 715 text lines from 20 writers.

The readability classification task presented in this paper can be regarded as a special case of the more general problem of estimating whether or not a presented data is recognisable by a given system. The idea is to filter out unrecognisable data before it is passed on to the recognizer. A natural extension of this work would thus be to study the problem of *recognisability classification* on other recognition tasks. Additionally, instead of posing the readability estimation problem as a classification problem, it can be formulated as a regression problem with the aim of predicting the recognition rate without actually performing the recognition. Moreover, all experiments reported in this paper could be repeated with other thresholds (see Eq. 1) than the one used in this paper. These issues are left for future research.

Acknowledgements

This research is supported by the Swiss National Science Foundation NCCR program "Interactive Multimodal Information Management (IM2)" in the Individual Project "Visual/Video Processing".

REFERENCES

[1] F. Bimbot and G. Chollet. Assessment of speaker verification systems. In D. Gibbon amd R. Moore and R. Winski, editors, *Handbook of Standards and Resources for Spoken Language Systems*, pages 408–480. Mouton de Gruyter, 1997.

[2] L. R. Blando, J. Kanai, and T. A. Nartker. Prediction of OCR accuracy using simple image features. In *Proc. 3rd Int. Conf. on Document Analysis and Recognition*, volume 1, pages 319–322, 1995.

[3] H. Bunke. Recognition of cursive Roman handwriting – past, present and future. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, pages 448–461, 2003.

[4] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[5] S.-H. Cha and S. N. Srihari. Apriori algorithm for sub-category classification analysis of handwriting. In *Proc. 6th Int. Conf. on Document Analysis and Recognition*, pages 1022–1025, 2001.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at: *www.csie.ntu.edu.tw/∼cjlin/libsvm*.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, 2001.

[8] C. Hertel and H. Bunke. A set of novel features for writer identification. *Audio- and Video-Based Biometric Person Authentication*, pages 679–687, 2003.

[9] C. J. Hilditch. Linear skeletons from square cupboards. *Machine Intelligence*, 4:403–420, 1969.

[10] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[11] S. Johansson. *The tagged LOB Corpus: User's Manual*. Norwegian Computing Centre for the Humanities, Norway, 1986.

[12] J. Kanai, T. A. Nartker, S. V. Rice, and G. Nagy. Performance metrics for document understanding systems. In *Proc. 2nd Int. Conf. on Document Analysis and Recognition*, pages 424–427, 1993.

[13] S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

[14] M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33:25–41, 2000.

[15] E. D. Mandana, N. Sherkat1, and T. Allen. Handwriting style classification. *Int. Journal on Document Analysis and Recognition*, 6(1):55–74, 2003.

[16] U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 15:65–90, 2001.

[17] U.-V. Marti and H. Bunke. The IAM–database: An English sentence database for off-line handwriting recognition. *Int. Journal of Document Analysis and Recognition*, 5:39–46, 2002.

[18] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[19] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–285, 1989.

[20] A. Schlapbach and H. Bunke. Off-line writer identification and verification using Gaussian mixture models. *Machine Learning in Document Analysis and Recognition*, 11:409–428, 2008.

[21] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[22] P. Soille. *Morphological Image Analysis*. Springer, 1999.

[23] T. Steinherz, E. Rivlin, and N. Intrator. Off-line cursive script word recognition – a survey. *Int. Journal of Document Analysis and Recognition*, 2:90–110, 1999.

[24] M. Zimmermann and H. Bunke. N-gram language models for offine handwritten text recognition. In *Proc. 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 203–208, 2004.