Estimating 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

This paper presents a new approach to estimate the readability of a handwritten text. The estimation problem is posed as a regression problem where a Support Regression system is used to estimate the recognition rate of a text. It allows one to filter out unreadable data prior to recognition and thus helps avoiding needless recognition attempts. The estimated recognition rate can then be used to classify a text as either readable *or as* unreadable. *To find an optimal classification threshold for a given task, a cost function which allows one to define different costs for the error of classifying an unreadable text as* readable *and the error classifying readable text as* unreadable *is defined. The system is systematically evaluated on a data set of 1,830 text lines from 50 writers and shows very promising performance.*

Keywords: Readability Estimation, Support Vector Regression, Handwriting Recognition

1. Introduction

Recognizing unreadable text is hard and chances of false recognition are high. Filtering out unreadable text prior to recognition would help avoiding needless recognition attempts and reduce costs associated with false classification.

In previous work, we have cast the readability estimation problem as a classification problem [19]. A text is classified as either *readable* or *unreadable*. In the training phase, each text is transformed into a feature vector and the text recognition rate 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.

In this paper, instead of training a classifier with priorly labelled data we directly use the recognition rates to train a regression system. The goal of regression is to find a function that has a small deviation from the actually achieved recognition rates for all the training data and is as flat as possible [21]. In the recognition phase, the function is then used to estimate the recognition rate of a text without recognition.

Only very few works on readability estimation exists. The problem of predicting the accuracy of an Optical Character Recognition (OCR) has been studied in [1, 10]. Related to this work are writer identification (for an overview of recent work see [18]), handwriting style classification [12], and sub-category classification analysis of handwriting [4]. 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.

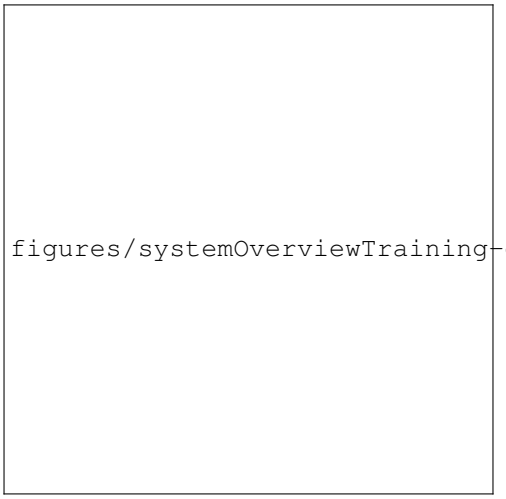

Figure 1. Training of the Readability Estimation System Using Regression

The rest of this paper is structured as follows. The next section presents the regression-based readability estimation system. Section 3 describes the data and the experimental setup. The regression results are presented in Section 4. The predicted recognition rates are then used to classify a text as either *readable* or *unreadable*. The results are presented in Section 5 and discussed in Section 6. Finally, Section 7 concludes the paper and proposes future work.

2 System Overview

This section describes the regression-based 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. The feature vector and the recognition rate are then merged and used to train a Support Vector regression system. A systematic

overview of the training procedure is shown in Fig. 1. 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 regression system which then estimates the recognition rate.

The rest of this section is structured as follows. Feature extraction and text recognition are presented in Subsections 2.1 and 2.2, respectively. Support Vector regression is described in Subsection 2.3.

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 [6]. These features are able to distinguish different writers. Therefore, it is reasonable to apply them to the readability estimation task as well.

Before feature extraction, a text line image is normalised. The normalisation operations are designed to improve the quality of the features extracted without removing writer specific information. Only a short description of the normalisation operations is given here; for more details see [6]. The grey scale text line images are binarised using Otsu's binarisation algorithm [16], clipped and then thinned by Hilditch's thinning algorithm which iteratively refines a text line image until a stable image is achieved [7].

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 [6]. The first group of *basic features* describe basic properties of a text line, such as the skew or the slant angle.

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.

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 from which features are extracted.

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.

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

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 [13] or speech recognition [17]. 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 [17].

The text recognition system described in the rest of this section uses *Hidden Markov Models (HMMs)* to model handwriting. Only a short description of the system is given here; for a detailed presentation we refer to [14]. 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.

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 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 system is trained by applying the Baum-Welch algorithm which iteratively maximises the probability for a given sequence of observations [17]. Recognition is performed by the Viterbi algorithm using dynamic programming to recursively maximise the likelihood of the state sequence [17]. 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 [9].

2.3 Support Vector Regression

Support Vector Regression (SVR) is used to estimate the recognition rate from the extracted feature set of a text line. Given training data $\{(x_1, y_1), \ldots, (x_l, y_l)\} \in \mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ denotes the space of the input patterns (e.g. $\mathcal{X} = \mathbb{R}^d$) the goal is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible[1]. In other words, we do not care about errors as long as they are less than ε, but will not accept any larger deviation. We start with a linear functions f, taking the form

$$f(x) = \langle w, x \rangle + b \quad \text{with} \quad w \in \mathcal{X}, b \in \mathbb{R} \qquad (1)$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathcal{X}$. *Flatness* in the case of Eq. (1) means that one seeks a small w. One way to ensure this is to minimize the norm, i.e., $\|w\|^2 = \langle w, w \rangle$. This problem can be stated as a convex optimization problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b & \leq \quad \varepsilon \\ \langle w, x_i \rangle + b - y_i & \leq \quad \varepsilon. \end{cases} \end{aligned} \qquad (2)$$

The assumption in Eq. (2) is that such a function f actually exists that approximates all pairs (x_i, y_i) with ε precision. However, this may not always be the case. Therefore slack variables ξ_i, ξ_i^* are introduced to cope with otherwise infeasible constraints of the optimization problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b & \leq \quad \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq \quad \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq \quad 0. \end{cases} \end{aligned} \qquad (3)$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated. Fig. 2 depicts the situation graphically. Only the points outside the shaded region contribute to the cost insofar, as the deviations are penalized in a linear fashion.

Next, one can show that w can be completely described as linear combination of the training patterns x_i [2]. The key observation is that the SV algorithm only depends on dot products between patterns x_i. Hence it suffices to know $\kappa(x, x') = \langle \Phi(x), \Phi(x') \rangle$ rather than Φ explicitly.

This observation is now used to extend the SV algorithm to the non-linear domain by mapping the training

[1]The outline in this section follows closely the presentation of SVR in [21].

Figure 2. Soft Margin Loss Setting for a Linear SVM

patterns $x_i \in \mathcal{X}$ into some feature space $\mathcal{F}$ using a function $\Phi : \mathcal{X} \to \mathcal{F}$ and allows us to restate the SV optimization problem:

$$\begin{aligned} &\text{maximize} && \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i + \alpha_i^*)(\alpha_j + \alpha_j^*)\kappa(x_i, x_j) \\ -\epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) \end{cases} \\ &\text{subject to} && \sum_{i=1}^{l}(\alpha_i + \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C] \end{aligned} \qquad (4)$$

where α_i and α_i^* are Lagrange multipliers [21]. The difference to the linear case in Eq. 3 is that w is no longer given explicitly. Furthermore, in the non-linear setting the optimization problem corresponds to finding the *flattest* function in *feature* space, not in input space.

Next we need to determine functions $\kappa(x, x')$ which correspond to a dot product in some feature space. A function $\kappa(x, x)$ is a valid kernel function if it fulfils the *Mercer* condition [15]. While different kernel functions exist, the *radial basis function (RBF)* kernel function

$$\kappa(x, x') = \exp(-\gamma\|x - x'\|^2), \quad \gamma > 0.$$

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, i.e., the linear kernel with a penalty parameter C has the same performance as the RBF kernel with some parameters (C, γ) [11]. In addition, RBF kernels are most widely used and have been extensively studied [20]. 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 [5].

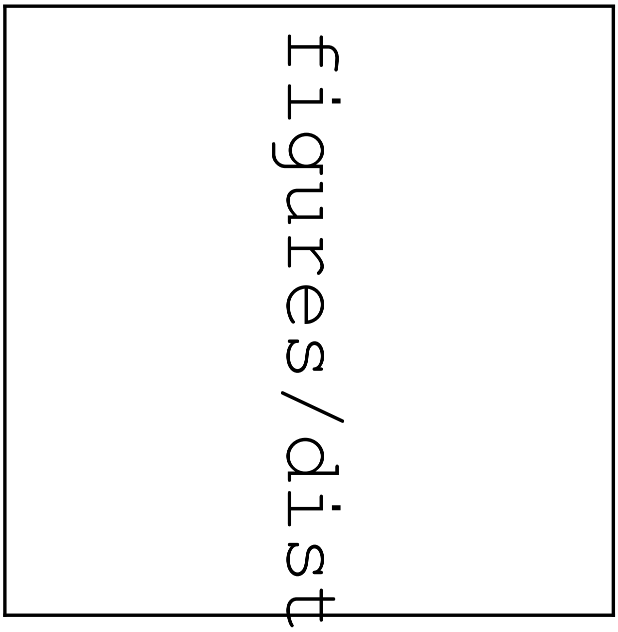

Figure 3. Distribution of the Recognition Rates on the Training Set

3 Experimental Setup

The text lines used in the experiments are part of the IAM handwriting database [14][2]. The database currently contains over 1,500 pages of handwritten text from over 650 writers. In total, 3,308 text lines from 347 text 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 regression-based 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 [6]. 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%. The distribution of the recognition rates for the text lines of the training set are shown in Fig. 3.

[2]The IAM handwriting database is publicly available at: www.iam.unibe.ch/fki/iamDB

4 Regression Results

For a given text line t, the SVR system outputs am estimated recognition rate $r_{\text{est}}(t)$. Two measures are defined to assess the performance of the SVR system. Firstly, the sum of the absolute difference between the recognition rate $r(t)$ and the estimated recognition rate $r_{\text{est}}(t)$ over all text lines divided by the number n of text lines is calculated:

$$\text{mean absolute error} = \frac{1}{n}\sum_{i=1}^{n} |r(t_i) - r_{\text{est}}(t_i)|$$

Secondly, the median absolute error is calculated where N denotes the total number of text lines:

$$\text{median absolute error} = \text{median}_{i \in N}\, |r(t_i) - r_{\text{est}}(t_i)|$$

Next, the meta-parameters of the SVR system are optimized. The meta parameters $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\vartheta \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$ are optimized on the validation set. The lowest mean absolute error of 20.29% is achieved by $C = 2^{11}$ and $\vartheta = 2^{-13}$. This result means, that in average, the recognition rate achieved by the text recognition system and the estimated recognition rate yielded by the SVR system vary by approximately 20%. The median absolute error is 17.83% and thus lower than the mean absolute error. This finding can be explained by the fact that only 32 text lines with $|r(t_i) - r_{\text{est}}(t_i)| > 40\%$ exist.

In Fig. 4, the absolute error $e_{\text{abs}}(t) = |r(t) - r_{\text{est}}(t)|$ for each text line t of the validation set is plotted in ascending order. As can be seen from Fig. 4, the absolute error is below 10% in over 25% and below 15% in over 42% of all cases.

5 Classification Results

In a real world scenario an interesting question is to decide whether or not a given text is readable. An example of such a scenario 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. 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.

To filter out unreadable text, the readability estimation problem is transformed into a classification problem [8]: 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 *unreadable*, while class

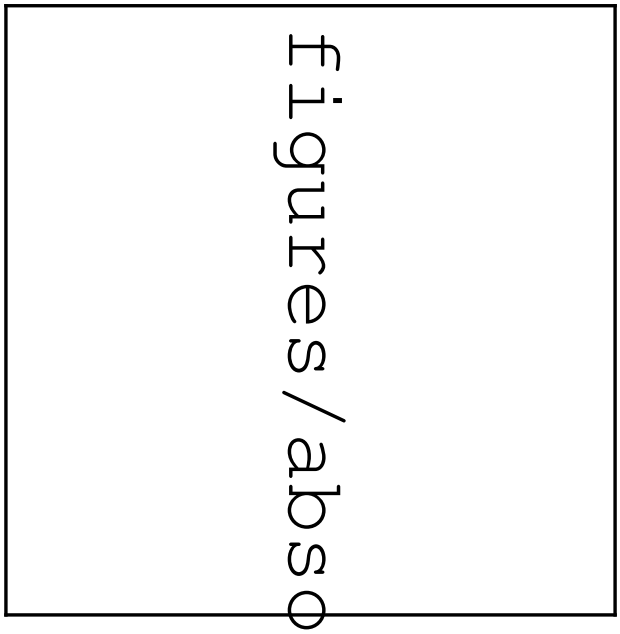

Figure 4. Absolute Error Rates on the Validation Set

c_2 indicates that the text is *readable*. Thus

$$X_t \in \begin{cases} c_1, & \text{if } r(X_t) \geq \vartheta \\ c_2, & \text{otherwise} \end{cases} \quad (5)$$

where $r(X_t) \in [0, 100]$ denotes the recognition rate (in percent) on text t achieved by the text recognition system described in Section 2.2. The threshold $\vartheta \in [0, 100]$ controls the classification. Clearly, if ϑ is set to a high value, more text is classified as *unreadable*.

Four possible cases are defined for the readability classification task. The first two cases occur when the system correctly labels a readable text as *readable* (denoted *Correct Readable Classification (CRC)*) or when the system correctly labels an unreadable text as *unreadable* (denoted *Correct Unreadable Classification (CUC)*). In the other two cases the system makes a mistake. Firstly, the system can falsely label a readable text as *unreadable* (denoted *False Unreadable Classification (FUC)*) and, secondly, it can falsely label an unreadable text as *readable* (denoted *False Readable Classification (FRC)*). The two types of errors are quantified as *Type I Error* $= \frac{FRC}{FRC+CFC}$ and as *Type II Error* $= \frac{FUC}{FUC+CRC}$. The correct classification rate is defined by $\frac{CRC+CUC}{CRC+CUC+FUC+FRC}$.

The output of the readability classification depends on the cost function f_{cost}. The rationale of the cost function presented in this paper is to minimize both the number of *FRC* relative to the total number of readable text (i.e., $RC = CRC + FRC$) as well as the number of *FUC* relative to the total number of unreadable text (i.e., $UC = CUC + FUC$). The cost function thus is defined as

$$f_{\text{cost}}(c, \vartheta) = c * \frac{FRC(\vartheta)}{RC(\vartheta)} + (1 - c) * \frac{FUC(\vartheta)}{UC(\vartheta)} \quad (6)$$

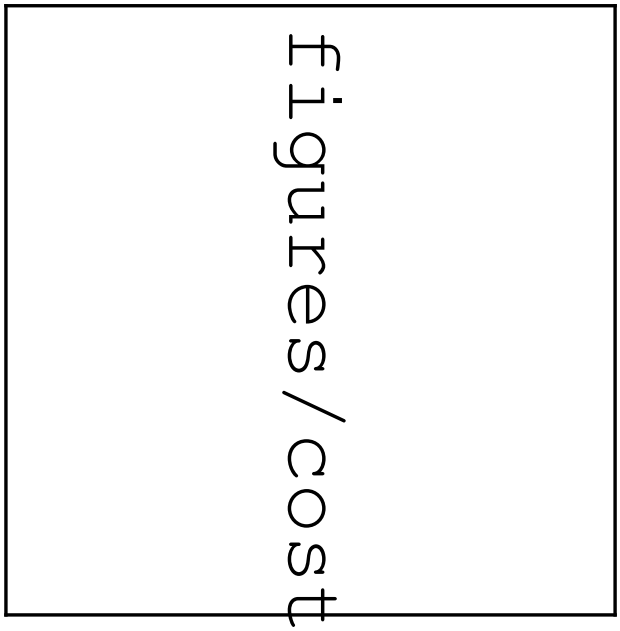

Figure 5. Costs for the Two Different Scenarios on the Validation Set

where the parameter $c \in [0, 1]$ denotes the cost of wrongly classifying an unreadable text as readable and, consequently, $(1 - c) \in [0, 1]$ denotes the cost of wrongly classifying a readable text as unreadable. The parameter $\vartheta \in [0, 100]$ denotes the recognition rate above which a text is classified as *readable* (see Eq. 5).

Varying the cost $c \in [0, 1]$ enables us to investigate scenarios with different costs for the two types of errors. The following two scenarios are investigated in this work. The first scenario characterizes the case where wrongly classifying an unreadable text as *readable* and wrongly classifying a readable text as *unreadable* produce the same cost (i.e., $c = \frac{1}{2}$). The second scenario reflects a scenario where an automatic reading system wrongly classifies an unreadable address as *readable* and the letter or the parcel is then delivered to a wrong address and is either lost or returned and then has to be reprocessed. In this case we assume that wrongly classifying an unreadable text as *readable* is ten times more costly than wrongly classifying a readable text as *unreadable* (i.e., $c = \frac{10}{11}$).

In Fig. 5 shows the f_{cost} for the two scenarios by systematically varying threshold $\vartheta \in [0, 100]$ on the validation set. For the first scenario with $c = \frac{1}{2}$ minimal costs of 0.278 are obtained at $\vartheta = 45\%$. This means that the costs are minimal if a text is For the second scenario with $c = \frac{10}{11}$ minimal costs of 0.075 are obtained at $\vartheta = 37\%$.

Tables 1 and 2 show the classification results for the two scenarios. The scenario with cost $c = \frac{1}{2}$ classifies 80.00% of the text lines as *readable* and 20.00% as *unreadable*. The *Type I Error* is 48.45% and the *Type II Error* is 7.14%. A correct classification rate of 80.90% is achieved.

Table 1. Results for Scenario with Cost $c = \frac{1}{2}$

Classifier's Decision	Actual Class of the Text	
	readable	*unreadable*
readable	221	47
unreadable	17	50

Table 2. Results for Scenario with Cost $c = \frac{10}{11}$

Classifier's Decision	Actual Class of the Text	
	readable	*unreadable*
readable	246	45
unreadable	7	37

The scenario with cost $c = \frac{10}{11}$ classifies 75.52% of the text lines as *readable* and 24.48% as *unreadable*. The *Type I Error* is 54.88% and the *Type II Error* is 2.77%. A correct classification rate of 84.48% is achieved.

6 Discussion

In preliminary experiments to classify the regression results as either *readable* or *unreadable*, the cost function was defined as

$$f'_{\text{cost}}(\vartheta) = \frac{CRC(\vartheta) + CUC(\vartheta)}{RC(\vartheta) + UC(\vartheta)} \tag{7}$$

i.e., divides the correct classified text lines by all text lines. However, the cost function f'_{cost} produces the best results for $\vartheta = 0\%$ meaning that all text lines are labelled as *unreadable*. Under this cost function the classification task degenerates to an one class classification problem and f_{cost} was used instead.

Interestingly using the cost function f_{cost} of Eq. 6, the lowest costs are produced if the readability threshold ϑ is set to a value below $\vartheta = 50\%$, i.e., to $\vartheta = 45\%$ and to $\vartheta = 37\%$, respectively. This result show that optimal results are achieved with low thresholds which contradicts the intuition that a high threshold produces best results.

7 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 regression problem where a Support Regression system is used to estimate the recognition rate of a text. The regression system is tested on a test set of 715 text lines from 20 writers. In over half of all cases, the absolute error between the actual and the estimated recognition rates is below 21%.

The estimated recognition rate can then be used to classify a text as either *readable* or as *unreadable* using a threshold ϑ. To find an optimal threshold for a given task, a cost function which allows to define different costs for the error of classifying an unreadable text as *readable* and the error classifying readable text as *unreadable* are defined. Two scenarios with different costs are studied and show correct classification rates above 80% on the test set.

The readability classification task studied 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. A natural extension of this work would thus be to study the problem of *recognisability classification* on other recognition tasks. This issue is 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] 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.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *Proc. of the Annual Conf. on Computational Learning Theory*, pages 144–152, 1992.

[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] 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.

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

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

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

[8] 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.

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

[10] 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.

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

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

[13] 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.

[14] 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.

[15] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions Royal Society, London*, A 209:415–446, 1909.

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

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

[18] 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.

[19] A. Schlapbach, F. Wettstein, and H. Bunke. Automatic estimation of the readability of handwritten text. Submitted for publication.

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

[21] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 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.