

Combining On-Line and Off-Line Bidirectional Long Short-Term Memory Networks for Handwritten Text Line Recognition

Marcus Liwicki^{1,2} and Horst Bunke¹

¹Institute of Computer Science and Applied Mathematics
University of Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland
{liwicki, bunke}@iam.unibe.ch

²German Research Center for AI (DFKI GmbH)
Knowledge Management Department, Kaiserslautern, Germany
{firstname.lastname}@dfki.de

Abstract

In this paper we present a multiple classifier system (MCS) for on-line handwriting recognition. The MCS combines several individual recognition systems based on bidirectional long short-term memory networks. To obtain diverse recognizers, we use different feature sets based on on-line and off-line features. Furthermore, we generate a number of different recognizers by changing the initialization of the networks. To combine the word sequences output by the recognizers, we incrementally align these sequences using the recognizer output voting error reduction framework (ROVER). For deriving the final decision, different voting strategies are applied. The best combination ensemble has a recognition rate of 83.64%, which is significantly higher than the 81.26% achieved by the best individual classifier.

1. Introduction

Although the problem of handwriting recognition has been considered for more than 40 years [2, 15, 19], there are still many open issues, especially in the task of unconstrained handwritten sentence recognition. In this paper we consider the recognition of text written on a whiteboard, which is a relatively new task. Recently, a novel recognition strategy based on bidirectional long short-term memory networks (BLSTM) has been presented for this problem [13]. This approach uses connectionist time classification (CTC) to allow the underlying neural networks to be trained and tested on unsegmented data.

Having the on-line and the off-line recognition systems described in [13] and [12] available, it may be beneficial to combine both systems. From such a combination, an improved recognition performance can be expected [18, 20]. A general overview and an introduction to the field of MCS is given in [9]. Several combination methods for character, numeral and word recognition have been proposed in the literature [7, 20, 23]. However, the

combination of the outputs of multiple handwritten text line recognizers differs from standard multiple classifier combination, because the output of a text line recognizer is a sequence of word classes rather than just one single word and the number of words may differ between several recognizers. Therefore the recognizer output voting error reduction (ROVER) [4] framework is applied in this paper. To combine the output of the recognizers under this framework, the word sequences are incrementally aligned using a string matching algorithm. Then a voting strategy is applied to get the final result. A similar MCS approach based on hidden Markov models (HMMs) has been introduced recently [11]. While the approach in [11] combines only three recognizers, using just the number of occurrences in its voting strategy, more recognizers are combined in this paper and advanced voting strategies are proposed.

The rest of the paper is organized as follows. A description of all recognition systems for the combination experiments is given in Section 2, and the methodology for combining the different recognizers is introduced in Section 3. Section 4 presents the experiments and results, and finally, Section 5 draws some conclusions and gives an outlook to future work.

2 Recognition Systems

Thirty individual recognition systems are used for the combination described in this paper. They are based on three different feature sets. A total of ten BLSTM classifiers have been trained on each feature set.

2.1 Preprocessing and Feature Extraction

The first set of features uses off-line images generated from the on-line data [12]. The text line to be recognized is normalized with respect to skew, slant, writing width and baseline location. Normalization of the baseline location means that the body of the text line (the part which is located between the upper and lower baselines), the as-

center part (located above the upper baseline), and the descender part (below the lower baseline) will be vertically scaled to a predefined size each. Writing width normalization is performed by a horizontal scaling operation, and its purpose is to scale the characters so that they have a predefined average width. To extract the feature vectors from the normalized images, a sliding window is used. The width of the window is one pixel, and nine geometrical features are computed at each window position. Thus an input text line is converted into a sequence of feature vectors in a 9-dimensional feature space. The following features are computed: the average gray value of the pixels in the window (1) and their center of gravity(2), the second order moment in vertical direction (3), the positions (4,5) and gradients (6,7) of the uppermost and lowermost black pixel, the number of black-white transitions between the uppermost and lowermost pixel in an image column (8), and the proportion of black pixels to the number of pixels between these two points (9).

To get the second set of features, the on-line data are used directly. The individual text lines are corrected with respect to their skew. For this purpose we perform a linear regression through all points. For slant normalization, we compute the histogram over all angles between the lines connecting two successive points of the trajectory and the horizontal line [8]. After normalization, delayed strokes, e.g. the crossing of a “t” or the dot of an “i” are removed, using simple heuristics. Next, we perform an equidistant resampling of the point sequence, i.e. the original sequence of points is replaced by a sequence of points on the trajectory where all consecutive points have the same distance to each other. The optimal value for the distance has been empirically found. This step is needed because different writers write at a different speed. The next important step is the computation of the baseline and the corpus line. For this purpose, we compute two linear regressions through the minima and maxima of the y -coordinates of the strokes and remove the least fitting points. This correction step is done twice which results in the estimated baseline (minima) and corpus line (maxima). The baseline is subtracted from all y -coordinates to make them equal to the x -axis. This step also gives us the three main writing areas. As the last preprocessing step, the width of the characters is normalized. The extracted features are the following: the pen-up/pen-down feature (1); the hat-feature, which indicates if a delayed stroke has been removed at the same horizontal position (2); the speed (3); the normalized x - and y -coordinate (4,5); the writing direction (6,7); the curvature (8,9); the vicinity aspect (10); the vicinity slope (11,12); the vicinity curliness (13); the vicinity linearity (14); the ascenders and descenders in the off-line vicinity of the considered point (15,16); and the context map, where the two-dimensional vicinity of the point is transformed to a 3×3 map and the resulting nine values are taken as features (17-25).

The third set of features is based on the same feature extraction steps as used for the second feature set, up to two differences. Before applying the normalization oper-

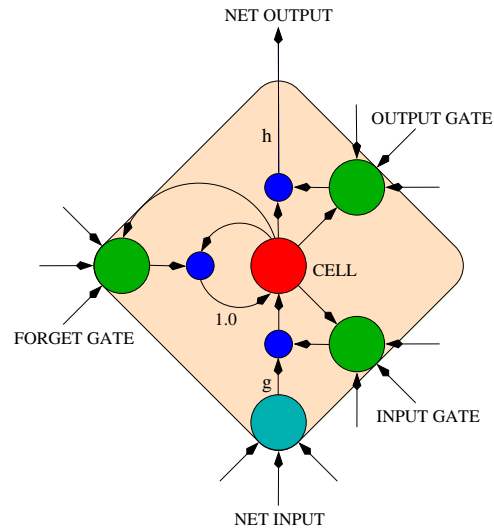


Figure 1. LSTM memory block with one cell

ations, an additional step is introduced, which is motivated by the nature of the whiteboard data. The text lines on a whiteboard usually have no uniform skew along the whole line and the slant and size of the letters is not the same at the beginning and at the end of a line. This is caused by the fact that people stand, rather than sit, during writing and the arm does not rest on a table. Therefore the text line is split into subparts and the preprocessing is done for each subpart separately. Splitting is done at gaps that are larger than the mean gap size. Also the size of both subparts that result from splitting has to be greater than a predefined threshold. Slant correction is done for each subpart individually. It is supplemented with the following steps. We weight the angle histogram values with a Gaussian with its mean at the vertical angle, and the variance empirically set. This is beneficial because some words are not properly corrected if a single long straight line is drawn in horizontal direction, which results in a large histogram value. We also smooth each histogram entry with its direct neighbors using the window (0.25, 0.5, 0.25), because in some cases the correct slant is at the border of two angle intervals and a single peak at another interval may be slightly higher. This single peak will become smaller after smoothing.

2.2 BLSTM combined with CTC

Recurrent neural networks (RNNs) [16] are a natural choice for on-line handwriting recognition, since they are able to access extensive contextual information when transcribing characters or words. Until recently however, RNNs were limited to making separate classifications at every timestep in an input sequence. This severely limited their applicability to domains such as speech recognition and cursive handwriting recognition, since it required that the training data be pre-segmented, and that the network outputs be post-processed to give the final transcriptions.

Connectionist temporal classification (CTC) [3] is an RNN objective function designed to overcome the above mentioned problem. It uses the network to define a prob-

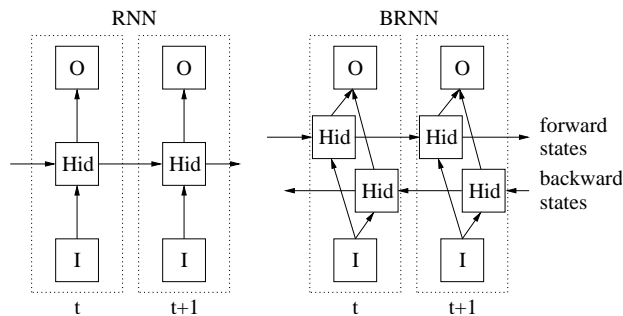


Figure 2. RNN and BRNN (unfolded in time)

ability distribution over a fixed set of labels plus an additional ‘blank’, or ‘no label’ unit. It then interprets the sequence of network outputs as a probability distribution over all possible transcriptions for a given input sequence, and trains the network by maximizing the log probabilities of the correct transcriptions on the training set.

As already mentioned, the advantage of RNNs derives from their ability to access contextual information. It is therefore important to choose an RNN architecture that maximizes the amount of context available.

Long short-term memory (LSTM) [6] is an RNN architecture specifically designed to bridge long time delays between relevant input and target events, making it very suitable for handwriting recognition, where long range context is required to disambiguate individual labels. An LSTM hidden layer consists of multiple recurrently connected subnets, known as memory blocks. Each block contains a set of internal units, or cells, whose activation is controlled by three multiplicative units: the input gate, forget gate and output gate. Figure 1 provides a detailed illustration of an LSTM memory block with a single cell. The cell has a recurrent connection with fixed weight 1.0. The three gates collect input from the rest of the network, and control the cell via multiplicative units (small circles). The activations of the input and output gates scale the input and output of the cell while that of the forget gate scales the recurrent connection of the cell. The cell input and output squashing functions (g and h) are applied at the indicated places.

Whereas standard RNNs make use of previous context only, bidirectional RNNs (BRNNs) [17] are able to incorporate context on both sides of every position in the input sequence. This is useful in handwriting recognition, since it is often necessary to look to both the right and left of a given letter in order to identify it. Figure 2 shows a comparison of a standard RNN and a BRNN in two time states. In the BRNN there exist two hidden layers, i.e. one layer for each direction, and the input/output neurons (marked with I/O) are connected to both.

Combining the above two concepts gives bidirectional LSTM (BLSTM). The recognition architecture used in this paper is based on BLSTM networks combined with CTC. CTC allows the network to be trained with unsegmented data. The intuition is that, because we don’t know where the labels within a particular transcription will oc-

In mid-april Anglesey

W_1 : In mid-april Angle say
 W_2 : It mid-april Anglesey
 W_3 : I a mid-April Anglesey

$$WTN_1 = W_1 + W_2 :$$

In	mid-april	Angle	say
It	mid-april	Anglesey	ϵ

$$WTN_2 = WTN_1 + W_3 :$$

In	ϵ	mid-april	Angle	say
It	ϵ	mid-april	Anglesey	ϵ
I	a	mid-April	Anglesey	ϵ

Figure 3. Example of iteratively aligning multiple recognition results

cur, we sum over all the places where they could occur. In general, a large number of paths will correspond to the same label sequence, so a naive calculation of the conditional probability is unfeasible. However, it can be efficiently evaluated using a graph-based algorithm, similar to the forward-backward algorithm for HMMs, which is therefore called CTC forward-backward algorithm [3].

The number of hidden cells is fixed to 100. The only parameter that is optimized on the validation set is the number of training iterations. To obtain different classifiers for each feature set, the BLSTMs have been initialized randomly using a different random seed for each initialization.

3 Combination Methodology

Since the combination of the outputs of multiple handwritten text line recognizers differs from standard multiple classifier combination, the ROVER combination strategy has been chosen. This strategy consists of two stages. Because the recognizers output word sequences corresponding to whole text lines and because there may be a different number of words in each sequence, the output sequences are aligned into a *word transition network* (WTN) first. A voting strategy is then applied to select the best scoring word at each location for the final transcription.

Finding the optimal alignment for n sequences is NP-complete [22]. Thus an approximate solution for the alignment is chosen. This solution aligns the multiple sequences incrementally by building WTNs. At the beginning, the first and the second word sequence are aligned in a single WTN, using the string matching algorithm described in [21]. The resulting WTN is aligned with the next word sequence giving a new WTN, which is then aligned with the next recognition result, and so on. This method does not guarantee an optimal solution, but in practice the suboptimal solutions often provide an alignment of sufficiently high accuracy. An example alignment

of the output of three recognizers (denoted by W_1, W_2 , and W_3) is shown in Fig. 3. The columns in the WTNs denote correspondence sets (sets of corresponding words) that are identified by the alignment process. Note that the symbol ϵ marks *null*-transitions, i.e., transitions where an empty string is chosen as an alternative.

After alignment a voting module extracts the best scoring word sequence from the WTN. One possibility is to take only the number of occurrences of a word w for making a decision. In case of ties, the output of the best performing system on the validation set is taken. In the example of Fig. 3 it would be the “In” from W_1 which yields the final output “In mid-april Anglesey”. Note that this is the correct transcription, which is not present in the recognition results of any single recognizer.

In addition to the frequency of occurrence, the confidence of the recognizers can be taken into account. The trade off between the frequency of occurrence and the confidence score is weighted with a parameter α . Let c be a correspondence set of the WTN containing n word classes w_1, \dots, w_n . The number of occurrences of each word class w_i is denoted by N_i . Then the score $S(w_i)$ of a word w_i is:

$$S(w_i) = \alpha * \frac{N_i}{\sum_{j=1}^n N_j} + (1 - \alpha) * C(w_i), \quad (1)$$

where $C(w_i)$ is the combined confidence score of word class w_i . There exist several methods to calculate $C(w_i)$. First, the average of all confidence scores of the occurrences of word class w_i can be used. Second, the maximum of these confidence scores may be chosen. Setting $\alpha = 1$ results in a voting that takes only the number of occurrences into account.

Another parameter needed in the combination is the confidence score of the *null*-transition $C(\epsilon)$. It determines how often the *null*-transition is taken instead of another word class. If no *null*-transition is taken, i.e., $C(\epsilon)$ is very low, the output transcription tends to be longer and more insertion errors occur. On the contrary, if $C(\epsilon)$ is too large, more deletions occur. The two parameters α and $C(\epsilon)$ are optimized on a validation set in this paper.

4 Experiments and Results

The experiments have been performed on the IAM-OnDB [10], a large on-line handwriting database acquired from a whiteboard. It consists of 13,040 written lines, containing 86,272 instances of 11,050 distinct words. For the alignment and construction of the WTN, the Rover tool from the National Institute of Standards and Technology (NIST) implementing the ROVER framework [4] has been used¹.

The IAM-OnDB-t2 benchmark task has been used for the experiments. The benchmark data is divided into four sets, one set for training the models, two sets for validating meta parameters, and one independent test set for

¹The Rover tool is part of the NIST Scoring Toolkit available for download at ftp://jaguar.nsl.nist.gov/current_docs/sctk/doc/sctk.htm

Table 1. Recognizers used for the combination experiments

System		Best accuracy (average)
CTC	off-line (1)	76.43 % (73.6 %)
	on-line (2)	81.26 % (79.8 %)
	on-line (3)	81.07 % (80.1 %)

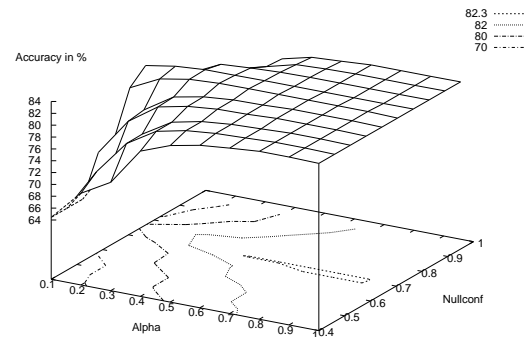


Figure 4. Optimization of the parameters α and $C(\epsilon)$ (Nullconf)

measuring the final recognition results. The vocabulary contains the 20,000 most frequent words out of three corpora (LOB Corpus, Brown Corpus, Wellington Corpus). The parameters for the recognition and the combination have been optimized on the first validation set. A bigram language model is included in the experiments. This language model has been optimized on the second validation set. The language model training is based on three different corpora (LOB Corpus, Brown Corpus, Wellington Corpus). To guarantee that the test data is not used for training, the texts of the IAM-OnDB have been removed from the LOB Corpus. The performance of the classifiers is measured on the word level accuracy in all our experiments.

A summary of the performance of the individual recognition systems on the test set is given in Table 1. Altogether 30 classifiers are involved in the experiments, i.e. 10 classifiers for each feature set. The different feature sets are denoted by “off-line (1)” for off-line features, “on-line (2)” for on-line features, and “on-line (3)” for on-line features extracted after the refined preprocessing. The accuracy of the best performing classifier for each feature set is given in the right column. Additionally, the average accuracy of the ten classifiers is given in brackets. The best neural network classifier has been trained without the refined preprocessing (“on-line (2)”). It performs at 81.26 %, which is 0.19 % higher than the best CTC with refined preprocessing (“on-line (3)”). However, this maximum performance is perhaps an outlier, since the average accuracy of the “on-line (3)” CTC is 80.1 %, compared to 79.8 % of the “on-line (2)” CTC. The off-line CTC systems perform with an average accuracy of 73.6 %.

Several parameters of the MCS have to be optimized during validation. First, there are the parameters for vot-

Table 2. Results of the selected combinations on the validation set

System	Accuracy
Best individual classifier	74.63 %
Three classifiers	77.03 %
Optimized weighting	77.20 %
Optimized mix	80.40 %

ing described in Section 3. Then, the order of the classifiers for the alignment has to be chosen. Finally, the number of classifiers to be included in the ensemble must be decided. In the following these issues will be described in greater detail.

The parameters for voting, α and $C(\epsilon)$, are optimized on the validation set. Before voting can be applied, the outputs of the recognizers need to represent a confidence measure, $C(w_i)$ see (1). The performance on the validation set has been used as confidence score $C(w_i)$ for each of the classifiers. The final voting strategy consists in taking the maximum rather than the average of the individual confidence scores in (1), because it returned better recognition results in all experiments on the validation set. Figure 4 illustrates the accuracy on the validation set for different parameter combinations for an example classifier ensemble. It is observed that the values do not change significantly for $\alpha > 0.8$. The highest accuracy in this example is reached at ($\alpha = 0.5$, $C(\epsilon) = 0.7$).

The order of the classifiers has an influence on the alignment, for only a suboptimal alignment strategy is chosen. A commonly used strategy begins with the best classifier on the validation set and sequentially adds the next best classifier. This strategy has been adopted in this paper.

To get the number of individual classifiers to be included in the ensemble, we investigated several strategies. First, only the best classifier for each feature set has been selected, resulting in an ensemble of three classifiers (second row in Table 2). This ensemble has a higher recognition accuracy on the validation set than the best single classifier (first row in Table 2). Second, the weighing of these classifiers has been optimized using weights from 0 to 1 with a step size of 0.1. The best weights are 0.6 for “off-line (1)”, 0.5 for “on-line (2)”, and 0.5 for “on-line (3)”. However, there is only a little performance increase on the validation set (third row in Table 2). Finally, the number n_c ($c = 1, 2, 3$) of individual classifiers from each feature set has been varied from one to ten, e.g. if $n_1 = 3$, the three best “off-line (1)” systems are used. This results in 10^3 combinations that have been tested on the validation set. The results on the validation set show a promising increase of the recognition accuracy (last row in Table 2). The optimized ensemble of classifiers uses six “off-line (1)” systems, one “on-line (2)” system, and six “on-line (3)” systems.

The final results on the test set appear in Table 3. The optimized combination performs statistically significantly better than the best individual classifier at a significance level of 1 %. The recognition accuracy is 83.64 %, corresponding to error reduction of more than 12 %.

Table 3. Results on the test set

System	Accuracy
Best individual classifier	81.26 %
Three classifiers	81.27 %
Optimized weighting	81.18 %
Optimized mix	83.64 %

Table 3 also shows the performance of the combination of the best individual classifiers for each feature set (second row). Surprisingly, the recognition accuracy drops from 81.27 % to 81.18 % if the optimal weights found on the validation set are used for weighting the classifiers. This confirms the observation during validation that the weighting does not lead to a large difference in performance. Note that the performance on the test set is higher than on the validation set. This has already been observed in previous works [11]. The main reason for this observation seems to be that the tests set contains data that is easier to recognize.

5 Conclusions

In this paper a multiple classifier system (MCS) for the recognition of handwritten notes written on a white-board has been presented. While several MCS for character, numeral and word recognition have been proposed, only few methods exist for the combination of complete text lines, because the output of a text line recognizer is a sequence of word classes rather than just one single word and the number of words may differ between several recognizers. Therefore the ROVER framework has been used to combine the output sequences of the recognizers. This framework first aligns the word sequences incrementally in a word transition network, and then, at each output position, applies a voting strategy to select the final result.

The experiments have been performed on the IAM-OnDB-t2 benchmark. Several BLSTM recognizers have been combined. For the experiments a language model is included, which has been trained on the second validation set, while the first validation set has been used to optimize the individual systems and the combination parameters. The experimental results on the test set show a highly significant improvement of the recognition performance. The final recognition accuracy is 83.64 %, corresponding to an error reduction of more than 12 % compared to the best individual classifier.

An interesting outcome of our experiments is that applying an MCS using only three classifiers did not increase the performance significantly, even if the weighting is optimized. It can be concluded that simply applying an MCS is not helpful in all cases. However, using different recognizers by changing the initialization of the network leads to better recognition rates.

While we already achieved a performance increase, there are still possibilities to further improve the system in future work. This can be done by using more elaborated confidence scores. For example, the word-level confidences of the individual classifiers or other confidence measures proposed in the literature [1, 5, 14] could be taken.

Acknowledgments

This work was supported by the Swiss National Science Foundation program “Interactive Multimodal Information Management (IM)²” in the Individual Project “Visual/Video Processing”, as part of NCCR. The authors thank Roman Bertolami for providing tools for the combination. Furthermore, thanks are due to the IDSIA in Switzerland for providing the neural network toolkit.

References

- [1] R. Bertolami, M. Zimmermann, and H. Bunke. Rejection strategies for offline handwritten text line recognition. *Pattern Recognition Letters*, 27(16):2005–2012, 2006.
- [2] H. Bunke. Recognition of cursive Roman handwriting – past present and future. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 448–459, 2003.
- [3] S. Fernández, A. Graves, and J. Schmidhuber. Sequence labelling in structured domains with hierarchical recurrent neural networks. In *Proc. 20th Int. Joint Conf. on Artificial Intelligence*, pages 774–779, 2007.
- [4] J. Fiscus. A post-processing system to yield reduced word error rates: recognizer output voting error reduction ROVER. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 347–352, 1997.
- [5] N. Gorski. Optimizing error-reject trade off in recognition systems. In *Proc. 4th Int. Conf. on Document Analysis and Recognition*, volume 2, pages 1092–1096, 1997.
- [6] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5–6):602–610, 2005.
- [7] Y. S. Huang and C. Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(1):90–94, 1995.
- [8] S. Jäger, S. Manke, J. Reichert, and A. Waibel. On-line handwriting recognition: the NPen++ recognizer. *Int. Journal on Document Analysis and Recognition*, 3(3):169–180, 2001.
- [9] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons Inc, 2004.
- [10] M. Liwicki and H. Bunke. IAM-OnDB – an on-line English sentence database acquired from handwritten text on a whiteboard. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, volume 2, pages 956–961, 2005.
- [11] M. Liwicki and H. Bunke. Combining on-line and off-line systems for handwriting recognition. In *Proc. 9th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 372–376, 2007.
- [12] M. Liwicki and H. Bunke. Handwriting recognition of whiteboard notes – studying the influence of training set size and type. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 21(1):83–98, 2007.
- [13] M. Liwicki, A. Graves, H. Bunke, and J. Schmidhuber. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *Proc. 9th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 367–371, 2007.
- [14] J. Pitrelli and M. P. Perrone. Confidence-scoring post-processing for off-line handwritten-character recognition verification. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 278–282, 2003.
- [15] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.
- [16] R. Rojas. *Neural Networks – A Systematic Introduction*. Springer-Verlag, 1996.
- [17] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681, November 1997.
- [18] O. Velek, S. Jäger, and M. Nakagawa. Accumulated-recognition-rate normalization for combining multiple on/off-line Japanese character classifiers tested on a large database. In *Proc. 4th Workshop on Multiple Classifier Systems*, pages 196–205, 2003.
- [19] A. Vinciarelli. A survey on off-line cursive script recognition. *Pattern Recognition*, 35(7):1433–1446, 2002.
- [20] A. Vinciarelli and M. Perrone. Combining online and offline handwriting recognition. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, pages 844–848, 2003.
- [21] R. Wagner and M. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21:168–173, 1974.
- [22] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.
- [23] X. Ye, M. Cheriet, and C. Y. Suen. StrCombo: combination of string recognizers. *Pattern Recognition Letters*, 23:381–394, 2002.