Self-training for Handwritten Text Line Recognition

Volkmar Frinken and Horst Bunke

Institute for Computer Science and Applied Mathematics
University of Bern, Switzerland
{frinken,bunke}@iam.unibe.ch

Abstract. Off-line handwriting recognition deals with the task of automatically recognizing handwritten text from images, for example from scanned sheets of paper. Due to the tremendous variations of writing styles encountered between different individuals, this is a very challenging task. Traditionally, a recognition system is trained by using a large corpus of handwritten text that has to be transcribed manually. This, however, is a laborious and costly process. Recent developments have proposed semi-supervised learning, which reduces the need for manually transcribed text by adding large amounts of handwritten text without transcription to the training set. The current paper is the first one, to the knowledge of the authors, where semi-supervised learning for unconstrained handwritten text line recognition is proposed. We demonstrate the applicability of self-training, a form of semi-supervised learning, to neural network based handwriting recognition. Through a set of experiments we show that text without transcription can successfully be used to significantly increase the performance of a handwriting recognition system.

1 Introduction

Off-line handwriting recognition (HWR) is the task of recognizing a handwritten text from a sheet of paper that was scanned, photographed or digitized otherwise. Opposed to on-line handwriting recognition where temporal and spatial information about each stroke is available, the off-line recognition task is performed using only the image of the written text. Many important applications are based on off-line handwriting recognition, e.g. postal address identification [3], Bank check processing [15], prescreening of handwritten notes [20], or the creation of digital libraries of historical documents [8]. After several decades of ongoing research, however, off-line handwritten text recognition is still considered a difficult problem that is only partially solved [4,17].

To create an automatic handwriting recognition system, a set of images of handwritten text along with their correct transcription is needed for training. As it turns out, one of the key problems encountered when building a writer independent recognition system[1] is the great variety in writing styles between

[1] A writer independent system is one that recognizes text from writers that have not contributed to the training set.

I. Bloch and R.M. Cesar, Jr. (Eds.): CIARP 2010, LNCS 6419, pp. 104–112, 2010.

different persons. Hence, the amount of training data needed is extremely large. Unfortunately, the transcription of the handwritten text has to be done manually which makes the acquisition of training data costly and time consuming. On the other hand, collecting handwritten samples itself can be done very efficiently. Consequently, unlabeled data can be made easily available in large amounts. Hence the question arises whether such unlabeled data can be helpful for handwriting recognition systems. It has been shown that in various classification scenarios unlabeled examples can indeed significantly improve the recognition accuracy using semi-supervised learning [5]. Most of the existing works, however, deal with the standard classification scenario where a single point in a feature space has to be mapped into the label space [16,21]. In the current paper, a more general problem is considered in the sense that a (possibly long) sequence of feature vectors has to be mapped to a (usually much shorter) sequence of labels, or characters. Some research has been done on sequential semi-supervised learning, mostly with Hidden Markov Models, but only moderate success has been achieved following this approach [11,12]. Only few publications exist that deal specifically with semi-supervised learning for handwritten word recognition. In [1,2] the authors adapt a recognition system to a single person by using unlabeled data. This system is highly specialized after the adaptation and not suitable for general handwriting recognition, though. The task of unconstrained writer independent single word recognition was addressed in [6,7]. In this paper, we extend this approach by not restricting the focus on single, manually segmented words, but considering entire text lines.

The semi-supervised learning approach addressed in this paper is self-training. Under this paradigm, one starts with an initial recognizer trained on the available labeled data. This recognizer then classifies all unlabeled data and sorts them according to their recognition confidence. The most confidently recognized samples are assumed to be correct and added to the training set. Using the augmented training set, a new recognizer is created. This procedure of enlarging the training set can be continued for several iterations. A crucial point in this process, however, is to decide which elements should be added and which not. If, on the one hand, the data is selected too strictly, not enough samples might be added to change the training set substantially. On the other hand, if large amounts of incorrectly labeled data are added, the recognition accuracy of the created systems might decrease. For the task of text line recognition, we investigate the use of different confidence thresholds and their effect on self-learning. To the knowledge of the authors, this paper is the first to deal with semi-supervised learning for text-line recognition.

The rest of the paper is structured as follows. In Section 2, details of the task of handwritten word recognition are presented together with the recognizer used in this work. Semi-supervised learning and self-training are discussed in Section 3. The experiments are presented in Section 4 and the paper concludes with Section 5.

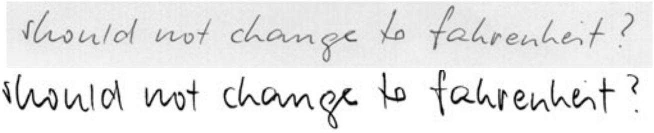


Fig. 1. A visualization of the effects of the preprocessing steps

2 Handwritten Text Line Recognition

2.1 Preprocessing

The database used in this paper consists of 1,539 pages of handwritten English text, written by 657 writers[2] [14]. All pages of the database are already segmented into individual text lines. The segmented text lines are normalized prior to recognition in order to cope with different writing styles. First, the skew angle is determined by a regression analysis based on the bottom-most black pixel of each pixel column. Then, the skew of the text line is removed by rotation. Afterwards the slant is corrected in order to normalize the directions of long vertical strokes found in characters like 't' or 'l'. After estimating the slant angle based on a histogram analysis, a shear transformation is applied to the image. Next, a vertical scaling is applied to obtain three writing zones of the same height, i.e., lower, middle, and upper zone, separated by the lower and upper baseline. To determine the lower baseline, the regression result from skew correction is used, and the upper baseline is found by vertical histogram analysis. Finally the width of the text is normalized. For this purpose, the average distance of black/white transitions along a horizontal straight line through the middle zone is determined and adjusted by horizontal scaling. The result of the preprocessing steps can be seen in Fig. 1. For more details on the text line normalization operations, we refer to [13].

2.2 The HWR System

The recognizer used in this paper is a recently developed recurrent neural network, termed *bidirectional long short-term memory* (BLSTM) neural network [10]. A hidden layer is made up of so called *long short-term memory* blocks instead of simple nodes. These memory blocks are specifically designed to address the *vanishing gradient problem* which describes the exponential increase or decay of values as they cycle through recurrent network layers. This is done by nodes that control the information flow in and out of each memory block.

The network is made up of two separate input layers, two separate recurrent hidden layers, and one output layer. Each input layer is connected to one hidden layer. The hidden layers are both connected to the output layer. The network is *bidirectional*, i.e. a sequence is fed into the network in both the forward and the backward mode. The input layers consist of one node for each feature. One

[2] http://www.iam.unibe.ch/fki/databases/iam-handwriting-database

input and one hidden layer deal with the forward sequence, the other input and hidden layer with the backward sequence. At each position p of the input sequence of length l, the output layer sums up the values coming from the hidden layer that has processed positions 1 to p and the hidden layer that has processed the positions l down to p. The output layer contains one node for each possible character in the sequence plus a special ε node, to indicate “no character”. At each position, the output activations of the nodes are normalized so that they sum up to 1, and are treated as probabilities that the node’s corresponding character can occur at this position.

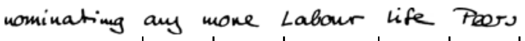
The output of the network is therefore a matrix of probabilities for each letter and each position. A likelihood is assigned to each path through the matrix by multiplying all probability values along the path. The letters visited along the optimal path, i.e. the one with maximum likelihood, give the recognized letter sequence. Note, however, that the optimal path may not correspond to any existing word. Given a dictionary of all recognizable words, the Connectionist Temporal Classification (CTC) token passing algorithm returns a sequence of words from the dictionary whose likelihood is (locally) optimal. This sequence is the final output of the recognizer. For more details about BLSTM networks and the CTC token passing algorithm, we refer to [9,10].

3 Self-training

3.1 Overview

The way semi-supervised learning is applied in this paper is self-training. It is a methodology widely applicable due to its general and abstract formulation. It states that all available labeled data should be utilized to train an initial recognizer in a classic supervised manner. This recognizer is then used to classify the unlabeled data. Each classification result is stored along with its recognition confidence. The most confidently recognized elements are considered as correctly recognized and hence added to the already existing training set which is then used to create a new recognizer. These steps are repeated until some criterion is met, e.g. the convergence of the recognition accuracy.

Applying this scheme, a decision has to be made at each iteration which elements are to be added to the training set and which not. This can be done by comparing the recognition confidence to some threshold. Using a very strict threshold ensures that as few as possible falsely recognized elements are added to the new training set, but also that not many elements are added at all. This may lead to a training set that is not substantially different from the original training set and produces recognizers that are not much different from the initially created one. When the threshold is too loose, more data is added to the training set at the cost of misclassified data. Therefore a trade-off has to be found between data quality and quantity.


nominating any more Labour life Peers

dominating	any	more	Labour	the	Few
nominative	a	none	Labour	the	Peru
nominal	in c	more	Labour	the	Dr
dominating	any	more	Labour	with	Peru
mining	any	one	labour	the	Ten
dominating	any	more	labour	like	e
dominating	any	more	labour	the	Dr
nomination	any	one	Labour	the	Kerr
morning	any	more	Labour	He	or
noting	any	more	Labour	the	Few

dominating	any	more	Labour	the	Few
$\frac{4}{10}$	$\frac{8}{10}$	$\frac{7}{10}$	$\frac{7}{10}$	$\frac{7}{10}$	$\frac{2}{10}$

Fig. 2. A text line, the aligned outputs of 10 neural networks, and the resulting recognition confidences. Note that the recognition of the best network is used as the final transcription (given in the first line of the table).

3.2 Recognition Confidence

In order for the entire process to work, a reliable measure of the recognition confidence is crucial since it specifies what elements are used to train a new recognizer. A simple approach in the case of text recognition using BLSTM neural networks would be to use the likelihood of the path though the letter probability matrix. However, preliminary experiments have shown, this method does not serve well as an overall realiability measure of the output.

Therefore we exploit the fact that the BLSTM neural network is initialized with random weights and create an ensemble of several neural networks by random initialization. Clearly, each network of this ensemble produces a different recognition result for a given input text in general. Therefore, we count how often a word occurs in the set of produced transcriptions, and use this count as a confidence measure. Of course, the individual neural networks may output word sequences of different lengths. Consequently, the word sequences of all outputs have to be aligned with each other. Since finding an optimal alignment is NP-complete [19] we use an approximation algorithm that works well for the considered application.

To create the alignment, the networks are ranked according to their performance on the validation set first. Then, the best network's output is fixed and the other word sequences are sequentially aligned to it. By this procedure, an adequate recognition with a reliable confidence measure for each word is efficiently generated. An example of the procedure can be seen in Fig. 2.

In Fig. 3, a sample plot can be seen that shows the number of words being correctly as well as incorrectly recognized as a function of the recognition confidence. This function can be evaluated on the validation set and used to define the threshold that is applied when deciding what elements are used for retraining. For an optimal increase in the recognizer's accuracy, the threshold should

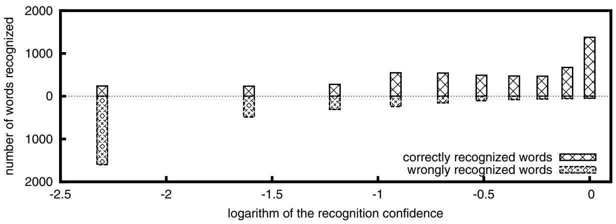


Fig. 3. The number of correctly and incorrectly recognized words as a function of the logarithm of the recognition confidence. Since the recognized confidence is calculated from the numbers of networks agreeing with the transcription, the confidence takes on only a few distinct values.

maximize the number of correct elements that are added, while at the same time minimizing the number of wrong elements. Obviously, this is not an easy task. In this paper, we investigate three different threshold selection methods.

3.3 Confidence Thresholds

The first threshold is called *High Threshold* and is set to 0 (See Fig. 3). It means that a word is added to the retraining set only if all networks agree on the output. A second, more refined threshold is the *Medium Threshold*. It is set so that all elements added are more likely to be correct than wrong. This threshold is found by choosing the lowest value returning more correctly that incorrectly recognized samples in the validation set. (In Fig. 3, a possible value is *Medium Threshold* $= -1$). The last threshold investigated is the *Low Threshold*. It is set to $-\infty$ so that all words, regardless of their recognition confidence, are added to the training set.

4 Experimental Evaluation

To analyze the effects of self-training on the recognition accuracy, we performed experiments on the IAM database [14] using the thresholds described in 3.3. The database is split up into a working set of 6161 text lines, a validation set of 920 text lines and a writer independent test set of 929 text lines. These three sets are writer disjoined, i.e. a person who has contributed to any of the three sets did not contribute to any of the other sets. Therefore, we test the applicability of the method for writer independent recognition instead of adapting to a specific style of writing as done in [1,2]. The working set is randomly split up into a training set consisting of 1000 labeled text lines and a set of 5161 unlabeled text lines.

Initially, we trained 10 neural networks on the training set. The networks then transcribed the unlabeled text lines and the transcriptions were sorted according to the networks' performance on the validation set. In the next step, the transcriptions were aligned and used to compute a new confidence measure for each word. Then we applied the confidence threshold and added the appropriate

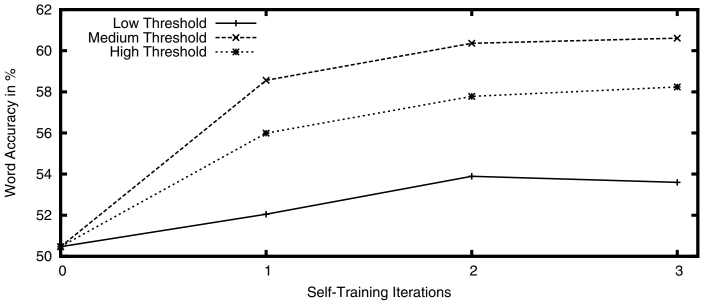


Fig. 4. The average recognition accuracy of 10 neural network based recognition systems on the test set as a function of the self-training iterations

words to the initial training set. The initial neural networks were then retrained on the extended training set. We repeated this process for several iterations. Due to the high computational costs, we did not wait for the accuracy to converge but instead fixed the number of iterations.

For recognition of text lines, the accuracy is defined by $Acc = \frac{n-S-D-I}{n}$ where n is the number of words in the ground truth, S is the number of misrecognized words (substitutions), D is the number of words that don't appear in the recognized text (deletions) and I is the number of words that appear in the recognized text but not in the ground truth (insertions) [18].

We evaluated the word accuracy of the initial system and the system obtained after each iteration. The results can be seen in Fig. 4. The initial system has an average accuracy of 50.47%. During the course of the iterations, an increase in accuracy can clearly be observed. Using the *Low Threshold* retraining rule, an increase up to 53.89% on average was observed. A larger increase achieved the *High Threshold* retraining rule of 58.24% on average. The largest increase, however, was achieved using the *Medium Threshold* retraining rule. Using this rule, the average accuracy of the system after three iteration was at 60.61%. Note that each single increase in each iteration is statistically significant at the $\alpha = 95\%$ level. As a comparison and to evaluate how good such a system can get, we additionally trained ten neural networks on the entire working set of 6161 labeled text lines and reached an accuracy of 71%.

The *High Threshold* retraining rule added mostly correct elements to the training set, but only a few words from a text line and not even from each of the lines of the unlabeled set was chosen. Hence, there are not many samples that are added and the training set changes only slightly. The *Low Threshold* rule ensures that each word from each text line is added. This produces enough correct data as not to impede the recognition accuracy, but the large amount of noise also hinders a larger increase. A good trade-off appears to be the *Medium Threshold*, which picks words from each text line that are likely to be correct. Therefore the *Medium Threshold* rule adds more data than the *High Threshold* rule with less noise than the *Low Threshold* retraining rule.

5 Conclusion

We have presented strategies for self-training in the field of handwritten text line recognition. These strategies decrease the need of manually labeled training data and reduce the cost of building recognition systems. Furthermore, by focusing on text line recognition as opposed to single word recognition, not even a segmentation into distinct words is necessary. To the knowledge of the authors, this is the first time that such an approach has been proposed. In a set of experiments we demonstrated the applicability of the procedure and compared the proposed thresholding strategies among each other. In each experiment we evaluated the average recognition accuracy of different recognition systems after each self-training iteration. The highest increase in recognition accuracy, from about 50% to 60%, was observed when making a compromise between including too much noisy data and excluding too much correctly labeled data. Although the increase is remarkable, it still leaves room for further improvements which will be investigated in the future. Next steps include investigations into larger ensembles of diverse recognizers such as HMMs and single word recognizers after an automatic segmentation of the text line into words. Co-training with different recognizers will also be along our line of research.

While previous approaches to using semi-supervised learning in handwriting recognition were restricted to either single writer scenarios or to single word recognition, the system described in this paper is quite general. It can successfully deal with multiple writers and complete text lines. Consequently, it can potentially be applied to many real world tasks, such as transcription of handwritten notes or historical manuscripts.

Acknowledgments

We thank Alex Graves for kindly providing us with the BLSTM Neural Network source code.

References

1. Ball, G.R., Srihari, S.: Prototype Integration in Off-Line Handwriting Recognition Adaptation. In: Proc. Int'l. Conf. on Frontiers in Handwriting Recognition, pp. 529–534 (2008)
2. Ball, G.R., Srihari, S.N.: Semi-supervised Learning for Handwriting Recognition. In: 10th Int'l Conf. on Document Analysis and Recognition (2009)
3. Brakensiek, A., Rigoll, G.: Handwritten Address Recognition Using Hidden Markov Models. In: Dengel, A.R., Junker, M., Weisbecker, A. (eds.) RL 2004. LNCS, vol. 2956, pp. 103–122. Springer, Heidelberg (2004)
4. Bunke, H.: Recognition of Cursive Roman Handwriting - Past, Present and Future. In: Proc. 7th Int'l Conf. on Document Analysis and Recognition, vol. 1, pp. 448–459 (August 2003)
5. Chapelle, O., Schölkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge (2006)

6. Frinken, V., Bunke, H.: Evaluating Retraining Rules for Semi-Supervised Learning in Neural Network Based Cursive Word Recognition. In: 10th Int'l Conf. on Document Analysis and Recognition, pp. 31–35 (2009)
7. Frinken, V., Bunke, H.: Self-Training Strategies for Handwritten Word Recognition. In: Perner, P. (ed.) ICDM 2009. LNCS, vol. 5633, pp. 291–300. Springer, Heidelberg (2009)
8. Govindaraju, V., Xue, H.: Fast Handwriting Recognition for Indexing Historical Documents. In: First Int'l Workshop on Document Image Analysis for Libraries, pp. 314–320. IEEE Computer Society, Los Alamitos (2004)
9. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist Temporal Classification: Labelling Unsegmented Sequential Data with Recurrent Neural Networks. In: 23rd Int'l Conf. on Machine Learning, pp. 369–376 (2006)
10. Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A Novel Connectionist System for Unconstrained Handwriting Recognition. IEEE Transaction on Pattern Analysis and Machine Intelligence 31(5), 855–868 (2009)
11. Inoue, M., Ueda, N.: Exploitation of Unlabeled Sequences in Hidden Markov Models. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(12), 1570–1581 (2003)
12. Ji, S., Watson, L.T., Carin, L.: Semisupervised Learning of Hidden Markov Models via a Homotopy Method. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2), 275–287 (2009)
13. Marti, U.V., Bunke, H.: Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System. Int'l Journal of Pattern Recognition and Artificial Intelligence 15, 65–90 (2001)
14. Marti, U.V., Bunke, H.: The IAM-Database: An English Sentence Database for Offline Handwriting Recognition. Int'l Journal on Document Analysis and Recognition 5, 39–46 (2002)
15. Palacios, R., Gupta, A., Wang, P.S.: Handwritten Bank Check Recognition of Courtesy Amounts. Int'l Journal of Image and Graphics 4(2), 1–20 (2004)
16. Seeger, M.: Learning with Labeled and Unlabeled Data. Tech. rep., University of Edinburgh, 5 Forest Hill, Edinburgh, EH1 2QL (2002)
17. Vinciarelli, A.: A Survey On Off-Line Cursive Word Recognition. Pattern Recognition 35(7), 1433–1446 (2002)
18. Vinciarelli, A., Bengio, S., Bunke, H.: Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models. IEEE Trans. on Pattern Analysis and Machine Intelligence 26(6), 709–720 (2004)
19. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. Journal of Computational Biology 1(4), 337–348 (1994)
20. Ye, M., Viola, P.A., Raghupathy, S., Sutanto, H., Li, C.: Learning to Group Text Lines and Regions in Freeform Handwritten Notes. In: Ninth Int'l Conf. on Document Analysis and Recognition, pp. 28–32. IEEE Computer Society, Los Alamitos (2007)
21. Zhu, X.: Semi-Supervised Learning Literature Survey. Tech. Rep. 1530, Computer Science, University of Wisconsin-Madison (2005), http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf