

Keyword Spotting for Self-Training of BLSTM NN based Handwritten Recognition Systems

Volkmar Frinken^a, Andreas Fischer^b, Markus Baumgartner^c, Horst Bunke^c

^a*Computer Vision Center, Autonomous University of Barcelona, Edifici O, 08193
Bellaterra, Barcelona, Spain*

^b*Centre for Pattern Recognition and Machine Intelligence
Concordia University*

1455 de Maisonneuve Blvd West, Montreal, Quebec H3G 1M8, Canada

^c*Institute of Computer Science and Applied Mathematics, University of Bern,
Neubrückstrasse 10, CH-3012 Bern, Switzerland*

Abstract

The automatic transcription of unconstrained continuous handwritten text requires well trained recognition systems. The semi-supervised paradigm introduces the concept of not only using labeled data but also unlabeled data in the learning process. Unlabeled data can be gathered at little or not cost. Hence it has the potential to reduce the need for labeling training data, a tedious and costly process. Given a weak initial recognizer trained on labeled data, self-training can be used to recognize unlabeled data and add words that were recognized with high confidence to the training set for re-training. This process is not trivial and requires great care as far as selecting the elements that are to be added to the training set is concerned. In this paper, we propose to use a BLSTM NN handwritten recognition system for keyword spotting in order to select new elements. A set of experiments demonstrate the applicability of self-training on modern and historic data and the benefits of using keyword spotting over previously published self-training schemes.

Keywords:

Document Retrieval, Keyword Spotting, Handwriting Recognition, Neural Networks, Semi-supervised Learning

1. Introduction

The automatic transcription of handwritten text, such as letters, manuscripts, or books has received substantial attention over the last decades [1, 2]. Still, the recognition of unconstrained text cannot be considered a solved problem.

Email addresses: vfrinken@uab.cvc.es (Volkmar Frinken),
an_fisch@encs.concordia.ca (Andreas Fischer), markus.baumgartner@students.unibe.ch
(Markus Baumgartner), bunke@iam.unibe.ch (Horst Bunke)

Most state-of-the-art handwriting recognition systems follow the learning-based approach to pattern recognition, i.e., given a large set of input-output pairs, the underlying mapping is learned automatically. To cope with the high variety encountered in the different styles of handwritten text, a large number of such training data is required to build a robust recognizer.

Nowadays even extremely well trained systems are unlikely to produce a perfect transcription. Yet, the output can still be useful as a starting point for manual post-processing and error correction, which reduces the overall work done by humans [3]. Generally it can be said, the better a system is supposed to be, the more training data is needed. Training data for handwriting recognition systems, i.e., images of text lines labeled with their machine-readable transcription, however, has to be created manually in a tedious and costly process.

The research described in this paper is about reducing the amount of human work that is put into generating training sets by making use of unlabeled data which is usually abundant and can be gathered at little or no cost. This is done by exploiting the well known fact that such unlabeled data contain valuable information as well. Approaches to learning that make use of both types of data, labeled and unlabeled, are called semi-supervised learning.

Semi-supervised learning has received remarkable attention in the last few years. 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 [4, 5]. 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, i.e., words or characters. Previous research on sequential data has been mostly done with moderate success using Hidden Markov Models [6, 7].

For the specific task of semi-supervised learning for handwriting recognition, only few publications exist. In [8, 9], the authors adapt a recognition system to a single person by using unlabeled data. In contrast, we consider in this paper the more general task of writer independent handwriting recognition.

A promising approach to semi-supervised learning for writer independent handwriting recognition is self-training [10]. Under this paradigm one starts with an initial system trained on the available labeled data. This system is then used to select words from the set of unlabeled data that have been recognized with high confidence. The most confidently recognized samples are assumed to be correct and added to the training set. Using the augmented training set, a new system 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.

Self-training for handwriting text line recognition has been proposed in [11]. The authors use a complete text transcription system that recognizes every line of the unlabeled data using a large language model. Words recognized with a

sufficient confidence are used for retraining. This paper is an extension of a previously published version [12] in which we proposed to select the elements used for retraining via keyword spotting. It has been shown that keyword spotting reduces the computational costs dramatically compared to text line transcription. Secondly, keyword spotting does not need any language information. Hence, this approach is specifically suited in cases where little or no language information is available.

The extension of this paper over [12] is twofold. Firstly, we give a more detailed explanation and introduction into the field of semi-supervised learning. Secondly, we test the proposed approach on a new database of historic text. This is a particularly interesting application, since general text recognition systems for historic data do not exist but often have to be created individually for each manuscript with huge costs. Hence, semi-supervised learning without the need for any language information would help tremendously.

The rest of the paper is structured as follows. The next section provides information about the task of handwriting recognition, including preprocessing and a description of the recognition system. An introduction to semi-supervised learning in general and self-training in particular together with related work are reviewed in Section 3. In Section 4, keyword spotting and approaches to using a recognition system for keyword spotting are introduced. Details about proposed approach are given in Section 5. Section 6 presents an experimental evaluation and conclusions are drawn in Section 7.

2. Handwriting Recognition

2.1. Preprocessing

To focus on the recognition task, we omit the description of all processing steps up to text line extraction. For details see [13]. Once extracted, the text lines are normalized 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, 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. For more details on the text line normalization operations, we refer to [14]. 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.

A normalized text line image is represented by a sequence of N feature vectors x_1, \dots, x_N with $x_i \in \mathbb{R}^n$. This sequence is extracted by a sliding window

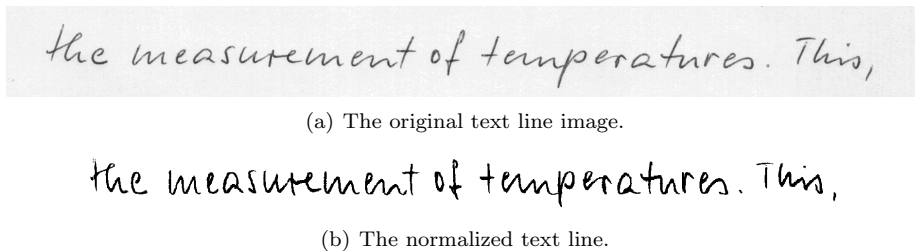


Figure 1: A visualization of the text line preprocessing.

moving from left to right over the image. At each of the N positions of the sliding window, n features are extracted. The sliding window has a width of one pixel. It is moved in steps of one pixel, i.e., N equals the width of the text line. From each window $n = 9$ geometric features are extracted, three global and six local ones. The global features are the 0th, 1st and 2nd moment of the black pixels' distribution within the window. The local features are the position of the top-most and that of the bottom-most black pixel, the inclination of the top and bottom contour of the word at the actual window position, the number of vertical black/white transitions, and the average gray scale value between the top-most and bottom-most black pixel. To compute the inclination of the top and bottom contour, the sliding window to the left of the actual one is considered. For further details on the feature extraction step, we refer to [14].

2.2. BLSTM Neural Networks

The considered keyword spotting system is based on a recently developed recurrent neural network, termed *bidirectional long-short term memory* (BLSTM) neural network [15]. Instead of simple nodes, the hidden layers are made up of so-called *long short-term memory* (LSTM) blocks. 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 into and out of each memory block. The input layer contains one node for each of the nine geometrical features, the hidden layer consists of the LSTM cells, and the output layer contains one node for each possible character plus a special ε node, to indicate “no character”.

The network is *bidirectional*, which means that the sequence of feature vectors is fed into the network both ways, forward and backward. This is because the form of a handwritten character does not only depend upon the previous but also upon the following character. The *bidirectional* architecture is realized by two input and two hidden layers. One input and one hidden layer deal with the forward sequence, and the other input and hidden layer with the backward sequence. The output layer sums up the activation levels from both hidden layers at each position in the text line. The output activations of the nodes in the

output layer are then normalized to sum up to 1. Hence they can be treated as a vector indicating the probability for each letter to occur at a particular 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 [16, 15].

3. Semi-supervised Learning and Self-Training

Supervised learning algorithms use data elements and their class labels to construct the class regions in the input space. In semi-supervised learning, unlabeled data, elements without a class label, are given as well. Hence an algorithm can try to extract useful information from unlabeled data, which is their distribution. One of the most common semi-supervised learning assumptions is therefore the *semi-supervised smoothness assumption*, which states that, if two data points are close to each other and lie in a region with a high density of input data, then they should belong to the same class [10].

A motivation of using unlabeled data comes from the fact that unlabeled data is often easy and cheap to acquire while labeled data is sparse and expensive. In handwritten text recognition, unlabeled data are images of written text, which are ubiquitously available. Labeled data, in contrast, are text lines together with a correct transcription.

3.1. Related Work

This section gives a brief overview of related work in the area of semi-supervised learning. More extensive reviews can be found in [4, 5, 10]. From a chronological point of view, the first methods that uses unlabeled data have been proposed at least five decades ago [17, 18, 19] using the *Expectation-Maximization* (EM) algorithm [20].

One of the most straight-forward approaches to semi-supervised learning is through self-training [17, 18, 19, 21, 22]. Due to its descriptive nature, it is generally applicable to every form of learning-based classification and recognition system. It is an iterative learning scheme whose underlying idea is similar to EM. A weak, initial recognizer is trained using the labeled data only. Afterwards, during the self-training iterations, the recognizer is used to label the entire set of unlabeled data and assign a confidence measure to each classification. The most confidently classified elements are assumed to be labeled correctly and added to the training set, which is then used to train a new recognizer.

Theoretical background whether or not semi-supervised learning works and under what circumstances, is limited and restricted to a few cases. Some work

exists showing that unlabeled data can help. However, formal properties can be proven only under very restrictive settings. An analysis about the generalization error in a *probably approximately correct* (PAC) [23] framework has been attempted in [24]. Still, the resulting error bounds are “usually ridiculously far from being tight” [5]. Furthermore, a rather negative picture of the usefulness of unlabeled data is painted in [25]. Generally it can neither be guaranteed that unlabeled data help nor that it does not even have a negative influence. In fact, when the model assumptions of the mixture models of the input distributions are wrong, “unlabeled data is dangerous” [26].

3.2. Semi-Supervised Learning for Handwriting Recognition

Straight-forward semi-supervised learning frameworks flexible enough to handle the restrictions imposed by the many-to-few mapping of sequential data are self-training and co-training. For recordings of a TV news channel, self-training for speech recognition has been successfully applied in [27, 28, 29, 30]. In the domain of handwritten text, only few works have been published. A semi-supervised adaptation to a specific writer is presented in [9, 31], where the system is adjusted to increase the recognition accuracy of a specific writer at the cost of reducing the unconstrained recognition accuracy. To the best knowledge of the authors, the only work on semi-supervised learning for unconstrained handwriting recognition, is earlier work done by the authors of this paper [32].

4. Keyword Spotting

Keyword spotting is the task of retrieving all instances of a given word or phrase from a collection of documents. Due to the multitude of possible scenarios the details on how this is done can vary immensely. In general, keyword spotting can be classified in various ways. A popular scheme is to differentiate between *query-by-example* and *query-by-string*. In the former case, a region of a document is defined by the user and the system should return all regions containing the same text. *Query-by-string*, on the other hand, supports search queries of arbitrary character combinations, regardless of whether they occur in the dataset or not. Since this needs obviously a model for every character, these methods are often associated with learning-based approaches, while *query-by-example* can be done without any learning step.

A further criterion used for structuring keyword spotting approaches is the type and pre-segmentation of input data, since methods differ for single words, entire text lines, or even whole, unsegmented pages. The return value is usually a spotting score (the higher, the more likely it is that the keyword occurs in the input document) and for line- and page-based approaches sometimes also a position or bounding box.

For these last two cases we want to determine, for a fixed region R , the probability of the keyword w to occur in that region, which is the probability to be in any of the subregions $r \subset R$. This is often approximated by finding the

probability of the keyword at its most likely place.

$$\text{score}(w|R) = \prod_{r \in R} p(w|r) \tag{1}$$

$$\approx \max_{r \in R} p(w|r) \tag{2}$$

4.1. BLSTM NN based Keyword Spotting

In this paper we use the BLSTM NN-based text line recognition system of [15] for keyword spotting. As stated in Section 2, the neural network act as a mapping of the input sequence to a sequence of posterior character probabilities which serves as the input into the CVC Token Passing algorithm.

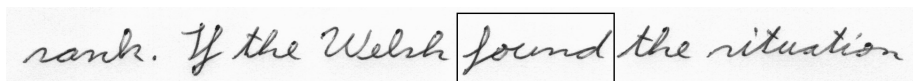
For keyword spotting, however, the CTC Token Passing algorithm is not performed and the character probability sequence is extended by an additional entry with a constant value of 1. By adding this symbol at the beginning and at the end of the keyword, the algorithm finds the best path through the output matrix that passes through the symbol added at the beginning, then through all the characters of the keyword, and then through the symbol added at the end. In other words, the path traverses through the letters of the keyword at their most likely position while the rest of the text line has no influence. This way, we get a keyword spotting score that reflects the product of all character probabilities of the optimal subsequence that starts with the space before the first character of the keyword and ends with the space after the its last character.

Note that the spotting score is a product of character probabilities and depends therefore on the length of the keyword. Hence, to normalize this score, in order to make results comparable to results from other keywords, the logarithmic spotting score is divided by the keyword’s length (the number of letters in the keyword). The output of the retrieval system is a ranked list of lines, or positions, together with the likelihood of the keyword. An example output of the system is shown in Fig. 2. For more details, we refer to [33, 34].

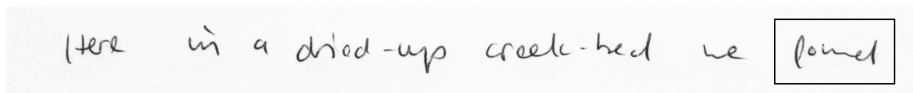
4.2. Combination of Several Systems

To increase the performance of different keyword spotting systems, combination techniques can be applied. We exploit the fact that BLSTM neural networks are initialized before training with random values. Clearly, two differently initialized networks are likely to produce a different output, even when trained on the same training data, which renders the generation of an ensemble of different keyword spotting systems a straight forward task. Hence the combination on an ensemble of neural networks, even when trained on exactly the same data, is likely to perform better than any single neural network.

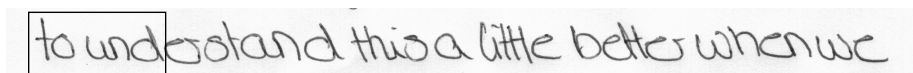
In [35], the authors investigate different combination techniques for keyword spotting algorithms in general and BLSTM neural network based systems in particular. By averaging the returned likelihood of all systems before ranking the output, a significant increase in the keyword spotting performance was obtained. We follow the same approach in this paper. Instead of one neural network, an ensemble of several neural networks is used, each of which spots a

A horizontal strip of handwritten text in cursive script. The text reads "rank. If the Welsh found the situation". The word "found" is enclosed in a black rectangular box.

(a) Returned log Likelihood: -1.7125

A horizontal strip of handwritten text in cursive script. The text reads "Here in a dried-up creek-bed we found". The word "found" is enclosed in a black rectangular box.

(b) Returned log Likelihood: -8.4097

A horizontal strip of handwritten text in cursive script. The text reads "to understand this a little better when we". The words "to understand" are enclosed in a black rectangular box.

(c) Returned log Likelihood: -11.0900

Figure 2: Search results for the word “found”.

given keyword separately. Afterwards, the normalized likelihood scores of the networks are averaged. Note that the returned keyword spotting score of a network returns the likelihood of a word at its most likely position in the text line, regardless of whether or not the positions found by the different systems are the same.

To infer the final position of a word, the networks are ranked according to their performance on a validation set and the keyword position returned by the best network is used. In case of multiple occurrences of a keyword on the same line, only the best one is returned. However, to return all keywords, the algorithm can be repeated on the remaining parts of the text line.

5. Proposed Approach

In this paper we propose to perform self-training to improve the performance of a handwriting recognition system by using keyword spotting to find words in the set of unlabeled data that are used for retraining. This is different from previous approaches, where a recognition system was used to recognize the unlabeled text lines.

To be used for semi-supervised learning, the final application and the specific recognizer dictate the type of keyword spotting to be considered. In the application at hand we focus on learning-based approaches for pre-segmented text lines. Hence, we want to find words in the set of unlabeled data by using keyword spotting instead of a full fledged recognizer. For the popular recognition systems based on HMM and BLSTM NN, fast keyword spotting approaches without the need of a language model have been proposed [33, 36].

Of course, the best keyword spotting system is a complete text recognizer, given enough time and a sufficient language model. Since the proposed methods do not consider any context and language information, the results may not be as

accurate. Yet, in this paper, we demonstrate that it is possible to design a setup for semi-supervised learning based on keyword spotting that does not suffer from a performance decrease, compared to a system based on text line recognition, while at the same time, the benefits of having a reduced computation time are dramatic.

In fact, keyword spotting is several orders of magnitude faster than text line transcription. To give a quantitative evaluation of the run time, consider that it takes several minutes to decode a text line, while a keyword can be spotted with the same system in about one millisecond [33].

Independent of the specific method, the selection of new elements used for retraining the system has to be done with great care. When only those few elements whose label can be estimated with the highest confidence are added, the training set does not change substantially. Furthermore, the most confidently recognized elements are often in the center of the area of the input space that is already assigned to one class, hence the risk of selecting elements that do not make a difference is very high. Increasing the number of elements, on the other side, comes at the risk of adding noise to the training data and too much noise in the training data will impede the recognition. Hence, a good trade-off between data quality and data quantity is crucial to the success of the approach.

A further advantage of using keyword spotting for self-training is that the expected noise of the added elements is an input parameter and can be controlled directly, rather than measuring it afterwards. This allows for a more detailed analysis of the achieved performance gain.

5.1. Precision Based Thresholds

The proposed procedure works as follows. All words in the keyword list are spotted on the validation set. Then, threshold θ is set to the normalized likelihood score value that produces a precision of $\theta_{val}(prec)$ on the validation set (cf. Fig. 3). Finally, all words in the keyword list are spotted on the entire set of unlabeled data and every word that has been spotted with a matching score higher than θ is added to the training set. In Fig. 4, an overview of the proposed system is given.

5.2. Combining Several Precisions

The set of words S_θ that are added to the training set are chosen according to their spotting score θ , which, in turn is computed from the desired precision value $\theta = \theta_{prec}$. Obviously, sets resulting from different spotting thresholds form a chain:

$$S_{\theta^1} \subset S_{\theta^2} \subset \dots \subset S_{\theta^n} \text{ for } \theta^1 > \theta^2 > \dots > \theta^n$$

Given several precision thresholds, it is possible to add different sets to the re-training data. This has the effect, that some of the elements are added multiple times, depending upon how confidently they are spotted. Thus, in a combination $S_{\theta^1} + S_{\theta^2}$ all keywords spotted with a precision higher than θ^1 are weighted twice.

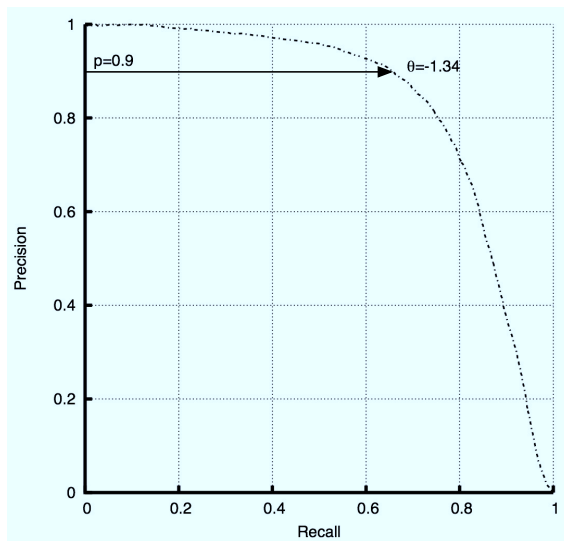


Figure 3: Determining the likelihood threshold using the validation set and the desired precision.

6. Experimental Evaluation

6.1. Comparison with standard approach

To compare the proposed approach to existing self-training approaches that perform a complete text line recognition using a language model, we performed experiments on a dataset of continuous text lines containing modern English data.

6.1.1. Setup

We used the IAM off-line database, which consists of 1 539 pages written by 657 writers [13]¹. The text segments originate from the London/Oslo/Bergen (LOB) corpus [37] and form a representative cross-section of modern English language. The database is split up into a working set of 6 161 text lines, a validation set of 920 text lines and a writer independent test set of 929 text lines. The three sets are writer disjunct, i.e., a person who has contributed to any one of the three sets did not contribute to any of the two other sets. The working set is randomly split up into a training set consisting of 1 000 labeled text lines and a set of 5 161 unlabeled text lines.

In the first set of experiments, the effects of using several different precision thresholds are compared to each other. Five precision thresholds are investigated, $\theta_{prec}^1 = 0.5$, $\theta_{prec}^2 = 0.8$, $\theta_{prec}^3 = 0.9$, $\theta_{prec}^4 = 0.95$, and $\theta_{prec}^5 = 0.99$.

¹<http://www.iam.unibe.ch/fki/databases>

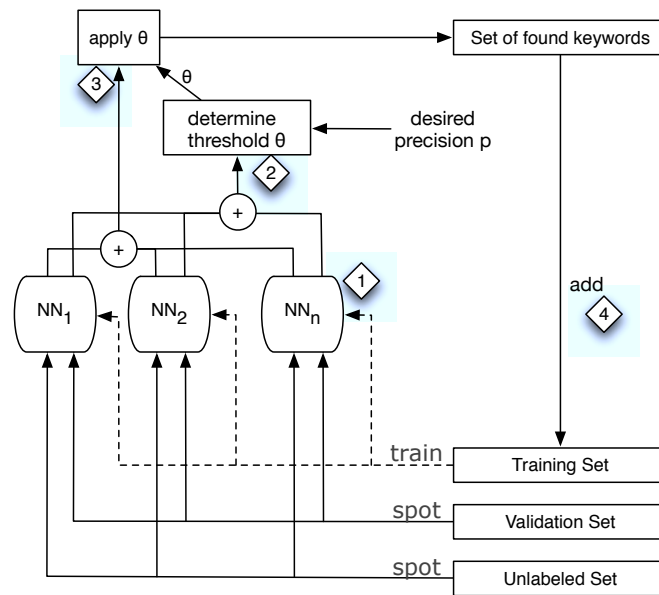


Figure 4: Overview of the proposed system. An initial, labeled training set is used to train an ensemble of BSLTM neural networks (1). With these networks, keywords are spotted on the validation set using a combination technique. The performance is used to compute the threshold ϑ , according to a desired precision value (2). Next, the ensemble is used to spot the same keywords on the set of unlabeled data. All positions that are found with a likelihood greater than ϑ (3) are considered to be correct. Finally, these words are added to the training set (4).

In a second set of experiments, the extension of combining several thresholds is investigated in which several instances of the same word can be added to the training set several times. The following six setups are used.

$$\begin{aligned}
A &= S_{0.8} + S_{0.9} \\
B &= S_{0.8} + S_{0.95} \\
C &= S_{0.8} + S_{0.99} \\
D &= S_{0.8} + S_{0.95} + S_{0.99} \\
E &= S_{0.8} + S_{0.99} + S_{0.99} \\
F &= S_{0.8} + S_{0.9} + S_{0.95} + S_{0.99}
\end{aligned}$$

All of these setup start with the basic set of $S_{0.8}$ and give additional weight to the most confident recognition. In order to to this, sets A , B , and C each add one of the previously used subsets of $S_{0.8}$, namely $S_{0.9}$, $S_{0.95}$, or $S_{0.99}$. Setup D and E extend the idea to combine three sets and setup F finally adds up all four sets.

6.1.2. Results

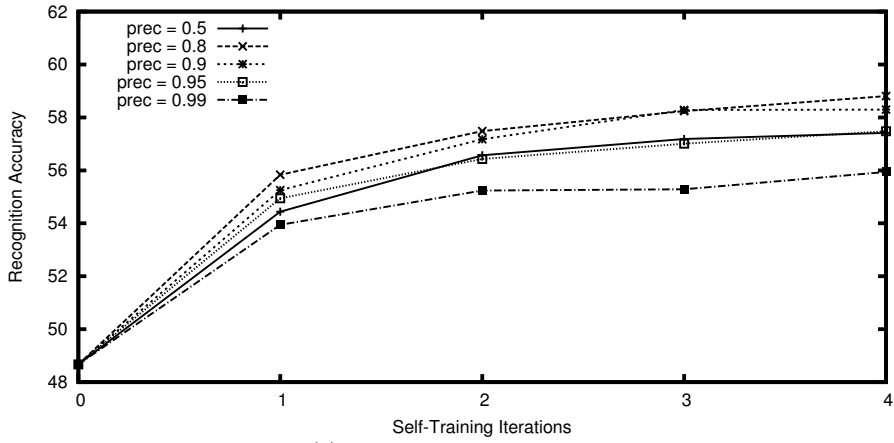
The performance of the systems as a function of the self-training iterations is shown in Fig. 5. In Fig. 5(a), the performance of using single precision values as retraining decision is given. Fig. 5(b) shows the extension that combined the sets from several thresholds. Finally, Fig. 5(c) displays, as a comparison, the performance of the transcription-based self-training system that makes use of a language model.

Several observation can be made. First of all, we note that all investigated retraining rules achieve a statistically significant performance increase compared to the one trained on the initial training set.

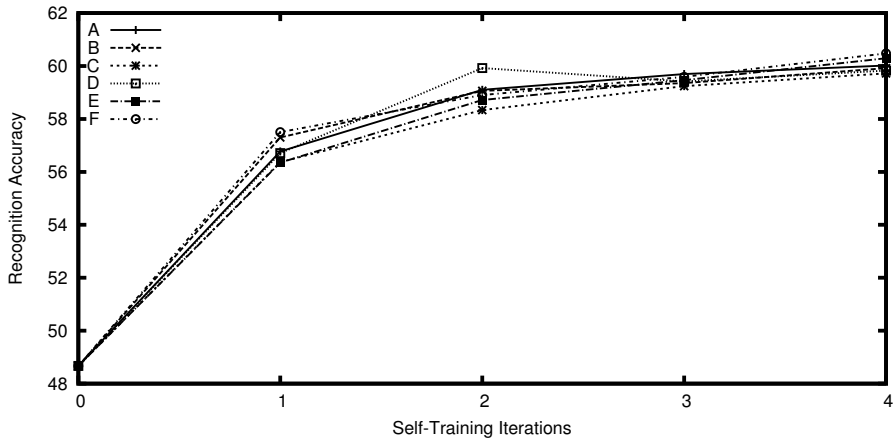
The performance of the retraining rules that make use of a single precision value in Fig. 5(a) clearly demonstrate the importance of a good balance between data quality and data quantity. The most conservative retraining rule that only used images that are found with a precision of $prec = 0.99$ has the lowest performance. By relaxing the precision threshold, the performance increases until it reaches a maximum at $prec = 0.8$. The least restricting retraining rule $prec = 0.5$ again performs significantly worse.

From Fig. 5(a) and Fig. 5(b) it becomes apparent that retraining rules based on a single precision value perform overall not as good as the retraining rules that combine the retraining sets of several precision values. Interestingly, the specific combination of the thresholds does not seem to have a substantial impact. The networks retrained using the worst combination, $D = S_{0.8} \cup S_{0.95} \cup S_{0.99}$ reach an average recognition accuracy of 59.82%, while retraining using the best combination, $F = S_{0.8} \cup S_{0.9} \cup S_{0.95} \cup S_{0.99}$ leads to an accuracy of 60.48%.

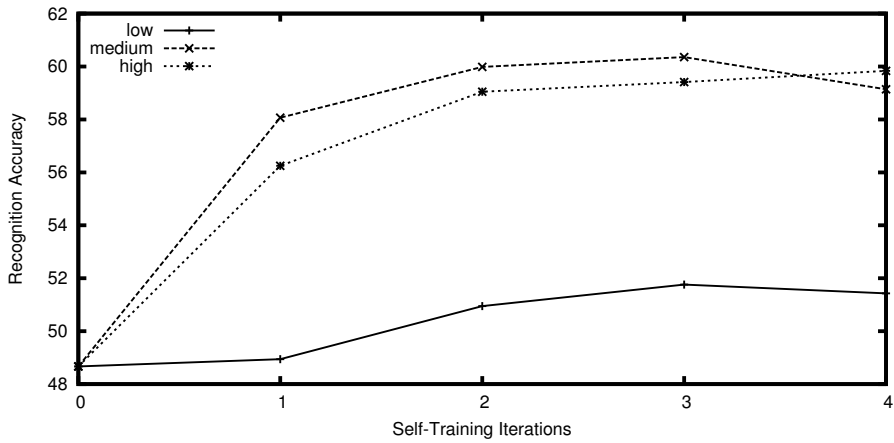
The comparison with the reference system shows that the keyword spotting based retraining rules perform worse than the best transcription based retraining rules during the first 3 iterations. However, the performance increase using



(a) Single Precision Threshold



(b) Combining Several Precisions



(c) transcription-based reference system

keyword spotting based self-training continues longer and the final results after the fourth iteration are better than the reference system.

As a final comparison and to evaluate how good the BLSTM NN system can get on this database, we trained ten neural networks on the entire working set of 6 161 labeled text lines and reached an accuracy of 71%.

6.2. Application for Historical documents

Additionally we wanted to investigate the applicability of the proposed approach for historical data. For historic data, modeling the underlying language poses several challenges because, among other problems, not enough texts are available to form a large corpus. Also the lack of a unified spelling system reduces the number of instances of each word and written texts are often in the form of poetry or administrative texts. For such highly specialized texts, a general language model does not seem highly appropriate.

For historic texts, generating a valid ground can be quite costly, since linguistic experts are needed to translate the text. Hence, for these experiments, we focus on the final recognition accuracy as a function of the size of the input data.

6.2.1. Setup

We use the PARZIVAL database presented in [38] for our experimental evaluation. This database contains digital images of medieval manuscripts originating in the 13th century. Arranged in 16 books, the epic poem *Parzival* by Wolfram von Eschenbach was written down in Middle High German with ink on parchment. There exist multiple manuscripts of the poem that differ in writing style and dialect of the language. The manuscript used for experimental evaluation is *St. Gall, collegiate library, cod. 857* that is written by multiple authors [39]. Figure 6 shows a sample from this database.

For each of the individual setups, we trained 10 initial neural networks and tested them individually on a validation set. The four networks with the lowest word error rate were then used in the self-training iterations. To keep the computational costs within reasonable bounds, the retraining of each neural network was limited to a maximum of 50 epochs. It stopped sooner if no improvement in label error rate was achieved for 11 epochs.

Inspired by the results on the IAM database, we chose to investigate the following self-training rules

| Single Precision Threshold | Combined Precision Thresholds |
|----------------------------|--|
| $S_{0.8}$ | $A = S_{0.8} \cup S_{0.9}$ |
| $S_{0.9}$ | $D = S_{0.8} \cup S_{0.95} \cup S_{0.99}$ |
| $S_{0.99}$ | $F = S_{0.8} \cup S_{0.9} \cup S_{0.95} \cup S_{0.99}$ |

We chose to test five different sizes of labeled training data. For this, we numbered all text lines and assigned to each text lines its number modulo 6. Then, the assignment to the different sets was done according to the following identification scheme

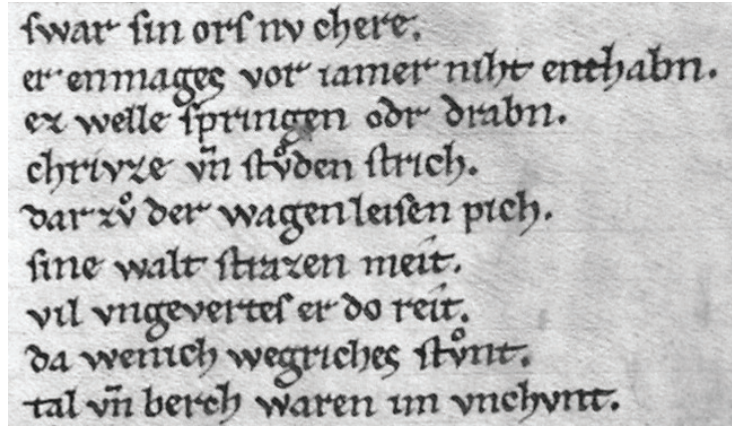


Figure 6: A sample from the PARZIVAL DB

Table 1: The number of text line for the different setups

| Setup | labeled set | unlabeled set | validation set | testing set |
|-------|-------------|---------------|----------------|-------------|
| tr4 | 2985 | 0 | 764 | 764 |
| tr2 | 1493 | 1492 | 764 | 764 |
| tr1 | 747 | 1492 | 764 | 764 |
| tr.2 | 347 | 1492 | 764 | 764 |
| tr.4 | 187 | 1492 | 764 | 764 |
| tr.8 | 94 | 1492 | 764 | 764 |

| line number modulo 6 | assigned set |
|----------------------|----------------|
| 0 | tr4, tr2, tr1 |
| 1 | tr4, tr2 |
| 2 | tr4, unlabeled |
| 3 | tr4, unlabeled |
| 4 | validation set |
| 5 | testing set |

This means, testing set and validation set each gets assigned $\frac{1}{6}$ th of the data and the set of unlabeled data gets one third of the text lines. Set *tr4* gets $\frac{4}{6}$ th of the text lines and serves as a reference setup where all text lines are labeled. For the experiments, we have *tr2* with $\frac{2}{6}$ th and *tr1* with $\frac{1}{6}$ th of the text lines as labeled data. We further reduced the number of labeled text lines using *tr.2* ($\frac{1}{12}$ th labeled lines), *tr.4* ($\frac{1}{24}$ th labeled lines), and *tr.8* ($\frac{1}{48}$ th labeled lines). The absolute numbers of the text lines for the different setups are given in Table 1.

The keywords spotted are all words of size larger than two characters that

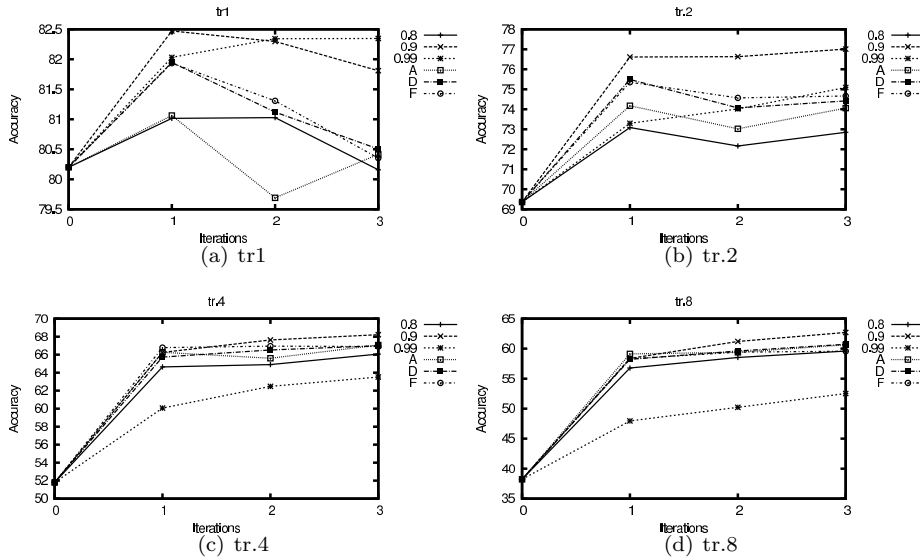


Figure 7: The detailed performance of the proposed algorithm on the PARZIVAL dataset for selected setups.

are found in the training set.

6.2.2. Results

To analyze the impact of self-training, we evaluated the recognition performance of the neural networks before and after each self-training iteration. The recognition has been done using a closed-vocabulary setup and with a bigram language model. The language model is trained on the combination of the labeled text lines and the validation set and smoothed using Kneser-Ney smoothing [40] using the SRILM toolkit.

The detailed results of the experiments for selected system can be seen in Fig. 7 where the word recognition accuracy of the testing set is shown as a



Figure 8: The initial recognition accuracy and the final recognition accuracy after 3 iterations of self-training.

Table 2: The word recognition accuracies in % before as well as after three self-training iterations for each self-training rule.

| | initial | $S_{0.8}$ | $S_{0.9}$ | $S_{0.99}$ | A | D | F |
|------|---------|-----------|--------------|--------------|-------|-------|-------|
| tr.8 | 38.22 | 59.60 | 62.71 | 52.52 | 60.65 | 60.73 | 59.57 |
| tr.4 | 51.79 | 66.07 | 68.22 | 63.52 | 67.06 | 67.00 | 66.90 |
| tr.2 | 69.36 | 72.85 | 77.01 | 75.09 | 74.06 | 74.42 | 74.67 |
| tr1 | 80.2 | 80.16 | 81.81 | 82.35 | 80.42 | 80.50 | 80.35 |
| tr2 | 83.71 | 82.98 | 84.39 | 84.66 | 83.44 | 84.19 | 83.81 |

function of the self-training iterations. The initial system is plotted at iteration number 0. In Fig. 8, a comprehensive summary of all results is depicted. This plot shows the initial and the final recognition accuracy (of the best self-training rule after) for the different sizes of labeled training lines as well as the performance of the tr_4 reference system. Exact numbers are shown in Table 2.

Observing the results of this experiment, the following observations can be made. First, one can clearly see the correlation between the size of the training data and the initial recognition accuracy on the validation set. A training set of 94 text lines results in a system that achieves a recognition accuracy of 38.22%, while 2985 lines in the training set lead to a system with 83.71%. Secondly, we can observe in Fig. 8 that self-training can be successfully applied for all tested sizes of the labeled training set. Furthermore we can see, that the achieved increase is larger, for less trained systems. With just 94 labeled text lines, the proposed approach achieves an remarkable increase in recognition accuracy from 38.22% to 62.71%.

One can also see, that the best performing self-training rule is not constant for all setups. For less trained systems (tr.8, tr.4, and tr.2), more relaxed retraining rules ($S_{0.9}$) seem to be better, while for better trained systems (tr1 and tr2) stricter rules need to be applied. In fact, a very relaxed self-training rule ($S_{0.8}$) impedes the recognition rate for those systems that are already well trained. The composed retraining rules (A , D , and F) also lead to an increase in recognition accuracy. However, they do not, contrary to the experiments on the IAM database, consistently outperform the single precision threshold retraining rules.

7. Conclusion

Words used for spotting are the ones found in the training set. Note, however, that due to the normalization of the likelihood, one global threshold can be applied to all keywords as opposed to keyword specific thresholds. Therefore, the overall best words are selected and if a keyword does not occur in the database, all spotted positions can easily be rejected. This would also enable the possibility to create lists of keywords automatically by using random character combinations.

An experimental comparison with an existing self-training approach shows an average increase in recognition accuracy from 48.67% to 60.48% using a new set of retraining rules could be achieved. In addition to being more resourceful, the new rules perform slightly better than current transcription-based approaches. This demonstrates that the lack of a language model does not necessarily decrease the accuracy gain achieved by self-training.

Further experiments demonstrate that self-training can successfully be applied to reduce the amount of work for the increasingly popular field of historical document processing. The recognition accuracy of a weakly trained system with just 94 labeled text lines could automatically be increased from 38.22% to 62.71%.

Finally, these results also give interesting insights into how BLSTM neural network learn character models. The results indicate that for weakly trained systems, a perfect transcription is not necessary for training. Once a character is well modeled by the network, noise in the training set, however, has a damaging influence.

Future work includes an integration of this approach into a complete bootstrapping framework for new recognition systems. In order to further increase the performance, co-training approaches using different keyword spotting methodologies might also be investigated.

Acknowledgments

We thank Alex Graves for kindly providing us with the BLSTM Neural Network source code. This work has been supported by the European project FP7-PEOPLE-2008-IAPP: 230653 as well as by the Swiss National Science Foundation (Project CRSI22_125220).

References

- [1] A. Vinciarelli, A Survey On Off-Line Cursive Word Recognition, *Pattern Recognition* 35 (7) (2002) 1433–1446.
- [2] R. Plamondon, S. N. Srihari, On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey, *IEEE Transaction on Pattern Analysis and Machine Intelligence* 22 (1) (2000) 63–84.
- [3] A. H. Toselli, E. Vidal, F. Casacuberta, *Multimodal Interactive Pattern Recognition and Applications*, Springer-Verlag, 2011.
- [4] X. Zhu, Semi-Supervised Learning Literature Survey, Tech. Rep. 1530, Computer Science, University of Wisconsin-Madison, http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf (2005).
- [5] M. Seeger, Learning with Labeled and Unlabeled Data, Tech. rep., University of Edinburgh, 5 Forest Hill, Edinburgh, EH1 2QL (2002).

- [6] S. Ji, L. T. Watson, L. Carin, Semi-supervised Learning of Hidden Markov Models via a Homotopy Method, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31 (2) (2009) 275–287.
- [7] M. Inoue, N. Ueda, Exploitation of Unlabeled Sequences in Hidden Markov Models, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (12) (2003) 1570–1581.
- [8] G. R. Ball, S. N. Srihari, Prototype Integration in Off-Line Handwriting Recognition Adaptation, in: *Proc. Int’l. Conf. on Frontiers in Handwriting Recognition*, 2008, pp. 529–534.
- [9] G. R. Ball, S. N. Srihari, Semi-supervised Learning for Handwriting Recognition, in: *10th Int’l Conf. on Document Analysis and Recognition*, 2009, pp. 26–30.
- [10] O. Chapelle, B. Schölkopf, A. Zien, *Semi-Supervised Learning*, MIT Press, Cambridge, MA, 2006.
- [11] V. Frinken, H. Bunke, Self-Training Strategies for Handwritten Word Recognition, in: *9th Industrial Conference on Data Mining, Lecture Notes in Artificial Intelligence*, 2009, pp. 291–300.
- [12] V. Frinken, A. Fischer, M. Baumgartner, H. Bunke, Semi-Supervised Learning for Cursive Handwriting Recognition using Keyword Spotting, in: *Int’l Conf. on Frontiers in Handwriting Recognition*, 2012, pp. 49–54.
- [13] U.-V. Marti, H. Bunke, The IAM-Database: An English Sentence Database for Offline Handwriting Recognition, *Int’l Journal on Document Analysis and Recognition* 5 (2002) 39–46.
- [14] U.-V. Marti, H. Bunke, 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 (2001) 65–90.
- [15] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, J. Schmidhuber, A Novel Connectionist System for Unconstrained Handwriting Recognition, *IEEE Transaction on Pattern Analysis and Machine Intelligence* 31 (5) (2009) 855–868.
- [16] A. Graves, S. Fernández, F. Gomez, J. Schmidhuber, Connectionist Temporal Classification: Labelling Unsegmented Sequential Data with Recurrent Neural Networks, in: *23rd Int’l Conf. on Machine Learning*, 2006, pp. 369–376.
- [17] A. K. Agrawala, Learning with a Probabilistic Teacher, *IEEE Trans. Information Theory* 16 (1970) 373–379.

- [18] S. C. Fralick, Learning to Recognize Patterns without a Teacher, *IEEE Trans. Information Theory* 13 (1967) 67–64.
- [19] H. J. Scudder, Probability of Error of Some Adaptive Pattern-Recognition Machines, *IEEE Transaction on information Theory* 11 (1965) 363–371.
- [20] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society, Series B* 39 (1) (1977) 1–38.
- [21] K. Nigam, R. Ghani, Understanding the Behavior of Co-training, in: *KDD-2000 Workshop on Text Mining*, 2000, pp. 105–107.
- [22] D. Yarowsky, Unsupervised Word Sense Disambiguation Rivaling Supervised Mmethods, in: *33rd Annual Meeting on Association for Computational Linguistics*, 1995, pp. 189–196.
- [23] L. Valiant, A Theory of the Learnable, *Communications of the ACM* 27 (1984) 1134–1142.
- [24] A. Blum, T. Mitchell, Combining Labeled and Unlabeled Data with Co-Training, in: *COLT’ 98: Proc. of the 11th annual Conference on Computational Learning Theory*, ACM, New York, NY, USA, 1998, pp. 92–100.
- [25] S. Ben-David, T. Lu, D. Pál, Does Unlabeled Data Provably Help? Worst-case Analysis of the Sample Complexity of Semi-Supervised Learning, in: *21st Annual Conf. on Learning Theory*, 2008.
- [26] F. G. Cozman, I. Cohen, Unlabeled Data Can Degrade Classification Performance of Generative Classifiers, in: *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*, AAAI Press, 2002, pp. 327–331.
- [27] T. Kemp, A. Waibel, Unsupervised Training of a Speech Recognizer: Recent Experiments, in: *ESCA (Ed.), Proceedings of the EUROSPEECH 1999*, no. 6, 1999, pp. 2725–2728.
- [28] L. Lamel, J.-L. Gauvain, G. Adda, Lightly Supervised and Unsupervised Acoustic Model Training, *Computer Speech and Language* 16 (2002) 115–129.
- [29] D. Yu, B. Varadarajan, L. Deng, A. Acero, Active Learning and Semi-Supervised Learning for Speech Recognition: A Unified Framework Using the Global Entropy Reduction Maximization Criterion, *Computer Speech and Language* 24 (3) (2010) 433–444.
- [30] F. Wessel, H. Ney, Unsupervised Training of Acoustic Models for Large Vocabulary Continuous Speech Recognition, *IEEE Transactions on speech and audio processing* 13 (1) (2005) 23–31.

- [31] P. Dreuw, D. Rybach, C. Gollan, H. Ney, Writer Adaptive Training and Writing Variant Model Refinement for Offline Arabic Handwriting Recognition, in: 10th Int'l Conf. on Document Analysis and Recognition, Vol. 1, 2009, pp. 21–25.
- [32] V. Frinken, H. Bunke, Self-Training for Handwritten Text Line Recognition, in: 15th Iberoamerican Congress on Pattern Recognition, 2010, pp. 104–112.
- [33] V. Frinken, A. Fischer, R. Manmatha, H. Bunke, A Novel Word Spotting Method Based on Recurrent Neural Networks, IEEE Trans. on Pattern Analysis and Machine Intelligence.
- [34] A. Fischer, A. Keller, V. Frinken, H. Bunke, HMM-Based Word Spotting in Handwritten Documents Using Subword Models, in: 20th Int'l Conf. on Pattern Recognition, 2010, pp. 3416–3419.
- [35] V. Frinken, A. Fischer, H. Bunke, Combining Neural Networks to Improve Performance of Handwritten Keyword Spotting, in: N. El Gayar, J. Kittler, F. Roli (Eds.), 9th Int'l Workshop on Multiple Classifier Systems, LNCS, 2010, pp. 215–224.
- [36] A. Fischer, A. Keller, V. Frinken, H. Bunke, Lexicon-Free Handwritten Word Spotting Using Character HMMs, submitted.
- [37] H. Kucera, W. N. Francis, Manual of Information to accompany A Standard Corpus of Present-Day Edited American English, for use with Digital Computers, Brown University, Department of Linguistics, Providence, Rhode Island, revised 1971. Revised and amplified 1979. (1964).
- [38] A. Fischer, M. Wüthrich, M. Liwicki, V. Frinken, H. Bunke, G. Viehhauser, M. Stolz, Automatic Transcription of Handwritten Medieval Documents, in: 15th International Conference on Virtual Systems and Multimedia, 2009, pp. 137–142.
- [39] M. Wüthrich, M. Liwicki, A. Fischer, E. Indermühle, H. Bunke, G. Viehhauser, M. Stolz, Language Model Integration for the Recognition of Handwritten Medieval Documents, in: 10th Int'l Conf. on Document Analysis and Recognition, 2009, pp. 211–215.
- [40] R. Kneser, H. Ney, Improved Backing-Off for M-Gram Language Modeling, in: Int'l Conf. Acoustic, Speech, and Signal Processing, Vol. 1, 1995, pp. 181–184.