Offline Cursive Handwriting: From Word To Text Recognition

Alessandro Vinciarelli [a]

IDIAP–RR 03-24

April 2003

[a] Institut Dalle Molle Intelligence Artificielle Perceptive, Rue du Simplon 4, 1920 Martigny, Switzerland. E-mail vincia@idiap.ch

IDIAP Research Report 03-24

Offline Cursive Handwriting: From Word To Text Recognition

Alessandro Vinciarelli

April 2003

Contents

Chapter 1

Introduction

1.1 Introduction

This thesis presents a PhD work on offline cursive handwriting recognition, the automatic transcription of cursive data when only its image is available.
Two main approaches were used in the literature to solve the problem. The first one attempts to segment the words into letters and then applies Dynamic Programming techniques to perform the recognition. The second one converts the words into sequences of observation vectors and then recognizes them with Hidden Markov Models. Some efforts were made to apply human reading inspired models, but their success was limited to applications involving small lexica (20-30 words).
The HMM based approach became dominant after some years and now few works still attempt to segment the words into characters. The Sayre's paradox (a word cannot be segmented before being recognized and cannot be recognized before being segmented [128]) was shown to be an important limit of any segmentation based approach.
The research focused essentially on two applications: postal address recognition and bankcheck reading. In both cases, the application environment provides information helpful for the recognition (the zip code and the courtesy amount respectively). This simplified the problem and had an important effect on the state of the art where several issues appear to be neglected:

- The steps preceeding the recognition (preprocessing, normalization, segmentation and feature extraction) are based on heuristics often involving data dependent algorithms and parameters to be set empirically. This leads to systems that are not robust with respect to a change of data and require a heavy experimental effort to find an optimal configuration of the parameter set.

- Although HMMs offer a solid theoretic framework to deal with the recognition problem, most of the systems presented in the literature are still far from taking full advantage from their application.

- The research focused on single word recognition neglecting, with very few exceptions, the transcription of texts or word sequences (e.g. dates) where language modeling can be used.

- The experimental setup is not always rigorous. It is frequent to find works where a clear distinction between training and test set is not made. The systems are often fitted to the test set by tuning the hyperparameters over it. This leads to an overestimation of the performance.

In this thesis, the problem of cursive handwriting recognition is addressed taking into account the above issues. The system we developed is based on a sliding window approach. This allows one to avoid the segmentation, a step that typically involves ad hoc algorithms and heuristics that risk to be data dependent and not robust with respect to a change of data.

The normalization (slant and slope removal) is based on a statistical approach that assumes very general hypotheses about the words and that it is completely adaptive. No hyperparameters are involved so that, when changing the data, no effort is needed to reconfigure this step for the new problem.
Several modeling aspects have been investigated in order to take more advantage from the HMM framework. The use of transforms (linear and nonlinear Principal Component Analysis, Independent Component Analysis) making the feature vectors more suitable for the recognition are shown to improve significantly the recognition rate. HMM adaptation techniques (Maximum Likelihood Linear Regression and Maximum A Posteriori adaptation) have been used to improve the performance of a system trained using multiple writer data over the words written by a specific person.
The recognition has been extended from single word to text line recognition. In this case, the only knowledge we have about the data to be recognized is that they are written in English and they must reproduce on average the statistics of any english text. Such hypothesis is used by applying Statistical Language Models (unigrams, bigrams and trigrams). Their application results in a significant improvement of the system.
All the performed experiments were designed to be correct from a statistical point of view, to avoid any fitting of the system to the test set and to give realistic measures of the performance.
The rest of this thesis is organzed as follows: chapter 2 gives an extensive survey of the state of the art in the domain, chapter 3 describes the recognition system we developed and used in our experiments, chapter 4 presents the data transforms we applied and the results of their application, chapter 5 reports about the use of HMM adaptation techniques, chapter 6 shows how the recognition was extended from single words to text line recognition and the final chapter 7 draws some conclusions and delineates possible future works.

Chapter 2

State of the art

2.1 Introduction

Handwriting recognition is the transcription of handwritten data into digital format. The aim is to process data that normally could be processed only by humans with computers. The evident advantage of computer processing is the possibility of dealing with huge amounts of information at high speed. The acquisition of the handwritten data can be performed online or offline. In the first case, special devices allow the computer to track and store the trajectory of the pen tip. In the second case, the image of handwritten data is scanned and stored as an image. The main difference is that online data contains temporal information that is not available in the offline case. This is important because it makes the recognition techniques typically applied in the literature more effective. This thesis presents a PhD work on offline cursive handwriting recognition.
Two main approaches were used in the literature to solve the problem. The first one attempts to segment the words into letters and then applies Dynamic Programming techniques to perform the recognition. The second approach converts the words into sequences of observation vectors and then recognizes them with Hidden Markov Models. Some efforts were made to apply human reading inspired models, but their success was limited to applications involving small lexica (20-30 words).
The HMM based approach became dominant after some years, and now, relatively few researchers still attempt to segment the words into characters. The Sayre's paradox (a word cannot be segmented before being recognized and cannot be recognized before being segmented [128]) was shown to be a significant limit of any segmentation based approach.
The research focused essentially on two applications: postal address recognition and bankcheck reading. In both cases, the application environment provides information that aids for the recognition (the zip code and the courtesy amount respectively). This simplified the problem and had an important effect on the state of the art where several issues appear to be neglected:

- The steps preceeding the recognition (preprocessing, normalization, segmentation and feature extraction) are based on heuristics often involving data dependent algorithms and parameters to be set empirically. This leads to systems that are not robust with respect to a change of data and require a heavy experimental effort to find an optimal configuration of the parameter set.

- Although HMMs offer a solid theoretic framework to deal with the recognition problem, most of the systems presented in the literature are still far from taking full advantage from their application.

- The research focused, with very few exceptions, on single word recognition neglecting. Little attention was payd to the transcription of texts or word sequences (e.g. dates) where language modeling can be used.

- The experimental setup is not always rigorous. It is frequent to find works where a clear distinction between training and test set is not made. The systems are often fitted to the test set by tuning the hyperparameters over it. This leads to an overestimation of the performance.

In this thesis, the problem of cursive handwriting recognition is addressed taking into account the above issues. The system we developed is based on a sliding window approach. This allows one to avoid the segmentation, a step that typically involves ad hoc algorithms and heuristics that risk to be data dependent and not robust with respect to a change of data.
The normalization (slant and slope removal) is based on a statistical approach that assumes very general hypotheses about the words and that it is completely adaptive. No hyperparameters are involved so that, when changing the data, no effort is needed to reconfigure this step for the new problem.
Several modeling aspects have been investigated in order to take more advantage from the HMM framework. The use of transforms (linear and nonlinear Principal Component Analysis, Independent Component Analysis) making the feature vectors more suitable for the recognition are shown to improve significantly the recognition rate. HMM adaptation techniques (Maximum Likelihood Linear Regression and Maximum A Posteriori adaptation) have been used to improve the performance of a system trained using multiple writer data over the words written by a specific person.
The recognition has been extended from single word to text line recognition. In this case, the only knowledge we have about the data to be recognized is that they are written in English and they must reproduce on average the statistics of any english text. Such hypothesis is used by applying Statistical Language Models (unigrams, bigrams and trigrams). Their application results in a significant improvement of the system.
All the performed experiments were designed to be correct from a statistical point of view, to avoid any fitting of the system to the test set and to give realistic measures of the performance.
The rest of this thesis is organzed as follows: this chapter gives an extensive survey of the state of the art in the domain, chapter 3 describes the recognition system we developed and used in our experiments, chapter 4 presents the data transforms we applied and the results of their application, chapter 5 reports about the use of HMM adaptation techniques, chapter 6 shows how the recognition was extended from single words to text line recognition and chapter 7 draws some conclusions and delineates possible future works.

2.2 State of the Art

Off-Line Cursive Word Recognition (CWR) is the transcription into an electronic format of cursive handwritten data. The main development of the field took place in the last decade [120][136] and some commercial products, based on CWR, are yet running in real world applications [36][60]. The recognition is often based not only on the handwritten data, but also on other information coming from the application environment. This made CWR technology effective only in few domains, indeed postal address reading (where the recognition of the zip code plays an important role) and bank check legal amount recognition (where the courtesy amount, i.e. the amount written in digits, helps the recognition of the legal amount, i.e. the amount written in letters). Many issues are then still open and the problem of the general CWR is still far from being solved.
Several aspects of the recognition process are however independent of the application domain and can be considered in a general framework. For this reason, this chapter is divided into two parts. The first one concerns the problems a CWR system must deal with. Each step of the processing is described in detail and the main techniques developed to perform it are shown. The second one focuses on applications presented in the literature and their performances.
The first part is composed of section 2.3, where the structure of a CWR system is outlined and the single processing steps are described in detail, the second part of section 2.4, where the main application domains of CWR are illustrated. In section 2.5, some conclusions are drawn. This chapter corresponds to reference [139].

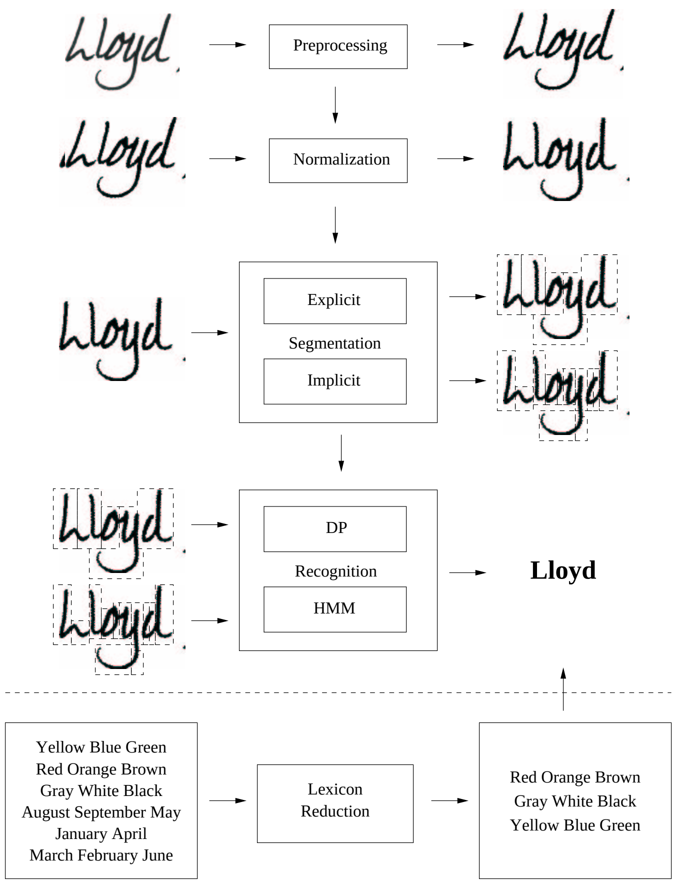


Figure 2.1: General Model of a CWR system.

2.3 Structure of a CWR System

The basic structure of a CWR system is shown in figure 2.1, the only exception to such architecture is given by the human reading inspired systems (see section 2.3.7) and the holistic approaches (see section 2.3.8). Some of the tasks performed in the recognition process are independent of the approach (e.g the preprocessing). Others are related to it and can be used to discriminate among different systems (e.g. the segmentation).
Usually, the raw data cannot be processed directly and the word images must be preprocessed in order to achieve a form suitable for the recognition, this is the aim of the *preprocessing*. The operations performed at this level depend on the data. The removal of background textures, rulers and similar elements is often needed when the word is extracted from forms or checks; a binarization is useful when the words are stored in gray level images. In general terms, the result of the preprocessing must be an image containing the word to be recognized without any other disturbing element.
The following step is the *normalization*. Slant and slope different than 0 (see figure 2.2) can be caused by acquisition and handwriting style. Their removal results in a word image invariant with respect to such factors, hence the name normalization.
The first two steps are independent of the recognition approach of the system. The same preprocessing and normalization algorithms can be shared by systems using different recognition approaches. This is no longer true for the *segmentation*, which depends on whether the system uses Dynamic Programming or HMMs to perform the recognition. In the first case, the segmentation is said to be *explicit*, i.e.

an attempt is made to isolate the single letters that are then separately recognized. In the second case, the segmentation is *implicit*, i.e. the word is fragmented into subletters and the only constraint to be respected is the oversegmentation: the word must be split at least in correspondence of the actual ligatures between characters. In other terms, each subunit of the word, must belong to only one character. In correspondence to the two alternatives, figure 2.1 shows two paths.
The *recognition* step uses the word fragments isolated by the segmentation to estimate a score for each element of the lexicon. The best scoring lexicon word is assumed as the interpretation of the handwritten data. Before the recognition, the fragments are converted into vectors through a *feature extraction process.*
A fundamental element in a CWR system is the *lexicon*, a list of the allowed interpretations of the handwritten data. Intuitively, by reducing the size of the lexicon, the accuracy of a CWR system can be improved since the probability of misclassification is reduced. A first, most important, limit to the lexicon size is given by the application environment (e.g., when recognizing legal amounts on bank checks, the only allowed transcriptions are numbers written in letters). A further lexicon reduction can then be achieved by analyzing the handwritten data itself and by extracting from the lexicon all the incompatible interpretations (if a handwritten word does not present ascenders or descenders, only transcriptions composed of letters without ascenders or descenders can be accepted). In the following sections, each step of the processing will be described in detail.

2.3.1 Normalization

In an ideal model of handwriting, a word is supposed to be written horizontally and with ascenders and descenders aligned along the vertical direction. In real data, such conditions are rarely respected. *Slope* (the angle between the horizontal direction and the direction of the implicit line on which the word is aligned) and *slant* (the angle between the vertical direction and the direction of the strokes supposed to be vertical) are often different than 0 and must then be eliminated (see figure 2.2).
The normalized images are invariant with respect to the sources of slant and slope (acquisition and handwriting style) and this is helpful to the recognition process. In Dynamic Programming based systems, the removal of slant and slope makes the characters less variable in shape and thus easier to be classified with pattern recognition techniques. Besides, the normalization creates segments where the handwritten data is piece-wise stationary, whose presence is a necessary assumption to the use of HMMs.
In the following two paragraphs, methods for removing respectively slope and slant are described. Some attempt to use the information lost after the normalization in the recognition step is described in section 2.3.1.

Slope correction and reference line finding

Most of the desloping techniques presented in the literature are inspired by the method proposed in [16]. This consists in giving a first, rough estimate of the *core region* (the region enclosing the character bodies), then in using the stroke minima closest to its lower limit to fit the ideal line on which the word is aligned. The image is rotated until such line is horizontal and the image is finally desloped (see figure 2.2).
The estimation of the core region, the fundamental step, is made by finding the lines with the highest horizontal density (number of foreground pixels per line). The core region lines are in fact expected to be more dense than the others. The horizontal density histogram is analyzed by searching for for features such as maxima and first derivative peaks, but these features are very sensitive to local characteristics and many heuristic rules are needed to find the actual core region lines [113].
Some alternative techniques were proposed in [34] and [145]. In such works, the density distribution is analyzed rather than the density histogram itself in order to make statistically negligible the influence of local strokes. The method presented in [34] is based on the entropy of the distribution (supposed to be lower when the word is desloped), while the technique in [145] applies the Otsu Method [66] in

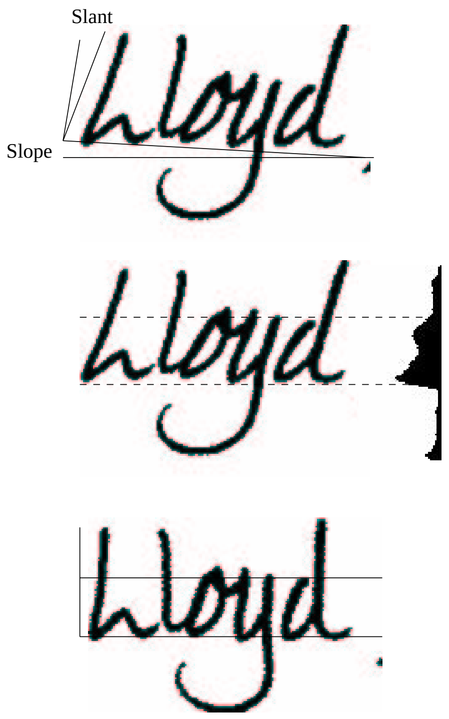


Figure 2.2: Preprocessing. The word, before preprocessing, shows slope and slant different than 0. In the horizontal density histogram, the lines corresponding to the core region are evident. The long horizontal stroke in the lower part of the y creates a high density area that can be erroneously assumed as the core region. After the preprocessing, the word appears horizontal with ascenders and descenders aligned along the vertical axis.

order to find a threshold distinguishing between core region lines (above the threshold) and other lines. In [19], the image is rotated for each angle in an interval and the rotated image giving the highest peak of the first derivative of the horizontal density histogram is assumed as desloped. Another important result of the desloping is the detection of the limits of the core region, called the *upper* and *lower baseline*, that play an important role as reference lines.

Slant correction

Most of the methods for slant correction are also based on the technique proposed in [16]. This relies on the selection of near vertical strokes the slope of which is assumed as local slant estimate. The global slant value is obtained by averaging over all the local estimates. Techniques based on such idea can be found in [19][49][84][131], each work using a different method to select the strokes involved in the global slant estimation. A method avoiding the selection of specific strokes can be found in [44], where all the points on the border are used to calculate the most represented directions.
A different approach was proposed in [82][145]: a measure of the “deslantedness” is performed over all the shear transforms of the word image corresponding to the angles in a reasonable interval. The transformed image giving the highest “deslantedness” value is the deslanted one.

Use of the writing style as a source of information

The normalization step, by eliminating characteristics introduced by the writing style, gives the handwritten words a standard form, but, in the meantime, destroys some information.

In some works, the possibility of using the writing style as a source of information helpful to the recognition process has been proposed. Experiments were performed on several approaches to group the handwriting styles into families.
In [35], stroke width, number of strokes per unit length in the core region, core region position, and the histogram of the quantized directions of generic strokes are used as features to characterize the writing style. Such features are selected because they are not related to any character in particular, so they do not depend on the word they are extracted from.
Fractal dimension related measures have been proposed for the same purpose in [13][53]. The fractal dimension is shown to be a very stable parameter for a writer even in samples produced in different years.
The features proposed are efficient in grouping the styles into well defined families, but no results were presented in terms of recognition rate improvement.

2.3.2 The segmentation

The segmentation of an image is performed by connecting, or identifying maximal connected sets of pixels participating in the same spatial event [66]. In CWR terms, this means to isolate fragments in the handwritten word supposed to be the basic information units for the recognition process. As pointed out in section 2.3, the segmentation can be explicit or implicit depending on whether the isolated primitives are expected to be characters or not.
The explicit segmentation is a difficult and error prone process because of Sayre's Paradox [128]: a letter cannot be segmented before having been recognized and cannot be recognized before having been segmented. Until now, no methods have been developed that are able to segment handwritten words exactly into letters [21][97].
On the contrary, implicit segmentation is easy to achieve because the only constraint to be respected is the oversegmentation (see section 2.3). The number of spurious cuts (points where the word is split even if does not correspond to actual ligatures between characters) needs not to be limited.
In principle, the segmentation is independent of the recognition technique, but the explicit segmentation is mostly performed in Dynamic Programming based systems, while implicit segmentation is used in architectures involving Hidden Markov Models. For this reason the segmentation was used elsewhere as a key component in distinguishing among different approaches [136]. This is, in our opinion, not completely correct because if the choice of the segmentation was free, the implicit segmentation, which is more easily performed, would be always preferred. The real problem is that when applying the Dynamic Programming, the fragments extracted from the word are supposed to be characters and only small variations with respect to this condition can be tolerated. The segmentation must then be as explicit as possible and spurious ligatures must be minimized.
HMMs are not only able to work on a sequence of fragments not necessarily corresponding to letters, but can also face with variations and noise that can occur in the sequence itself. This allows the use of an implicit segmentation.

2.3.3 Feature extraction

The features can be grouped into three classes depending on whether they are extracted from the whole word (high level features), the letters (medium level features) or subletters (low level features). In the next three subsections, each feature class is described in more detail.

Low level features

Low level features are extracted from letter fragments that have elementary shapes such as small lines, curved strokes, bars and similars. The features account, in general, for their position and simple geometric characteristics.
It is frequent to use features that describe the distribution of pixels with respect to reference lines:

the percentages of the stroke edges in core, ascenders and descenders regions are used in [18][17]. The distances of the foreground-background transitions from the median line of the core region are proposed in [55][111]. The percentages of foreground pixels in core, ascenders and descenders region are applied in [23][89].
To have an overall description of the shape, features like curvature [18][17], center of mass [89] and histogram of the strokes directions [89][131][130] are used.
In several works, the small strokes are considered deformations of the elements of a basic set of strokes and the deformation itself is used as a feature. In [16][47][117], the set of basic elements is composed of different curves or linear strokes (e.g. curves up, down, left or right directed).

Medium level features

Systems based on explicit segmentation must face with the recogniton of cursive characters. The biggest problem is the variability of their shapes [20], thus features performing an averaging over local regions are preferred. A normalized image of the character is used in [150]. Background-foreground transition distribution is used in [55]. In [56], bar features are proposed. In [118], the feature extraction is performed over trigrams and consists of a vectorization of the contour. The systems involving the character recognition must also cope with primitive aggregates that are not characters. To distinguish between actual letters and non-letters, a method is proposed in [55]: the presence of too many ascenders or descenders in a letter candidate is used as a rejection criterion.

High level features

Features such as loops, ascenders and descenders are often referred to as high level features. Since they consist of the detection of structural elements, they do not depend on the writing style and are then stable with respect to cursive variability. Together with loops, ascenders and descenders (the most used since they are easily detected), we also find junctions, stroke endpoints, t-bars and dots in the literature. In some works [18][131][130], high level features are extracted from the word skeleton (a representation of the word that allows an easy detection and ordering of structure elements).
In some cases [34][49], mainly in applications involving small lexica (such as bank check reading), the high level features are used to give a rough representation of the word. This allows one to discard part of the lexicon or to reject a result of the recognition process whose representation is not compatible with the detected one (see figure 2.6).

2.3.4 Lexicon reduction

The size of the lexicon is one of the most important parameters in CWR. As the number of allowed intepretations increases, the probability of misclassification becomes higher. For this reason several Lexicon Reduction Systems (LRS) were developed. In some cases they are based on other information than the word to be recognized (e.g. the zip code, in postal applications, limits the number of allowed transcriptions of a handwritten town name). In other cases they use the handwritten word itself to discard some interpretations from the lexicon. In this section, the attention will be focused on this latter category of LRS.
In general terms, an LRS takes as input a lexicon and a handwritten word and gives as output a subset of the entries of the lexicon. An optimal trade-off must be found between the compression rate of the lexicon and the number of times that the correct transcription is not discarded.
The LRS's are always based on a rough representation of the handwritten data that allows them to rank (depending on a compatibility score) the lexicon entries or discard those which are incompatible with the data. In [118], a system based on the detection of trigrams is shown to put the correct intepretation in the top 200 positions of a rank generated using a 16200 word lexicon with an accuracy of 96%. This corresponds to a compression rate of 1.2%. In [101], the handwritten word is converted into a string of characters representing structural elements (e.g. stroke extrema). For each entry of the lexicon, an ideal model (represented by a string) is given and the compatibility between lexicon

words and handwritten data is measured with edit distance. Starting from a lexicon of around 21000 words, an average compression rate of ~33% is achieved with an accuracy higher than 99%. The LRS described in [152] relies on the concept of key characters. These are the letters that are most easily recognized and are used to find the most compatible entries of the lexicon. The average reduction rate achieved is 72.9% with an accuracy of 98.6%.

2.3.5 The data

The data can be considered not only as an input to a CWR system, but as an actual part of it. The data to be recognized is not a simple collection of words without any relationship between them, but a sample of the data produced in some human activity. The nature of such activity creates conditions that influence the solution of the recognition problem. Changing the data means changing the problem: variations in lexicon size and number of writers significantly affects the performance.
Several databases [67][106][131][138] are available. Each one is related to some application, e.g the CEDAR database [67] is composed of postal material and allows the simulation of a postal plant. The use of the same data by many researchers allows a comparison of the results achieved, but the literature presented most often works showing results obtained over data used only by their authors. A solution to this problem has been proposed in [3], where the human performance is indicated as an absolute term of comparison. The same data used to test a system (or a representative subset of it) should be transcribed by a human reader. The performance of the human should be considered as the best result achievable over the data.

2.3.6 The recognition

The recognition consists of finding, among the words of the lexicon, the most compatible one with the handwritten data. The two main techniques for this task are Dynamic Programming and Hidden Markov Modeling that will be described in subsection 2.3.6 and 2.3.6 respectively.
These methods use the feature vectors extracted from the word fragments to estimate a score giving a measure of the matching of the handwritten data with a possible interpretation. The score can be a probability, a distance or a cost depending on the specific case. The best scoring word of the lexicon is assumed to be the correct interpretation of the data.

Dynamic programming

Dynamic programming [9][10] is often used after explicit segmentation to match a sequence of M letters with a sequence of N word fragments. This is defined in [122] as a synchronous sequential decison problem and its solution corresponds to finding the optimal cost path in the trellis of figure 2.3.
After having matched the first $m < M$ letters with the first $i < N$ word fragments, the following step can reach any further primitive n (where $n > i$ since the words are written from left to right). The optimal step, associating the first $m+1$ letters with the first n primitives, will be found as follows:

$$\phi_{m+1}(i,n) = min_l[\phi_m(i,l) + \zeta(l,n)] \tag{2.1}$$

where $\phi()$ represents the optimal cost and $\zeta()$ the cost to associate the $(m+1)^{th}$ letter to the aggregation composed by primitives $i+1, i+2, \ldots, n$. Equation 2.1 implies that any partial consecutive sequence of moves of the optimal sequence from i to n must also be optimal, and that any intermediate point must be the optimal point linking the optimal partial sequences before and after that point. This is the way the Bellmann's optimality principle [1] is put into a functional equation. Usually a maximum number k of primitives allowed to form a single character is set, then $n \in [i+1, i+k]$.
The cost is estimated in several ways: sometimes as a distance from letter prototypes (the optimal path will then be a minimum cost path) and sometimes as a probability for the aggregations of being

[1]An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the final decision.

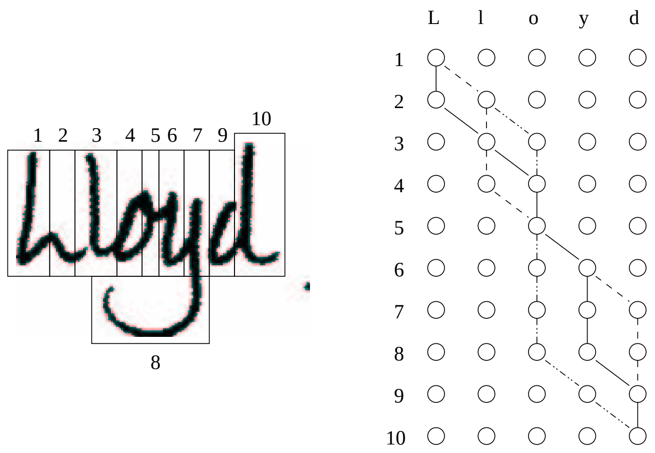


Figure 2.3: Recognition in systems based on segmentation and dynamic programming. The matching of the word with the sequence of primitives is found as the optimal path through the trellis.

a certain letter (the optimal path will then be the maximum cost path and, in equation 2.1, *min* will be replaced by *max*).
A problem in using DP was pointed out in [25][26]: since the cost is estimated through recognizers trained at the character rather than at the word level, then an improvement of the character recognizer does not lead necessarily to an improvement of the overall system. This makes it very difficult to control the system performance.

Hidden Markov Models

For a good introduction to Hidden Markov Models, see [11][14][122] and for their specific use in handwriting recognition see [73].
HMM based recognition systems convert the data images into a sequence of vectors. The recognition of such a kind of data requires simultaneous modeling of inherent statistical variations in sequential rates and feature characteristics. So far, HMMs are one of the most successful approaches dealing with such a problem. The sequence of vectors is assumed to be piecewise stationary, so that it can be modeled as a sequence of discrete stationary states $Q = \{q_1, q_2, \dots, q_T\}$, where $q_t \in (1, 2, \dots, S)$ belongs to a finite set of possible states. The sequence Q is *hidden* (hence the name) and cannot be observed directly. Each vector is assumed to be emitted following a distribution characteristic of the underlying state (see figure 2.4). The HMMs thus model a double embedded stochastic process.
The probability of a transition to state q_t at step t depends in principle on the state of the last d steps, but, in general, the assumption $d = 1$ is made and the transition probability can be expressed as:

$$p(q_t|q_{t-1}) = p(q_t|q_{t-1}, q_{t-2}, \dots, q_{t-d}). \tag{2.2}$$

The model is then called a first order HMM. A second important assumption is that the transition probability does not depend on t and is then stationary. The evolution dynamic of the system is entirely encoded by the matrix $A = \{a_{ij}\} = \{p(q_t = j|q_{t-1} = i)\}$ (see figure 2.4). To start the process, an initial state probability π_i (probability of being in state i at first step) must also be defined. The vector $\pi = \{\pi_1, \pi_2, \dots, \pi_S\}$ represents the initial state probability distribution.
When the system is in state j, an observation vector is emitted following the probability density function $b_j(\mathbf{o})$ defined in the space of the observation vectors. The b_j are continuous or discrete depending on the nature of the observations. The set $B = \{b_1(\mathbf{o}), b_2(\mathbf{o}), \dots, b_N(\mathbf{o})\}$ is the bridge between the hidden states and the observations.
A Hidden Markov Model λ is represented by the set: $\{A, B, \pi\}$. The estimation of the parameters

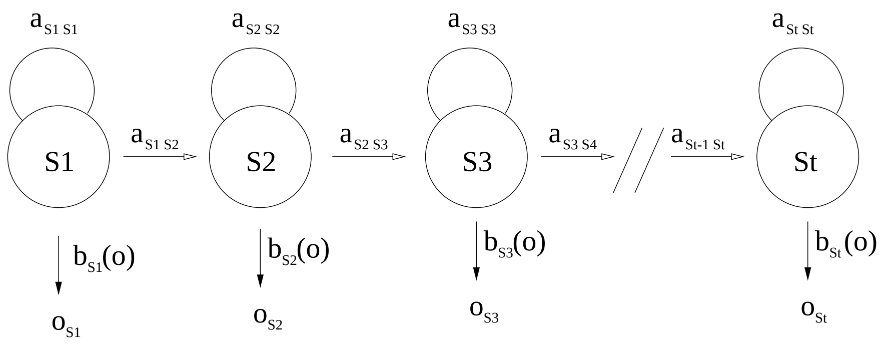


Figure 2.4: Hidden Markov Model. The sequence of the states is produced by the transition probabilities a_{ij}. At each state is associated an emission probability $b_j(\mathbf{o})$.

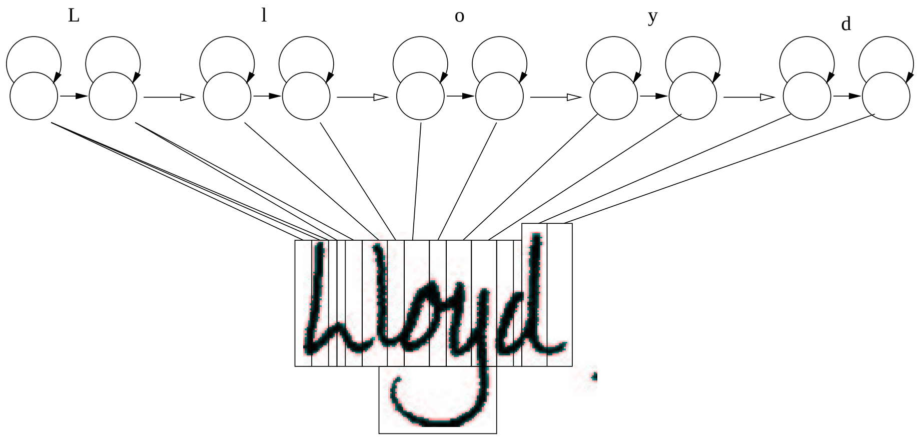


Figure 2.5: Recognition in systems based on HMMs. The statistical nature of HMMs allows them to cope with the cursive variability.

in A,B and π (the training), is performed with the Baum-Welch Algorithm, a particular form of the EM algorithm that train the models according to the Maximum Likelihood criterion [4][7][8][5][6]. A different HMM is typically built for each letter. The word models are obtained through concatenation. This makes the system more flexible with respect to a change of lexicon because it is not necessary to have samples of the words to recognize in the training set (a condition that can be difficult to achieve), but simply samples of the letters (a condition easy to achieve if enough words are available). However, the training algorithm is applied to word rather than single letter models. This is called *embedded* training and has a twofold advantage: the first one is that the letters are modeled as a part of a word as they actually are in cursive handwriting. The second one is that it is not necessary to segment the training data into letters (such a process can be expensive and time consuming).
Once the models are trained, it is possible to perform the recognition using the Viterbi algorithm [147]. This gives the best likelihood L that can be obtained by following a unique state sequence with the vectors extracted from the handwritten data. The state sequence giving the highest value of L is selected as transcription of the handwritten data.

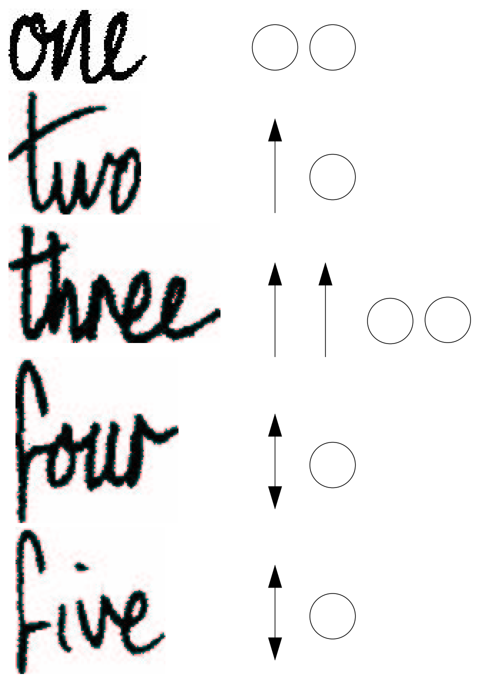


Figure 2.6: A representation of handwritten numbers between one and five in terms of global features. The representation allows us to discriminate one, two and three, but there is an ambiguity between four and five.

2.3.7 Human Reading Inspired Systems

Several human reading models were elaborated by cognitive scientists[29][74] [110][116]. None of them is able to account for every aspect of the *lexical access*, i.e. the process of giving meaning to a handwritten word, but some of them explain effects that can be useful in developing a CWR system. In particular, the McClelland and Rumelhart model [110] shows how the *Word Superiority Effect* (the fact that a letter is easier to be recognized when embedded in a word than when isolated) emerges.

The Word Superiority Effect can be considered as the influence of the word context in single letter recognition. This has been proposed also in works not inspired by psychological models, but the influence of the context had to be modeled directly while in the McClelland and Rumelhart model it is an effect of the global architecture.

The model is formed by three levels of processing (corresponding to different levels of abstraction) working in parallel: feature level, letter level and word level. The levels interact in that the results obtained at a certain level are used by the process at another level. The interaction is due to inhibitory or excitatory connections between levels. The feature level tends to excitate the letters corresponding to the features detected and to inhibit the others (e.g. a loop excitates the presence of the letter *o*, but inhibits the presence of the letter *s*). The letter level excitates the connections with the word level linking words that contain the letter detected. Viceversa the word level excitates the letters that are contained in excitated words, but were not detected at the letter level. The inhibitory connections work in a similar way and tend to decrease the excitation of words or letters that are not compatible with each other.

By using such a model in CWR systems, the influence of context is not modeled (e.g. as in [55]), but it is an effect of the global architecture.

Several works were inspired by this model [32][34][33][31][45][64][62][118]. In general, they present a

recognition cycle divided into two parts: a bottom up and a top down process. The bottom up process consists of extracting global features such as ascenders, descenders and loops. These kind of features are not expected to give the identity of a letter, but only to account for the global shape of the word. This representation is used to create a short list of candidates among the lexicon elements by selecting only the words with a compatible shape. (e.g. the presence of an ascender will allow us to discard all the words of the lexicon that do not contain letters with ascenders, see figure 2.6). In some cases, the short list contains only one word and the recognition process is completed, in other cases several words are selected and the top down process is needed. This consists of a verification, i.e. looking for the letters that compose each candidate of the short list. The global features, previously extracted, are used as anchor points and help in indicating where to look for a certain letter. The presence of a letter is verified by looking for local features referred to sometimes as secondary or conditional features in the sense that they are used only in conjunction with the primary ones when these are not sufficient.
This approach does not need the recognition of all the letters of the word, in some cases (when the short list contains only one word) it is not necessary to recognize any letter. This technique is based more on the discrimination between the words of the lexicon than on their recognition. Because of this, it is effective only in applications that involve a small ($25 \sim 30$ words) and static lexicon.

2.3.8 Holistic approaches

Some researchers explored the possibility of recognizing the cursive words with an holistic approach. This consists of considering only the global characteristics of the word to perform its recognition. In [126][112], the word is converted (through a global feature extraction) into a string that is compared with the words of the lexicon with edit distance or Levehnstein metric. This approach is effective with a small lexicon, when a rough representation of the word is enough to identify the correct transcription among the lexicon entries. When the lexicon size increases, too many words are compatible with the same word representation and a correct classification becomes difficult.
For this reason, the holistic approach is often used not to recognize the words, but rather to support other tasks. In [101][118], the use of holistic features is shown to be very effective in lexicon reduction (see section 2.3.4). In [27][98][100], the use of an holistic representation is used in transcribing handwritten postal addresses: first the system recognizes the zip code, then verifies (using an holistic representation of the word) whether the town and state names written on the envelop correspond to it or not. This is shown to prevent several errors in the final destination attribution of the mail piece. In [60][88], the holistic information is used in conjunction with features at lower level for the recognition of courtesy amouts on checks.

2.4 Applications

In the next subsections, the main applications of CWR will be presented. Bank check reading is described in subsection 2.4.1, postal applications are illustrated in subsection 2.4.2 and generic content document recognition is presented in subsection 2.4.3.

2.4.1 Bank check reading

Many works about CWR are dedicated to the bank check legal amount recognition. The developed systems are good enough to be used in commercial products as described in [60], where a family of systems able to work on french, english and american checks is claimed to have a performance close to a human reader (rejecting 30-40% of the data).
The reading of legal amounts involves small lexica (between 25 and 30 words) and each word to be recognized is produced by a different writer. An important advantage in bank check reading is the presence of the courtesy amount (the amount written in digits). This can be read reliably, but it is not relevant from a legal point of view, so an automatic check processing system must read also

Authors	DB size	performance
Côté et al. [34]	3113	74.0%
Guillevic et al. [64]	7837	71.8%
Dodel et al. [45]	1150	70%
Olivier et al. [115]	6000	69.6%
Knerr et al. [89]	189659	89.2%
Saon [127]	100000	82.5%
Kaufmann et al. [81]	13000	71.9%

Table 2.1: Performances in bank check reading. For each work cited in the first column, the table reports the size of the database used in the experiments and the best performance (in terms of words correctly recognized) obtained. The lexicon size is between 25 and 30 depending on the language of the legal amounts processed. The double horizontal line separates the results obtained with human reading model inspired systems (upper part) from that obtained with HMM based systems (lower part).

the amount written in letters. On the other hand, the redundancy of information when reading both courtesy and legal amount can improve the performance of the system.
In [89][115][117], an implicit segmentation is applied, and the recognition is performed with an Hidden Markov Model.
A segmentation free approach is proposed in [65][105][148], where a sliding window is applied to the amount image and a recurrent neural network is used in conjunction with HMMs. The sliding window is also used in [81], where a correction mechanism activated by a mismatch between courtesy and legal amount is proposed. In [127], the scaled images of the legal amount are considered as random field realizations and recognized in conjunction by HMM and Markov Random Fields [28]. A combination of methods based on analytic and global features was presented in [39][43][42][41]. This approach is especially conceived to work on italian amounts: these are more difficult to recognize because they are obtained by joining several basic words.
Bank check reading, due to the small lexicon, is the only domain where the human reading inspired systems were found to be effective [34][45][62][64].
The results obtained in terms of percentage of words correctly recognized in several works are shown in table 2.1. The double horizontal line separates the results obtained with methods inspired by psychological models and those obtained with other techniques.
In [60], the human performance is said to be around 99%. This rate can be achieved by current automatic readers only by discarding the more ambiguous samples.

2.4.2 Postal applications

Most of the works in the literature concern postal applications. The data involved in this domain are completely unconstrained; each word is written by a different writer, the words can be cursive, handprinted or a mix of the two styles. The lexicon depends, in general, on the output of a zip code recognizer. When the zip code is recognized, it is not even necessary to read further information in the address. When there is unacceptable ambiguity on the last, last two, or last three digits of the zip code, then it is necessary to read the town name and the lexicon will contain ten, one hundred or thousand words respectively.
Several works are based on segmentation and dynamic programming [25][50]
[111][132]. In [111], the performance is improved by using, together with the segmentation based system, a segmentation free system based on HMMs. The combination of two different approaches (lexicon free and lexicon directed) is also described in [132][135]. Techniques to calculate the score of a lexicon word, given the single character confidences are proposed in [25] and [50].

Authors	DB size	performance
Chen et al. [25]	996	90.8%
Gader et al. [55]	1000	85.8%
Kim et al. [84]	3000	88.2%
Mohamed et al. [111]	10567	89.3%
Kundu et al. [90]	3000	88.3%
Chen et al. [23]	1583	72.3%
El Yacoubi et al. [49]	19447	96.3%

Table 2.2: Performances in postal applications. For each work cited in the first column, the database size and the performance (in terms of percentage of words correctly recognized) for a 100-word lexicon is reported. The double horizontal line separates the results obtained with HMM based systems (lower part) from that obtained with explicit segmentation and dynamic programming based systems (upper part).

Authors	DB size	lexicon size	performance
Bozinovic et al. [16]	130	710	77%
Bunke et al. [18]	3000	150	99.3%
Senior et al. [131]	4053	1334	92.8%
Marti et al. [108]	50754	7719	60.0%

Table 2.3: Performances in generic content document recognition. For each work cited in the first column, the DB size, the lexicon size and the performance (in terms of percentage of correctly recognized words) are collected.

A system based on HMMs is presented in [23], where a modified Viterbi algorithm is described. In [49], after having performed an explicit segmentation, the system uses an HMM based technique to combine two feature sets: the first oriented to characters, the second to ligatures. The segmentation statistics (the probability of segmenting each letter into n primitives) are taken into account during the recognition process in [22][84][83][90]. A Minimum Edit Distance modified to better represent the errors that can occur in a CWR is described in [129]. In [48][100][99][119], the possibility of reading handwritten lines is investigated to recognize different forms assumed by the same address (e.g. *Park Avenue* or *Park Av.*).
A direct comparison among the systems is impossible because the results are obtained over different databases. However all the databases consist of real world data obtained by digitizing handwritten addresses. The performances presented in the literature are collected in table 2.2, where the double line separates the segmentation based works (upper part) from the HMM based systems (lower part). The performances are reported for lexicon size 100.

2.4.3 Generic Content Recognition

In the previous sections, systems concerned with specific applications were presented. The application environment involving the system is a source of information that have a strong influence on the recognition process. In the works presented in this section, the recognition was performed over data that did not allow the use of any other information than the handwritten words themselves. At most, if the words belong to a text, the linguistic knowledge would be introduced. The data used in the works related to this subfield of CWR is often created ad hoc by asking writers (in some case cooperatives) to produce samples.
In [16][47] the words produced by few writers are recognized. Both works are based on explicit seg-

mentation and use different level representations of the words that allow making hypotheses about the transcription and looking for its confirmation at the feature level. In [47], the confirmation is obtained as a degree of alignment of letter prototypes with the actual handwritten data. In [16] the confirmation is given by matching the sequence of expected events (e.g. loops, curve strokes of various shape, etc.) with the actual sequence detected in the handwritten word.
In [149][150], a word is segmented explicitly first and then an HMM is used to find the best match between the fragments (the similarity with character prototypes is used to calculate the probability of a fragment beeing a certain letter) and words in the lexicon. In [18][17], the words written by cooperative writers are represented, after a skeletonization of the word as a sequence of strokes organized in a graph. Each stroke is represented by a feature vector and their sequence is recognized by an HMM.
The first example of recognition of data extracted from a text (to our knowledge) is presented in [131][130]. The selection of the text is addressed by linguistic criteria, the text is extracted from a corpus supposed to be representative of current english. This allows the use of linguistic knowledge in recognition.
The data is produced by a single writer, so that an adaptation to his/her writing style can play a role in improving the system performance. In [131][130], the words are skeletonized and then a uniform framing is performed. From each frame a feature vector is extracted and an HMM is used for the recognition. A recurrent neural network is used to calculate the emission probabilities of the HMMs.
The use of linguistic knowledge was shown to be effective in recognizing whole sentences rather than single words in [104][106][108][107]. The applied language models are based on statistic distributions of unigrams and bigrams [77]. The use of syntactical constraints (expressed by transition probabilities between different syntactical categories) was experimented in [133][134].
The results achieved in the works presented in this section are obtained over different data sets, not only because they contain different data, but also because they were created with different aims and criteria. The performances of some systems dealing with generic words are reported in table 2.3.

2.5 Conclusions

This chapter presented a survey on Off-Line Cursive Word Recognition. A description of the major approaches was given as well as an overview of the applications presented in the literature.
The importance of the information brought by the application environment is highlighted by showing that most of the works on CWR concern two specific tasks: bank check and postal address reading. In both cases, the handwritten data to be recognized is accompanied by other information (courtesy amount and zip code, respectively) improving the performance of the system. In such domains, the accuracy achieved allowed the use of the developed systems in real world applications. There are, however, many other practical problems involving handwritten information that can take an advantage from the use of a CWR system. This leads the research efforts toward systems able to recognize data as generic as possible so that an eventual adaptation to a specific task can be quickly performed. The linguistic knowledge is investigated as a solution for the lack of other information than the handwritten data to be recognized.
Independently of specific data or applications, two approaches were shown to be mainly used: the first one is based on the explicit segmentation followed by Dynamic Programming, the second one relies on implicit segmentation and Hidden Markov Models. This latter technique was most frequently used in the recent years. The strong theoretic framework provided by HMMs limits the amount of heuristics needed to improve the system performance.
The human reading inspired architectures were also described and their inability to deal with lexica bigger than $25 \sim 30$ words was shown to limit their use to bank check recognition.
A list of available databases was presented. The data is not just an input in a CWR system. The influence of the nature of the data on the recognition process was shown. Furthermore, the availability of databases that can be shared by several research groups helps in making comparisons between different techniques.

The performances of the systems presented in the literature were finally reported, showing in some cases recognition rates sufficient for real world applications [36][60]. Off-Line CWR is a still evolving field. The important results related to specific application domains cannot be considered conclusive. The open issues to achieve a general Cursive Word Recognition are still many and important. The next chapters illustrate the development of a system aimed to the recognition of unconstrained english texts. Issues left open in the literature are addressed: algorithms minimizing the use of heuristics are used for preprocessing and normalizing the data, a sliding window approach is used to avoid segmentation and related problems, the modeling is improved by applying data transforms and HMM adaptation techniques, and the recognition is extended from single words to texts using language models.

Chapter 3

The recognition system

3.1 Introduction

The previous chapter described the common structure of a cursive recognizer and gave an overview of the solutions offered by the literature for each step of the processing. Most systems try to deal with the variability of cursive handwriting (the main problem for a recognizer) by implicitly trying to model and take into account all the variations due to different handwriting styles.
This is for example what happens when using feature sets that account for small details of the handwritten strokes. They are sensitive to noise and enhance rather than smooth the handwriting variability. In a similar way, when a deslanting algorithm is based on the detection of vertical or quasi-vertical strokes of the word (as in most systems), small variations in the slant of a single stroke can lead to completely different results in the deslanting.
In the development of our system, we followed a different strategy. We made general assumptions about the data and we based our algorithms on them. We took into account only high level characteristics of the words. Some parts of the handwritten words were even discarded because they are believed not to be useful for recognition.
The result is a system that involves a very small amount of heuristics, only two parameters to be set empirically (this limits the experimental effort needed to find an optimal configuration of the system) and that it is robust and flexible with respect to a change of data.
This can be seen in an experiment described at the end of this chapter (originally published in [145]) where our deslanting algorithm is compared with another technique for the same task [16]. The results show not only that our method improves the recognition rate, but also that it adapts automatically to different data sets. It is not necessary to explore the parameter space each time the data changes as happens for the other algorithm.
The system is based on the sliding window approach: a fixed width window shifts column by column from left to right and, at each position, a feature vector is extracted from the isolated frame. The sequence of vectors so obtained is modeled with continous density Hidden Markov Models.
In the rest of the chapter the single steps of the processing are shown in detail. Section 3.2 presents the normalization technique, section 3.3 describes the sliding window approach and the feature extraction, section 3.4 introduces the modeling technique and section 3.5 shows some experiments and results. The final section 3.6 draws some conclusions. The content of this chapter corresponds to references [146][145].

3.2 Normalization

The role of normalization is the removal of slant and slope, two effects due to acquisition and handwriting style (section 2.3.1). The technique we propose is based on two assumptions: the first one is

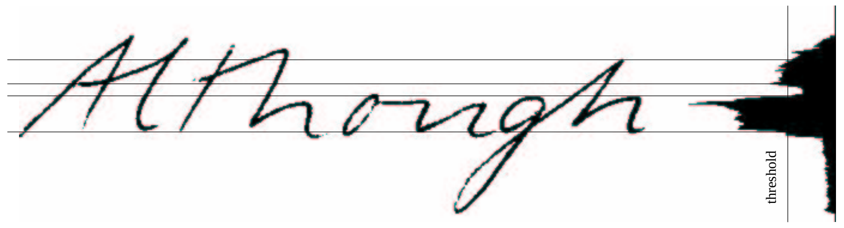


Figure 3.1: The proposed core region detection method: the density threshold gives a first rough estimate of the core region. A high density region is located in the ascender area, but can be discarded by simply selecting the high density region containing the highest number of foreground pixels.

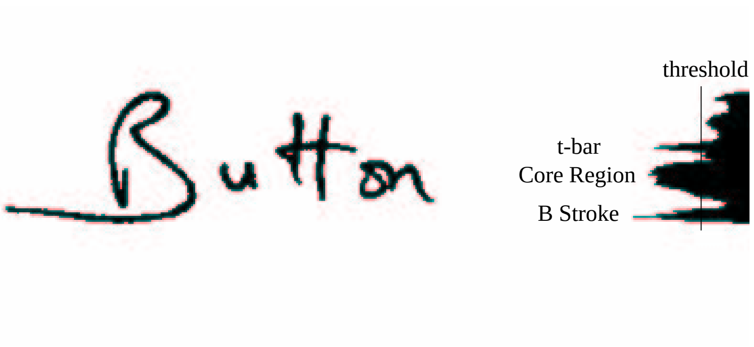
that the words present a core region where the horizontal density is higher than in the ascender and descender regions. The second one is that, when a word is deslanted (i.e. all the strokes supposed to be vertical are actually vertical) the number of columns containing a continuous stroke is maximum. These two assumptions are very general and apply to any handwritten word. Following the strategy mentioned in section 3.1, we avoided taking into account any further hypotheses in order to avoid being sensitive to noise. The result is a normalization step that is shown to improve the recognition rate with respect to other, more widely applied, techniques (see section 3.5).
The next two sections 3.2.1 and 3.2.2 describe our deslope and deslant algorithms respectively. Section 3.2.3 describes the Bozinovic-Srihari deslanting method that will be compared (in terms of recognition rate) with our deslanting algorithm.

3.2.1 Slope Removal

The traditional methods for slope removal [16][131] start by finding a first rough estimate of the *core region*, i.e. the region enclosing the character bodies (see figure 2.2). This estimate is biased by the fact that the word is not horizontal, so that the upper and lower limits of the estimated core region do not fit, as they should, the extrema of the character bodies. To solve this problem, the stroke minima closer to the lower limit of the estimated core region are used to fit (e.g. with the Least Squares Method) the line connecting the bottom points of the character bodies. This is the line (called the *lower base line*) on which the word is implicitly aligned. The image is desloped when this line is horizontal, a condition achieved through a rotation transform.

The first estimate of the core region is fundamental for removing the slope. The detection of the core region is based on the hypothesis that the density of the lines belonging to it is higher than the density of the other lines. This is evident in the horizontal density histogram (see figure 3.1), showing higher values in corresponding to the core region, and this property is used to perform its detection. Core region lines are usually obtained as the ones surrounding the highest density peaks, but this technique is strongly affected by the presence of long horizontal strokes that can be confused with the actual core region and that can lead to severe errors in normalizing the word (see figure 3.2). Many heuristic rules are necessary to handle the problem and the resulting process is not robust. In order to avoid such problems, we analyzed the distribution of the density values rather than the density histogram itself. The density distribution $P(h)$ is expected to be bimodal[1]: one mode corresponds to the high density lines of the core region, the other one corresponds to the other lines, where the density is low. A thresholding algorithm can be applied to the distribution to separate the two modes. The threshold can be used to distinguish between core region lines (above the threshold) and remaining lines (below the threshold).
The density histogram $h(i)$ reports the density for each row i. The density distribution describes the probability $P(h)$ of having h foreground pixels in a line. Long horizontal strokes (e.g. t-bars) give

[1]In the hypothesis that the data is distributed normally around the modes, these correspond to the averages of two gaussians.



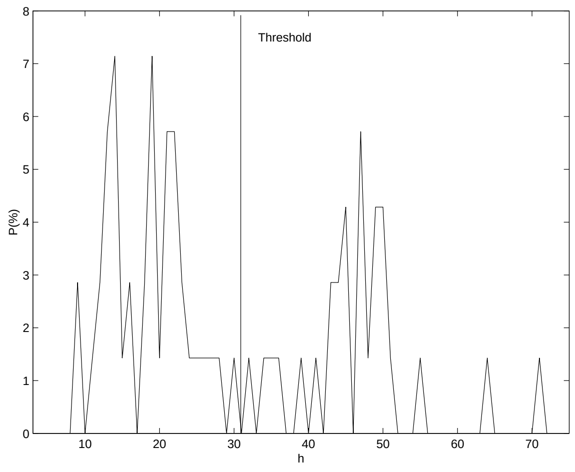


Figure 3.2: Peaks determined by horizontal strokes are evident in the density histogram (upper figure). In the density probability distribution (lower graphic), the density values of the peaks have a small probability. Their influence is negligible and they do not influence the calculation of the threshold.

rise to high peaks in the density histogram (see Fig. 3.2) but their influence in the density probability distribution is negligible because they only occur a few times. For this reason, the threshold extracted from $P(h)$ is robust with respect to local fluctuations of the horizontal density.
We use the Otsu thresholding algorithm [66] that is based on the minimization of the weighted sum of the variances of the following two sets: the group of density elements less than or equal to the threshold and the group of the density elements greater than the threshold. The weights are the probabilities of the respective groups, estimated as the percentage of elements belonging to each group. When several regions have a density higher than the threshold, the region with the highest number of foreground pixels is selected as the core region (see Figure 3.1).
Another approach focusing on the density distribution can be found in [34]. Here the entropy is calculated for distributions corresponding to several rotation-transformed images of the word. The image giving the lowest value of entropy is assumed to be the desloped image.
After the core region is found, we select the points that are used to fit the lower base line. The lowest point l_i of each column i is found, then the set $L = \{l_j : [y(l_{j-1}) < y(l_j)] \wedge [y(l_j) > y(l_{j+1})]\}$, where $y(l_i)$ is the vertical coordinate of the point, is created[2]. The average distance of the L elements from the lower limit of the estimated core region is calculated. The elements with distances less than the average are finally retained to fit the lower baseline. Fitting is done with the Least Mean Squares method. The slope of the fit is used as an estimate of the slope of the word.

[2]Since the image vertical axis is directed downward, the condition $y(l_{j-1}) < y(l_j)$ means that point l_j is lower than point l_{j-1}

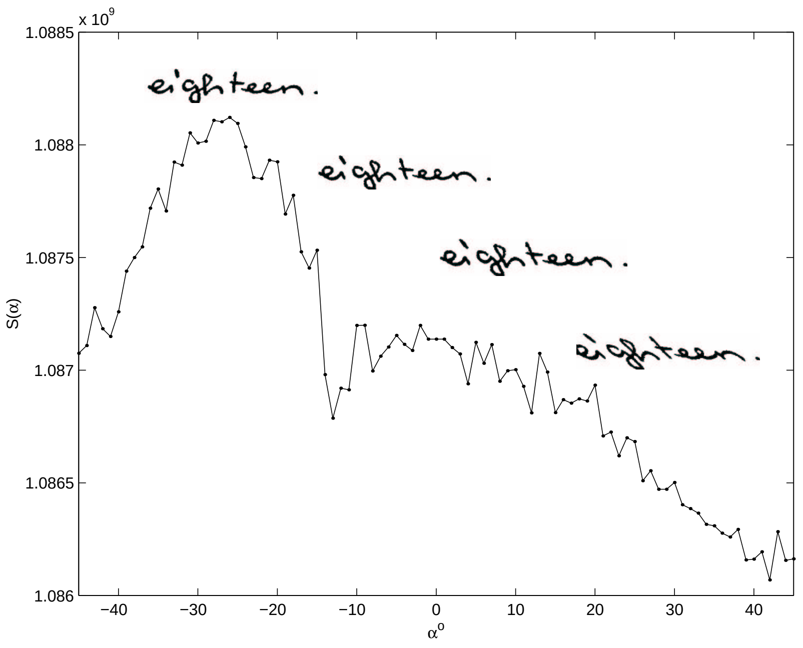


Figure 3.3: Deslanting algorithm. The plot of $S(\alpha)$ shows a high peak corresponding to the deslanted image. For high values of α, when the image is strongly deformed, the value of $S(\alpha)$ becomes lower.

3.2.2 The new deslanting technique

The new deslanting technique is based on the hypothesis that the word is deslanted when the number of columns containing a continuous stroke is maximum. For each angle α in a reasonable interval, a shear transform is applied to the image and the following histogram is obtained:

$$H_\alpha(m) = \frac{h_\alpha(m)}{\Delta y_\alpha(m)} \tag{3.1}$$

where $h_\alpha(m)$ is the vertical density (number of foreground pixels per column) in column m, and $\Delta y_\alpha(m)$ the distance between the highest and lowest pixel in the same column. If the column m contains a continuous stroke, $H_\alpha(m) = 1$, otherwise $H_\alpha(m) \in [0, 1[$.
For each shear transformed image, the following function:

$$S(\alpha) = \sum_{\{i:H_\alpha(i)=1\}} h_\alpha(i)^2 \tag{3.2}$$

is calculated. The angle α giving the highest value of S is taken as the slant estimate (see fig. 3.3). By using in $S(\alpha)$ the square value of the density rather than the density itself, the contribution of the longest vertical strokes is enhanced with respect to that of short strokes, which are often due to stronlgy slanted vertical letters.

The need for multiple shear transforms makes the method computationally intensive, but the computation can be reduced using a recursive procedure to obtain the shear transformed image for angle $\alpha + 1$ from the shear transformed image for angle α. On the other hand, the absence of parameters avoids any experimental effort in parameter optimization.
A similar approach, based on slanted histograms, can be found in [63]. Here, for each slanted image, the highest positive value $D(\alpha)$ of the first derivative of the vertical density histogram is calculated. The slant estimate is then found as $\arg\max_\alpha(D(\alpha))$. The main advantage of our algorithm with respect to such an approach is that the function $S(\alpha)$ is not influenced by a single stroke but by the

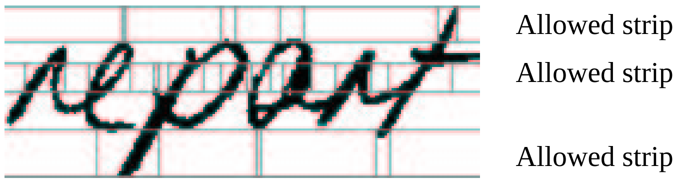


Figure 3.4: The Bozinovic and Srihari deslanting method: the strokes for the slant estimation are selected in the allowed stripes. These must be composed of lines where all of the continuous strokes are shorter than a value MR (Max Run) and higher than a second parameter SH (Strip Height). A stroke is enclosed between vertical lines connecting the borders of an allowed strip without intersecting foreground pixels. When one of the two halves of a stroke is empty, the stroke is discarded.

whole image. The derivative peak $D(\alpha)$ is, on the contrary, completely determined by a single vertical stroke which might be not representative of the slant across the word.
Another approach avoiding the selection of near vertical strokes and relying rather on a global measure over the word can be found in [82]. In this work, the authors use the Wigner Ville Distribution to estimate the degree of *deslantedness* of the word. The measure is repeated over shear transformed word images corresponding to different angles in an interval. The angle corresponding to the image giving the optimal measure is taken as a slant estimate.

3.2.3 The Bozinovic and Srihari deslanting technique

The BSM was proposed in [16] and it is based on the idea that slant information is mostly contained in the almost vertical strokes of the word. These are selected as follows: first all lines containing a continuous stroke longer than a parameter MR (Max Run) are discarded. The remaining lines form stripes that are accepted if their height is higher than another parameter SH (Strip Height) as shown in figure 3.4. The strokes in the finally remaining stripes enclosed between vertical lines connecting the strip borders without intersecting foreground pixels (see vertical segments in figure 3.4) are selected for the slant estimation.
For each selected stroke, the centroid of the upper and lower half are retained and the slope of the line connecting them is assumed as a local slant estimate. The average of all the local slant estimates is used as a global slant estimate. The image is deslanted with a shear transform based on the estimated slant.
The main disadvantage of the method is that parameter tuning is needed. This might result in a heavy experimental effort to find the optimal point in the parameter space (MR and SH), i.e. the point corresponding to the best performance of the recognition system. Furthermore, in the presence of several handwriting styles, pens, and handwritten word dimensions, the estimated parameters can be sub- optimal for many data.

3.3 Sliding Window and Feature Extraction

Once the word is normalized, the systems usually perform a segmentation (see section 2.3.2) by detecting points assumed to separate meaningful fragments of the word. The segmentation involves many heuristics [21][97] because it is difficult to determine general rules to split the handwritten data. Moreover, the features used to locate the splitting points are very noisy (e.g. minima of the word contour or of the vertical projection).
In a sliding window based system, a fixed width window moves from left to right along the word image and stops at regular steps. At each stop, a feature vector is extracted from the isolated frame. This can be considered as a simple form of implicit segmentation. In our system, the window stops at every column and two consecutive frames share all of the columns except for the rightmost one (see

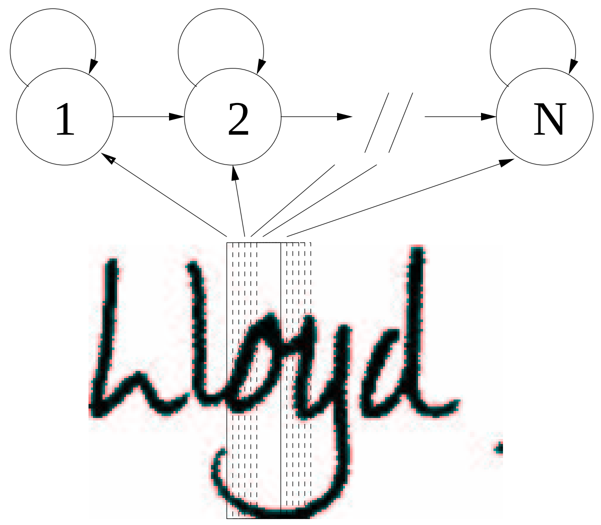


Figure 3.5: Markov modeling. The following frames are overlapped. The solid frame is the first position of the window on the letter, the dashed frames are the following ones. Each frame corresponding to a window position belongs to a state in a letter model. In our system, $N = 9$ and the window width is 10. Only self transitions or transitions to the next state are allowed.

figure 3.5).
The feature vectors extracted from the frames account for the distribution of foreground pixels across the window content. Once a frame is isolated, the area containing pixels is selected (see figure 3.6). This area is partitioned into 16 cells arranged in a regular 4×4 grid. The number of foreground pixels n_i is computed for every cell and the feature vector is obtained as follows:

$$\mathbf{f} = (\frac{n_1}{N}, \frac{n_2}{N}, \ldots, \frac{n_{16}}{N}) \tag{3.3}$$

where $N = \sum_i n_i$.
Because long horizontal strokes in the ascender and descender regions sometimes affect several frames belonging to letters different than the one of which they are part of, a simple heuristic is applied. When a frame is isolated, the positions of the upper and lower baseline are known. If there are foreground pixels outside the core region, but they are not connected to the content of the core region through a stroke, then they are eliminated (see the *y tail* in figure 3.6).
This feature extraction process has two important advantages: the first is that, being based on local averaging rather than on a precise reconstruction of the pattern, it is robust with respect to noise. The second is that its dimension is low (this is especially important when few training samples are available).

3.4 Hidden Markov Modeling

Once the feature extraction process is completed, the image is converted into a sequence of observation vectors. This makes HMMs especially suitable for the modeling because they are probability density function over the space of the vector sequences [121]. A brief and general description of HMMs is given in section 2.3.6 (for a deep introduction see [78][121]). This section focuses on the use of HMMs in this thesis. All aspects relevant to modeling are considered and the motivations behind our choices are given.
The first important aspect to decide about HMMs is the topology. This is encoded in the transi-

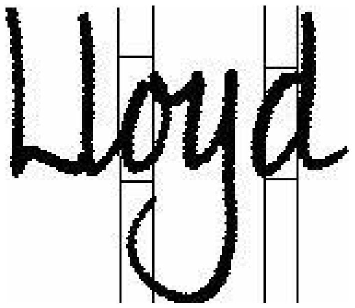


Figure 3.6: Feature Extraction. The extraction process is applied to the frame area actually containing foreground pixels. When some foreground pixels outside the core region are not connected to the content of the core region, they are not considered in the feature extraction process. This limits the noise due to long horizontal strokes in the ascender and descender regions.

tion probability matrix (see section 2.3.6) where a zero element corresponds to an impossible state transition. In the most general case, all elements of the transition matrix are non-zero. This means that, starting from a certain state, it is possible to reach any other state. Since a handwritten word is a sequence of symbols ordered from left to right, the most reasonable topology for handwriting recognition is the so called *Bakis* or *left-right* topology. In this case, the elements a_{ij} of the transition matrix have the following property:

$$\begin{array}{ll} a_{ij} \geq 0 & \text{for} \quad j = i, j = i+1 \\ a_{ij} = 0 & \text{otherwise} \end{array} \tag{3.4}$$

In such topology, only self-transitions and transitions to the following state are allowed. Most of the works presented in the handwriting recognition literature use left-right HMMs (in a few cases, transitions between state i and state $i+2$ are allowed) and this thesis is no exception. All models presented in this work have a Bakis topology.
The second important choice is whether to use discrete or continuous density Hidden Markov Models. The first type of model assumes that the observation vectors are grouped into classes belonging to a finite set C. The emission probability distribution estimates the probability of an element of a certain class being emitted when the system is in a given state. The emission probability density function is thus discrete. In the case of continuous density HMMs, the emission probability distribution is continuous and estimates the probability of an observation vector being emitted when being in a certain state. In this thesis, we use continuous density HMMs.
In principle, HMMs could be used to model the words directly, but this is not the most convenient choice. This requires in fact many samples of the words to be recognized in order to allow reliable training. For this reason, modeling the letters is preferred. The word models can then be obtained by concatenating single letter models. In this thesis, we always apply two important approximations: the first is that all the letter models have the same number of states S and the same number of Gaussians G in the mixtures. The second is that the same model is used for both upper and lower case versions of each letter.
The use of a common S for all models is not strictly correct because certain letters (e.g. i or e) are inherently shorter than others (e.g. m or w). The same applies for G beacuse certain letters are less variable than others and could then be modeled using fewer Gaussians. On the other hand, to set a value of S and G for each letter would lead to a heavy experimental effort not necessarily resulting in a significant improvement of the system performance. The use of the same model for both upper and lower case letters is essentially due to the fact that capital letters are rare compared to lower case characters. They typically do not occur often enough to train good models. Moreover, in some cases, the two versions of the letter differ only in the dimension (e.g. p and P) and the use of a scale invariant feature set (as in our case) makes the two letters equivalent.

The training of the models is performed using the Baum-Welch algorithm [4][5][6][7][8], a version of the Expectation-Maximization technique [38] especially suitable for Hidden Markov Models. The resulting models maximize the likelihood of the training set. The training is embedded, i.e. it is applied to word rather than single letter models. This has two main advantages: the first one is that the letters are modeled as a part of a word (as they actually are in cursive handwriting). The second one is that there is no need to segment the data into letters, a time consuming and error prone process.
The recognition is performed using the Viterbi algorithm [51][147]. This gives the best likelihood λ that can be obtained by following a unique state sequence with the vectors extracted from the handwritten data. Each word model in the lexicon corresponds to a unique state sequence, thus it is possible to estimate a value λ_w for each of them. The word w corresponding to the model giving highest value of λ_w (the most likely model) is selected as the transcription of the handwritten data.

3.5 Experiments and results

This section presents the results of a comparison (in terms of recognition rate) between the deslanting method proposed in [145] and described in section 3.2.2 and the deslanting algorithm developed by Bozinovic and Srihari [16] and described in section 3.2.3.
Two systems, involving the two different algorithms, are compared over both single and multiple writer data. The deslanting technique proposed in this chapter is shown not only to improve the recognition rate, but also to avoid the heavy experimental effort (needed when using the other algorithm) to find an optimal parameter set.
Systems corresponding to different parameter values were trained and tested on a separate training and validation set respectively. When using the new deslanting method, the only parameter to be changed is the number of states in the letter models (the number of Gaussians in the models is fixed). The range of this configuration parameter is constrained by the amount of training material available. When using the BSM, there are three parameters to be tuned: the number of states per letter model, MR and SH.
After having found the best parameter set for both deslanting techniques, the performance of the two corresponding systems was measured again over the test set, giving the final result.
Subsection 3.5.1 describes the data sets used in this experiment, section 3.5.2 and section 3.5.3 present the results over the Cambridge and Bern database respectively.

3.5.1 The data

Since the aim of this thesis is the recognition of handwritten texts, we used databases where the single words are extracted from handwritten pages. This allowed us to deal with two aspects that are specific to such an application with respect to other applications involving cursive word recognition (see chapter 2): the importance of the single writer data and a length distribution slanted towards short words (less than four letters).
Because a text is typically written by a single person, we always performed experiments over both single and multiple writer data that can be considered two apsects of the same problem. For this reason, we used two databases that will be referred to as *Cambridge* and *Bern* database.
The Cambridge database is publicly available [3] and it is composed of 4053 words produced by a single person. The database (originally presented in [131]) is split into training (2362 words), validation (675 words) and test set (1016 words).
The Bern database is a collection of 12199 samples (split into training, validation and test set containing 5347, 2715 and 4137 words respectively) written by around 200 persons. The samples were extracted from a database of handwritten pages collected at University of Bern [106].
The word length distribution of the two sets is shown in figure 3.7. The distribution is peaked in

[3]The data can be downloaded at the following ftp address: `ftp.eng.cam.ac.uk/pub/data`.

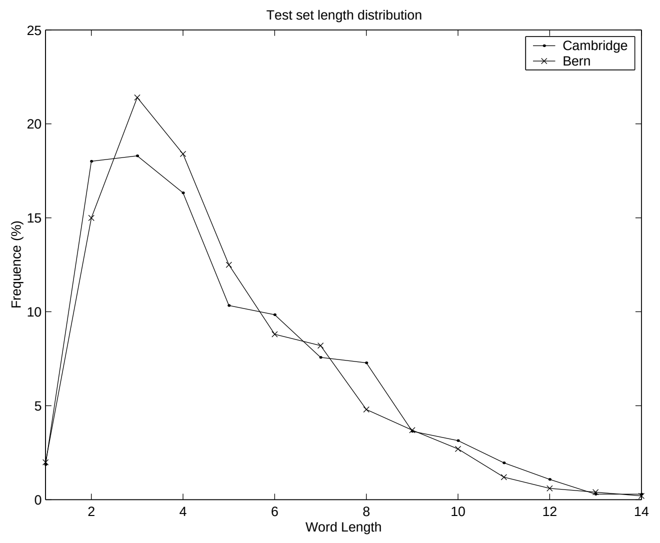


Figure 3.7: Word length distribution of the Bern and Cambridge Database.

both cases in correspondence of three letter long words. This makes the recognition task significantly harder than other single word recognition tasks (e.g. bank check reading). The recognition rate tends in fact to decrease for short words.

3.5.2 Cambridge Database results

The lexicon size, in the experiments over the Cambridge database, is 1334 and the number of states in the letter models tested spans from 1 to 11.
The best performance over the validation set obtained using the deslanting method described in section 3.2.2 is 84.23%. Exploration of the parameter space of the Bozinovic and Srihari algorithm was started at the point corresponding to the average value of the lengths of the continuous horizontal strokes over the database (MR coordinate), and the average height of the allowed stripes given such

		MR				
		8	9	10	11	12
SH	8	78.59	79.92	79.48	80.07	80.37
	9	77.99	79.33	79.48	80.22	80.67
	10	78.44	79.78	81.26	79.48	78.59
	11	79.18	79.63	79.03	79.33	78.44
	12	76.65	79.18	78.74	80.07	76.65

Table 3.1: Cambridge database results. Each column reports the accuracy results obtained changing SH for a given value of MR. Each line reports the results obtained changing MR for a given SH. The central point corresponds to the highest performance obtained. The results were measured over the validation set. Each reported score is the maximum among tests using different numbers of states per letter model, for a given pair (MR,SH).

value of MR (SH coordinate). We selected as optimal a point corresponding to the highest performance over a window centred about this point, and five points wide. Results over the validation set are shown in Table 3.1. The results over the test set are shown in Table 3.3. The recognition rate shows a 3.6%

relative improvement (the confidence on the statistical significance is more than 90%) using the new deslanting technique.

3.5.3 Bern database results

The lexicon size in these experiments is 100. Letter models with a number of states between 9 and 20 were trained to find the optimal value of this parameter.
The best performance achieved with the deslanting described in Section 3.2.2 over the validation set is 57.52% . For the exploration of the parameter space of the Bozinovic and Srihari method the

		MR				
		8	9	10	11	12
	10	52.26	52.18	54.18	52.33	54.41
	11	51.96	53.66	53.39	52.77	52.70
SH	12	53.96	52.70	54.41	52.03	53.00
	13	51.22	51.81	53.15	54.41	53.52
	14	51.74	51.00	51.96	51.51	53.22

Table 3.2: Bern database results. Each reported value corresponds to different values of MR and SH (see caption for Table 3.1). The results are obtained over the validation set.

starting point was selected with the same method as described in the previous subsection. The results obtained over the validation set are shown in Table 3.2. The two winning systems (one for each deslanting method) were tested over the samples of the test set. The use of the new method improves the recognition rate by 10.8% relative to the BSM method (see Table 3.3). This corresponds to a statistical fluctuation with a probability of less than 5%.

DB	BSM (%)	new (%)
Cambridge	81.56	84.50
Bern	54.95	60.89

Table 3.3: Results over the test set. First column reports the database, The following columns report the performance obtained with BSM and the new method respectively. The results were obtained over the test set.

3.6 Conclusions

This chapter presented our approach to the problem of cursive handwriting recognition. A baseline version of our recognizer was described in detail and the strategy followed at each step of the processing was presented.
All the algorithms we applied are based on general assumptions that can be valid for any cursive handwritten data. The only parameters that cannot be set a priori are the number of states and Gaussians in the letter model.
An original normalization technique for cursive handwritten words was presented. The algorithm is composed of two steps, deslope and deslant, and in both cases the use of any manually set parameters has been avoided. This is an advantage because parameter tuning often requires a heavy experimental effort. Furthermore, a parameter set can be optimal only for a given data set.
A comparison between the new deslanting technique and the well known BSM was performed. Two benchmarks were used for the comparison, the first is composed of words produced by a single writer, the second of samples written by several persons with different pens (see section 3.5.1). In both cases,

the system showed a significantly better performance when using the new deslanting technique. The effort needed to find the optimal system when using BSM is much higher and its optimal configuration is data dependent. Use of the new technique has therefore been shown effective in improving recogniton results, while also avoiding a lot of effort in parameter tuning.

Several problems remain open in this version of the system. Firstly, in the experiments shown above, the number of Gaussians is not changed during the validation phase. By finding the optimal number of Gaussians in the mixtures it is possible to improve the recognition rate. Secondly, the feature vectors are used as they are extracted by the system, without applying any algorithm that can compress and decorrelate them. In the next chapter, these problems will be addressed leading to a significant improvement of the recognition rate.

Chapter 4

Improving the system

4.1 Introduction

This chapter addresses some of the problems outlined in chapter 3. The number of Gaussians (that was fixed in the previous experiments) will be set through validation, and several transforms aimed at compression and decorrelation of data will be applied to the feature vectors leading to an improvement of the recognition rate.
Such problems are related not only to our system, but to any recognizer based on HMMs. These are widely applied in Offline Cursive Word Recognition [120][136][139]. The systems using such approaches (see e.g. [18][23][49][89] [90][111][131]) convert the handwritten data into vector sequences by splitting them into fragments and by extracting from each one of these a feature vector. The sequence of observations so obtained is modeled using HMMs that are probability density functions over the space of the vector sequences [121].
Although such an approach is common, no systematic attention was ever paid, to our knowledge, to the possibility of improving the performance of the HMMs by giving the feature vectors a form that is more suitable for the modeling. The raw vectors are typically affected by two fundamental problems: they are embedded in a space that has a dimension d higher than their *Intrinsic Dimension* (ID) and they are correlated (see section 4.2 for a description of both points). Such problems can be solved (or at least attenuated) by applying transforms that can compress and decorrelate the data.
In this work, we used Karhunen-Loeve Expansion, Nonlinear Principal Component Analysis (based on autoassociative neural networks) and Independent Component Analysis. The first two techniques are linear and nonlinear versions respectively of Principal Component Analysis. The last one is a transform that tries to model the data as a linear combination of statistically independent stochastic variables. Depending on the transform, the attention will be focused on compression or decorrelation, however all the transforms are shown to significantly improve the recognition rate of a baseline system that uses raw feature vectors. A maximum reduction of the error rate by 30.3% and by 16.2% was achieved over single and multiple writer data respectively.
The remaining of the paper is organized as follows: section 4.2 describes the disadvantages of using raw data, section 4.3 presents the data transforms applied, section 4.4 describes experiments and results obtained and section 4.5 draws some conclusions and final remarks. This chapter corresponds to references [142][141].

4.2 Raw Data Disadvantages

This section explains why compression and decorrelation (that can be obtained through the transforms presented in this work) of the feature vectors can improve the performance of the HMMs.
The ID of a data set is the minimum number of free variables needed to represent the data without

information loss. In other terms, a data set $\Omega \subset \mathbb{R}^d$ is said to have an ID equal to m if its elements lie entirely within an m-dimensional subspace of $\mathbb{R}^d$ (where $m < d$) [54]. The use of more dimensions than necessary leads to two main problems [86]: *Curse of Dimensionality* and increase of the number of parameters in the HMMs.
Curse of Dimensionality is a phenomenon that can be explained through a simple example [12][75]. Consider a space partitioned into regularly arranged cells: by increasing the dimension, the number of cells increases exponentially. A modeling problem consists essentially of estimating the distribution of the data and this can be done reliably only if the space occupied by the data is well sampled. If the number of cells increases exponentially, the number of data points necessary for a good sampling also increases exponentially. The use of too many dimensions can make a given amount of training data less effective or even insufficient for the modeling. A reduction of the dimension is thus necessary.
The other problem is the number of parameters in the HMMs. This is related to the dimension of the observation vectors. By reducing their dimension, the number of parameters can be significantly reduced. This is important because the amount of available training material limits the number of parameters that can be reliably trained.
The second important problem is the correlatedness. The data is said to be correlated when its covariance matrix:

$$C_{\mathbf{x}} = E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)] \tag{4.1}$$

(where $\mu = E[\mathbf{x}]$) is full. When continuous density HMMs are used, the emission probabilities are modeled as Gaussian mixtures and their covariance matrices should be correspondingly full. Because a full covariance matrix has too many parameters ($\frac{d}{2}(d+1)$ if d is the dimension of the vectors), it is common to use diagonal matrices (having only d nonzero elements). This still allows one to model correlated data, but requires more Gaussians than when the data is uncorrelated. By decorrelating the data, it is possible to reduce the number of necessary Gaussians which results in a more effective training of their parameters, for a fixed amount of training material.

4.3 Data Transforms

For the reasons explained in the previous section, it is desirable to have observation vectors with dimension as close as possible to their ID and uncorrelated. Such conditions can be achieved by projecting (either in a linear or a nonlinear way) onto an appropriate subspace of dimension m in the original data space of dimension d ($m \leq d$).
The literature presents several methods to perform such task (see [86] for a deep analysis of the problem). In this work, we applied Principal Component Analysis (PCA) and Independent Component Analysis (ICA). They have two important advantages. The first is that they are unsupervised, requiring no labeling of the training data (an operation that can be heavily time consuming in handwriting recognition). The second advantage is that they are based on very general assumptions about the data and can thus be applied to many different problems [75][86].
In both cases, the transforms are characterized by a set of parameters that can be estimated using a set T of training vectors. The set is composed of all feature vectors extracted from the words used to train the Hidden Markov Models.
In the next three subsections, the applied algorithms are explained in detail. Subsection 4.3.1 describes the Karhunen-Loeve Expansion (a linear version of PCA), subsection 4.3.2 presents a nonlinear version of PCA based on autoassociative neural networks and subsection 4.3.3 explains ICA.

4.3.1 Karhunen-Loeve Expansion

The KLE is based on the following linear transform:

$$\mathbf{y} = A\mathbf{x} \tag{4.2}$$

where A is a $d \times d$ matrix, $\mathbf{y}$ is the transformed vector and $\mathbf{x}$ is the original vector assumed to have $E[\mathbf{x}] = \mathbf{0}$. The elements of A are estimated so that, when only the first m components of $\mathbf{y}$ are retained, the Mean Squared Error (MSE):

$$E(m) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{y}_i^m - A_m \mathbf{x}_i)^2 \tag{4.3}$$

is minimized ($\mathbf{y}_i^m$ is the vector of the first m components of $\mathbf{y}_i$, A_m is the matrix composed of the first m columns of A and N is the total number of vectors in the training set). It can be demonstrated that the resulting matrix A has columns that corresponds to the eigenvectors of the covariance matrix $\Sigma = E[\mathbf{x}\mathbf{x}^T]$ ordered according to their respective eigenvalues (from the biggest to the smallest). A component y_j of $\mathbf{y}$ corresponds to the projection of $\mathbf{x}$ along the j^{th} eigenvector of the covariance matrix. The y_j's are called *Principal Components* (hence the name PCA). Each eigenvalue σ_j^2 accounts for the data variance along the direction of the corresponding eigenvector. For this reason, the subspace spanned by the first m columns of A accounts for more variance than any other subspace spanned by a different set of m orthogonal axes [75]. This is an important property. In the hypothesis that the variance of the data is given by information useful for a certain task (and not by noise), the subspace of the first m eigenvectors is the one containing more information than any other with the same dimension.
The MSE, when retaining only m Principal Components (see equation 4.3), can also be estimated as follows:

$$E(m) = \sum_{j=m+1}^{d} \sigma_j^2. \tag{4.4}$$

If the last eigenvectors are sufficiently small, the error in (4.4) is negligible and the original data can be compressed without a significant loss of information. This allows one to reduce the dimensionality of the vectors with the advantages described in section 4.2. Although the KLE is not a good estimator of the ID (in the sense that the m for which the error in (4.4) is negligible is typically higher than the actual ID of the data), a significant compression can often be achieved.
When $m = d$, the error in equation (4.3) is null and the transform corresponds to a change of reference frame. The resulting data is uncorrelated and, although no compression is achieved, the performance of the HMMs can still be improved. In fact, the transformed data has a diagonal covariance matrix and can be modeled more effectively than data having a full covariance matrix for a given number of Gaussians (see section 4.2).
Some systems presented in the literature (e.g. [104][81]) used KLE to compress the vectors, but no clear explanation of the criteria used to set the number of retained components was given. Moreover, no comparison was made with respect to a baseline system.
The fundamental limit of the KLE is the linearity of the transform in equation (4.2). When the data is distributed over nonlinear subspaces, the projection onto a linear subspace can be a rough approximation. Possible solutions are to use a nonlinear transform, or to map the data into a space (eventually with higher dimension) where the nonlinear surfaces are hyperplanes and then to apply a KLE in such a new space. This last approach is used with the autoassociative neural networks described in the next section.

4.3.2 Nonlinear Principal Component Analysis

Many pattern recognition problems can be solved with Neural Networks that (even if based on different underlying principles) are found to be implicitly equivalent or similar to statistical methods usually applied to solve them [96][103][123]. The main advantage is that different problems can be solved using a single theoretic framework without the need of deep domain specific knowledge [92].
A KLE can be obtained using an autoassociative (i.e. having as output the input pattern) Multi Layer Perceptron [12] with linear activation functions in the neurons. The training criterion must

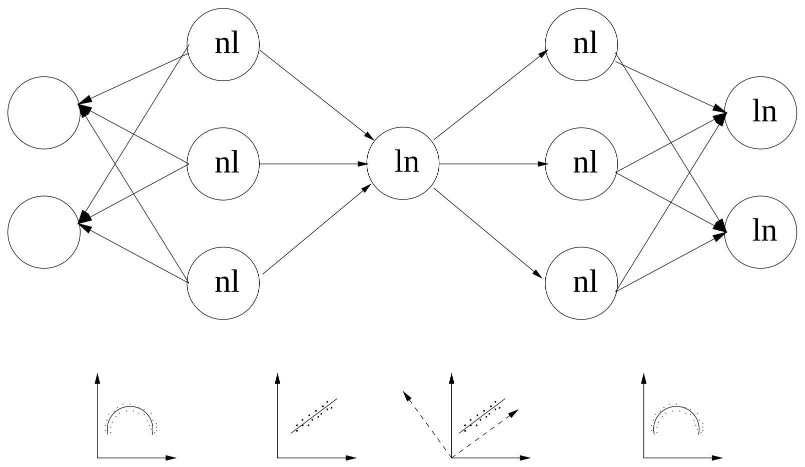


Figure 4.1: Five layer network performing Nonlinear Principal Component Analysis. The neurons labeled with *ln* have linear activation functions, while those labeled with *nl* have nonlinear activation functions. The lower part of the figure shows the effect on the data due to the different layers. At the input the data is distributed with some nonlinear structure. After the first nonlinear layer, the data is mapped into a space where the nonlinear structure becomes linear. The central layer performs a linear PCA (represented by the dashed reference frame). The remaining part of the network maps the data back into the original space.

be the minimization of the MSE. The hidden layer of such a network encodes the KLE of the input data [2][15]. In order to discard the less informative Principal Components, it is necessary to estimate the variance along the different components of the transformed vectors. Another possibility is to train several networks with different central layer sizes.

By changing the architecture of the network it is possible to perform a *Nonlinear Principal Component Analysis* (NLPCA). By non-linear it is meant that the data can be projected onto nonlinear subspaces (e.g. curved surfaces). This is an advantage when the original data is distributed over nonlinear structures and the projection onto linear subspaces (like in KLE) might lead to a high approximation error [52]. This allows one to overcome the main limit of the KLE (see previous subsection). NLPCA was applied to many domains [86], but it was never used, to our knowledge, in Offline Cursive Handwriting Recognition.

The network performing NLPCA has five layers (see figure 4.1). The activation functions are linear in the input, output and central layers and nonlinear in the others [12][75]. This is often called a *bottleneck network* because it forces the data to pass through the central layer that has a number of neurons less or equal to the dimension of the space the input vector is embedded in. The network is autoassociative and the training criterion is the minimization of the Mean Squared Error (see subsection 4.3.1).

The first nonlinear layer (see figure 4.1) performs a mapping of the original data into a space (called *feature space*) where nonlinear structures are transformed into linear ones. The central layer projects the mapped data onto a linear subspace of the feature space. The second nonlinear layer performs the inverse mapping of the first one and the final layer outputs the input pattern.

The NLPCA is encoded by the central layer. The subspace spanned by its output accounts for the biggest amount of variance of the input data (given the number of its neurons). The information loss is then minimized when the original data is compressed [102]. Unlike in KLE, the uncorrelatedness of the transformed data is not guaranteed [86]. This can create problems when using continuous density HMMs (see section 4.2).

A further problem is given by the fact that the mapping of the data onto the subspace spanned by the Principal Components is continuous. This creates problems when the original data is distributed over a surface that is discontinuous or self-intersecting. In correspondence of the discontinuities or of the point of self-intersection, the results are unreliable [102].

4.3.3 Independent Component Analysis

Independent Component Analysis can be defined using a statistical latent variable model. The components of the data vectors $\mathbf{x}$ can be thought of as linear combinations of m stochastic variables s_j that cannot be observed directly:

$$x_i = w_{i1}s_1 + w_{i2}s_2 + \ldots + w_{im}s_m \tag{4.5}$$

where the w_{ij} (with $i = 1, \ldots, d$ and $j =, 1, \ldots, m$) are some real coefficients. This is the basic ICA model, where the w_{ij} coefficients are not known and must be estimated (under assumptions as general as possible) using only the data vectors in T.
This corresponds to estimating the parameters of a linear transform:

$$\mathbf{x} = W\mathbf{s} \tag{4.6}$$

making the problem similar to that of KLE (see subsection 4.3.1). Only two assumptions must be made to ensure that the w_{ij} can be estimated. The first assumption is that the s_j are statistically independent, and the second is that their distribution is not Gaussian [30][70][71]. Two variables s_i and s_j are said to be statistically independent when $p(s_i, s_j) = p(s_i)p(s_j)$. The non-gaussian assumption is necessary because, when the s_i are Gaussians and statistically independent, the transform W can be identified only up to an orthogonal transform and this ambiguity must be avoided.
The above assumptions are very general and, for this reason, ICA can be applied to a wide spectrum of problems. However, the generality causes some ambiguities that cannot be avoided. For instance, it is not possible to determine the variance of the Independent Components because any scalar multiplier in one of the sources s_i can be canceled by dividing the corresponding column in W by the same factor [71].
From a practical point of view, a measure of *nongaussianity*, the *negentropy*, is the basis of the criterion used to find the Independent Components [69][72]. The negentropy of a variable $\mathbf{y}$ is defined as follows:

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y}) \tag{4.7}$$

where $\mathbf{y}_{gauss}$ is a Gaussian variable with the same covariance matrix and mean as $\mathbf{y}$, and H is the entropy:

$$H(\mathbf{y}) = -\int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y}. \tag{4.8}$$

An important advantage of $J(\mathbf{y})$ is that it is related to their mutual information if the y_i are forced to be uncorrelated (which is in our case a desirable condition):

$$I(y_1, y_2, \ldots, y_d) = J(\mathbf{y}) - \sum_i J(y_i). \tag{4.9}$$

The mutual information is the information theoretic measure of the independence of the variables. The Independent Components can then be obtained by finding the directions of maximum negentropy. This transforms the ICA problem into a numerical optimization problem. Since calculation of the negentropy is computationally expensive, several approximations of it have been proposed allowing a fast and reliable estimation of the independent components [68][69].
The ICA is helpful for both compressing and decorrelating the data. The statistical independence implies the uncorrelatedness, which improves HMM performance (see section 4.2). The compression can be achieved by setting $m < d$ in equation (4.5). A survey on the applications of ICA can be found in [71]. Although the algorithm has been used in a wide variety of problems, it has never been utilised in a handwriting recognition system.

4.4 Experiments And Results

The experiments were performed using the system and the data described in chapter 3. In the next three subsections, the estimation of the transform parameters, the results over the Cambridge database and the results over the Bern database are respectively presented.

4.4.1 Transform Parameters Estimation

In the case of KLE, the raw data is transformed through the matrix A (see above). This is obtained by estimating the eigenvectors of the covariance matrix of the vectors extracted from the words used to train the HMMs.
The word recognition experiments involve a validation and a test (see the following subsections). For the validation, the matrix A is obtained using the vectors extracted from the words of the training set. For the test, A is obtained using the vectors extracted from the union of the training and validation sets.
In the case of ICA, the approach used is the same as in the case of KLE, but it must be repeated for each number I of Independent Components. As explained above, the Independent Components cannot be ordered following some criteria, hence the only way to achieve a data compression is to estimate a different transform for each value of I.
A similar approach is taken for the case of the NLPCA except that there is a parameter to set: the number N of neurons in the second and fourth layers. An optimal N is selected for each number of Principal Components (neurons in the central layer) P. For a given P, several networks (with different N) are trained over the training set and tested over the validation set. The network with the smallest N having an MSE lower than a fixed threshold is retained as optimal. The value of the threshold is fixed so that the absolute error per component is 0.01. Our features are percentages (see section 3) obtained from sets containing typically less than 1000 elements. For this reason, the third digit after the comma is very noisy and it is not worth estimating it precisely.

4.4.2 Experiments Over The Cambridge Database

Our system is characterized by the number of states S and Gaussians G per state in its models. For simplicity, S and G are the same for all the letter models. The parameters S and G are selected as follows: models with a number of states S ranging from 8 to 12 and number of Gaussians G between 10 an 15 are trained over the training set and tested over the validation set. Both the ranges of S and G were determined by the available training material. The system showing the best performance was retained as optimal, retrained over the union of training and validation set and finally tested over the test set. This allows one to set the values of S and G by measuring the performance over a set (the validation set) that is independent of the test set. In this way S and G are not fitted to the test set and the performance of the corresponding system is not overestimated. The lexicon size is 1370.
The baseline system was obtained using the raw feature vectors. The recognition rate of the optimal system ($S = 11$ and $G = 12$) over the test set is **92.4%**.
When using KLE, NLPCA and ICA, the parameters to be set through validation are not only S and G, but also the number of retained Principal Components P or Independent Components I that ranged between 13 and 16. The best system based on KLE is obtained with $P = 16$ (using models having $S = 9$ and $G = 13$) and its recognition rate (measured over the test set) is **94.7%**. When using NLPCA, the performance over the test set is **94.0%** (the optimal P, S and G are 14, 9 and 12 respectively). The best system using ICA has $S = 11$, $G = 14$ and $I = 14$. Its recognition rate over the test set is **93.6%**. For both PCA and NLPCA, the result corresponds to an improvement of the baseline system with a probability higher than 90%. The difference in performance between PCA and NLPCA is caused (with high probability: more than 30%) by statistical fluctuations. In the ICA case, there is a significant probability ($\sim$ 15%) that the improvement with respect to the baseline system is simply due to statistical fluctuations.

To evaluate the recognition rate as a function of the number of retained Principal Components (Inde-

data	S	G	D	Acc.(%)
raw	11	12	16	92.4
KLE	9	13	16	**94.7**
NLPCA	9	12	14	94.0
ICA	11	14	14	93.6

Table 4.1: Performance over the test set for the Cambridge database. For each system selected through validation, the number of states S per letter model, the number of Gaussians G per state and the dimension D of the feature vector are reported together with the performance over the test set.

pendent Components), the best system (over the validation set) for each value of P (I) was retrained (over the union of the training and validation sets) and tested over the test set. The results are shown in the left plot of figure 4.2. The error rate is reduced by 30.3%, by 21.0% and by 15.8% using KLE, NLPCA and ICA respectively. The right plot of figure 4.2 shows the performance as a function of the word length for the best systems using raw data, KLE, NLPCA and ICA. The four systems have a similar performance when the length of the word is high (more than 7-8 letters), but for the short words (that are more difficult to recognize) the transform based systems work significantly better.
The Cambridge database has been used in several works in the literature and this allows a comparison of the results. The best result obtained in [131] (where the dataset was originally presented) is 92.8%. Another result can be found in [137], claiming a recognition rate of 94.6%. Finally, a performance of 90.8% (but with a 1000 words lexicon) is shown in [1]. The results we obtained are better than those obtained in such works and, to our knowledge, no better results were ever reported over the same data.
Together with a better performance, our system is also much more simple than the other systems. These use high-dimensional feature vectors (more than 70 features), involve a segmentation and classify the fragments extracted from the word with neural networks or other pattern recognition techniques.

4.4.3 Experiments On The Bern Database

The same procedure followed for the Cambridge database was used for the Bern data. In this case, S ranges between 12 and 16 while G goes from 10 to 15.
The best baseline system (the lexicon size is 100) has $S = 14$ and $G = 12$. Its recognition rate over the test set is **77.8**%.
KLE, NLPCA and ICA were applied using the same method described in the previous subsection with P and I ranging from 11 to 16. The system selected through validation when applying KLE has $S = 15$, $G = 12$ and $P = 16$ (the best performing system with $P = 14$ was not selected because it had a lower recognition rate over the validation set). Its performance (over the test set) is **78.8**%. The best NLPCA based system has $S = 15$, $G = 12$ and $P = 14$. The recognition rate is **81.4**%. The best system using ICA has $S = 14$, $G = 11$, $I = 16$ and a recognition rate over the test set of **78.6**%.

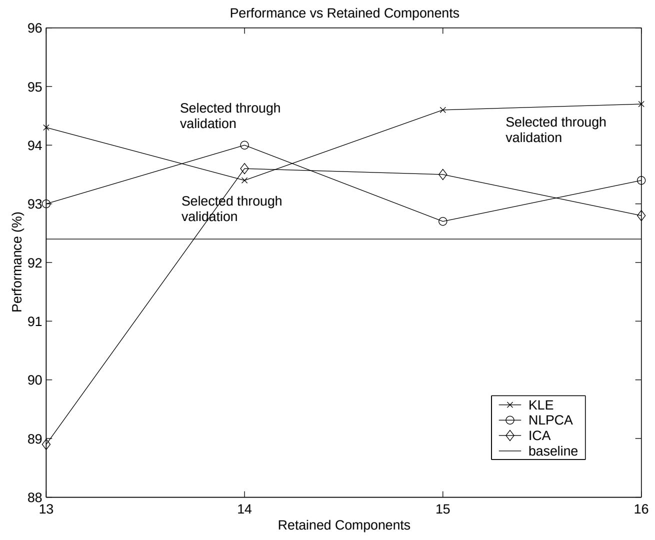
The probability of the improvement obtained with NLPCA being a statistical fluctuation is less than 1%. It is therefore possible to conclude that the baseline system recognition rate has definitely increased. In the case of the other transforms, the probability of the improvement being due to statistical fluctuations is around 15%.
The performance of the best system for each P (I) was measured over the test set (as for the Cambridge database) and the results are shown in figure 4.3. The NLPCA performs in general better than KLE and ICA. The latter obtains a slight improvement of the baseline system and appears to be not as effective over this database.
The right plot of figure 4.3 shows the performance of the system as a function of the word length. As for the Cambridge database, the transform based systems are shown to work significantly better over the short words (less than 4 letters).

data	S	G	D	Acc.(%)
raw	14	12	16	77.8
KLE	15	12	16	78.8
NLPCA	15	12	14	**81.4**
ICA	14	11	16	78.6

Table 4.2: Performance over the test set for the Bern database. For each system selected through validation, the number of states S per letter model, the number of Gaussians G per state and the dimension D of the feature vector are reported together with the performance over the test set.

4.5 Conclusions

This work addressed the problem of giving the feature vectors a form more suitable for being modeled with HMMs. This is done by applying three transforms (KLE, NLPCA and ICA) that were compared using both multiple and single writer data.
The transforms are beneficial from several aspects. They can allow one to compress the vectors resulting in data easier to model and HMMs with fewer parameters. Moreover, when using KLE and ICA, the data is decorrelated and is then more suitable for modeling with Gaussians holding diagonal covariance matrices (in this case, the performance of the system can be improved even without compression).
The transforms were evaluated over writer dependent and writer independent data sets. The error rate (compared to the baseline system) was reduced by 30.3% for the single writer data and by 16.2% for the multiple writer data.
No transform appears to be systematically better than the others. However, ICA was shown to give the lowest improvement over the baseline system across both databases. KLE and NLPCA are shown to perform close to each other for single writer data, while on multiple writer data, the NLPCA seems definitely superior. The presence of many writers results probably in a more complex distribution of the data. In this case, the NLPCA can then take more advantage from its better ability (due to nonlinearity) in capturing the structure underlying the data distribution.
Across both databases, the baseline system showed the lowest performance. This is because the feature vectors extracted from the handwritten samples are both correlated and embedded in a space of dimension d higher than their ID. Any system using feature vectors with the same problems can take advantage of the application of the transforms used in this work or of other techniques able to compress and decorrelate the data.



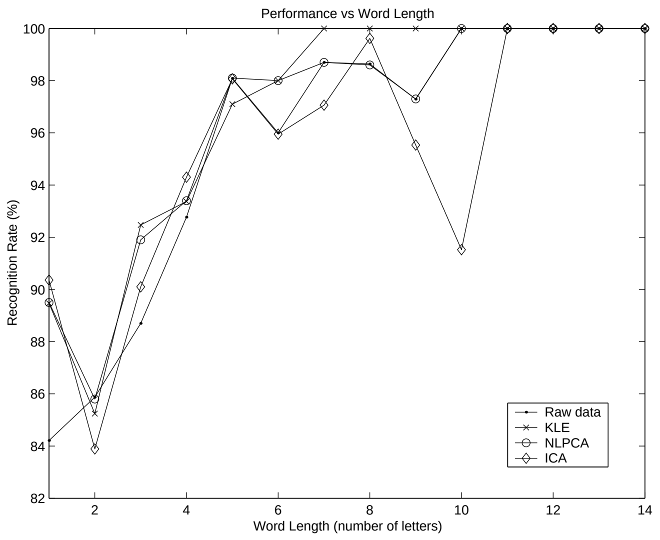


Figure 4.2: Performance over the Cambridge database. The left plot shows the performance of the system over the test set as a function of the number of Principal (Independent) Components retained. The systems selected through validation are highlighted. The right plot shows the performance (over the test set) of the systems selected through validation as a function of the word length.

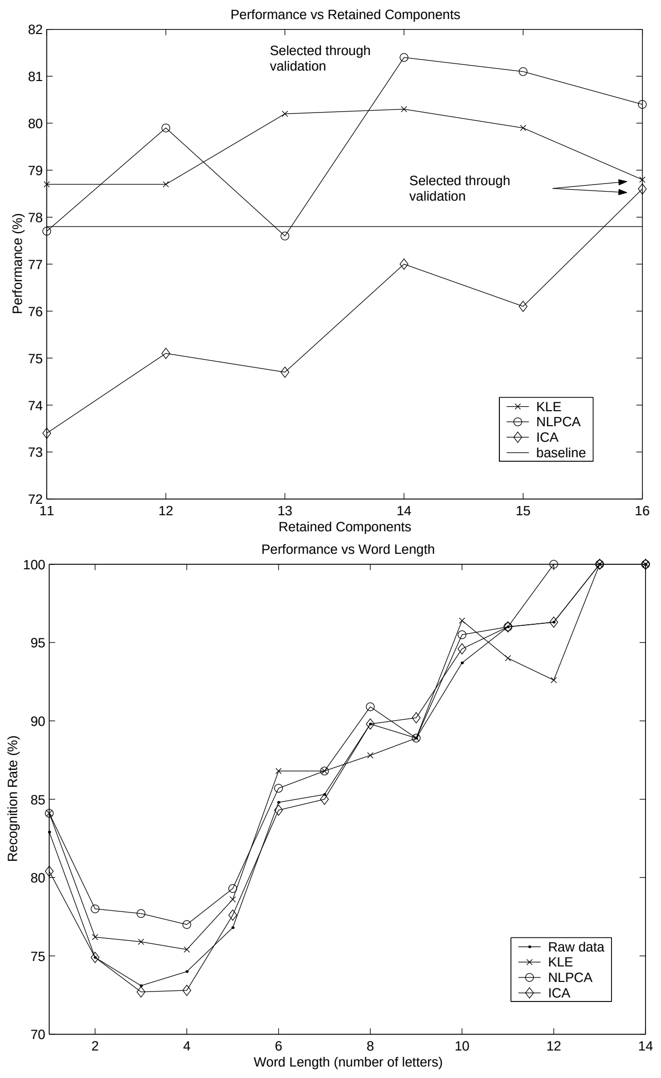


Figure 4.3: Performance over the Bern database. The left plot shows the performance of the system over the test set as a function of the number of Principal (Independent) Components retained. The systems selected through validation are highlighted. The right plot shows the performance (over the test set) of the systems selected through validation as a function of the word length.

Chapter 5

Writer Adaptation

5.1 Introduction

In view of the recognition of handwritten texts, it is important to focus on the recognition of samples produced by a single person. In the last years, several efforts were led, in the domain of Off-Line Cursive Script Recognition (CSR), towards such a direction [91][104][108][107][131]. In this task, the best performance is achieved by using, for training, samples produced by the writer himself. On the other hand, the training is reliable only if there is enough data and this might be difficult because the writer would be required to produce many samples.
This problem can be solved by applying adaptation techniques. The literature presents several techniques for HMM adaptation [40][59][95]. Their aim is to improve the performance of the models over specific subsets of the data they are trained to recognize.
A training set is typically composed of large sets of data produced by different sources. The HMMs trained over such data are not optimal with respect to any source, but they have a good performance on the data produced by all of them. Moreover, they have a good recognition performance also over data produced by sources not represented in the training set.
In some situations, it can be desirable to have models optimal for a certain source, but this is not possible because not enough source dependent data is available. When this is the case, the adaptation techniques can be used to fit the HMM parameters to the distribution of the source dependent data. The resulting models are closer to the optimal solution (represented by models trained using only source dependent data) than the source independent models and are the best solution in the absence of sufficient source dependent data.
The adaptation techniques are not influenced by the nature of the sources. It is not necessary to modify them depending on the specific application they are involved in. In the case of the handwriting recognition, the sources can correspond to the different writers.
The models are typically trained over samples produced by many sources, i.e. many writers. The resulting HMMs are not optimal for any single writer, but they are good for all of them. If the training set is sufficiently representative of the different handwriting styles, the models will achieve a good performance also over data written by persons that did not produce samples for the training set.
The adaptation allows us to optimize the models for a single writer given a set of his data that can be much smaller than the amount of data actually needed for writer dependent training. For this reason the adaptation process is referred to, in this case, as *Writer Adaptation.* The models obtained through the adaptation are called *Writer Dependent* (WD) in contrast to the original models being called *Writer Independent* (WI).
This work presents experiments performed in adapting continuous density HMMs (having mixtures of Gaussians as emission probabilities) trained over a WI database to WD data. Models adapted on WD data sets of different size are compared with WD HMMs trained over the same sets. The results show that, for our database, ~200 words (a considerable amount for a single writer) are needed to

obtain WD models performing better than the adapted ones. Moreover, with less than 30 WD words it is not possible to train WD models because not all the letters are represented in such a small set, while the result with the adaptation method already performs well. This shows that the adaptation can be a good solution for obtaining WD models when it is difficult to collect WD data but when reliably trained WI HMMs are available.
This chapter is organized as follows: section 5.2 describes the HMM adaptation techniques, section 5.3 reports experiments and results, and section 5.4 presents some conclusions. The contents of this chapter is published in references [143][140].

5.2 Adaptation Techniques

The adaptation techniques allow one to improve the performance of WI models over WD data when there is not enough data for reliably training WD models. The process consists of adapting the parameters $\theta = (\theta_1, \theta_2, \dots, \theta_M)$ of the WI models using the WD data.
The probability of having a parameter vector θ given the adaptation set of observations $\mathbf{O}$ can be written (using Bayes theorem) as follows [46]:

$$p(\theta|\mathbf{O}) = \frac{p(\mathbf{O}|\theta)p(\theta)}{p(\mathbf{O})} \tag{5.1}$$

where $p(\theta|\mathbf{O})$ and $p(\theta)$ are, respectively, the *posterior* and *prior* distribution of the parameters, and $p(\mathbf{O}|\theta)$ is the likelihood of the HMM with parameter set θ. The aim of the adaptation is to find the vector θ_{ad} maximizing the posterior:

$$\theta_{ad} = \arg\max_{\theta} p(\theta|\mathbf{O}). \tag{5.2}$$

This can be done in two different ways depending on the prior distribution of θ. If $p(\theta)$ is *noninformative*, i.e. does not give any information about how the θ components are likely to be, then the adapted parameters θ_{ad} are estimated with Maximum Likelihood (ML). Since a noninformative prior distribution corresponds to a constant uniform distribution $p(\theta) = c$, this amounts to solving the equation:

$$\frac{\partial p(\mathbf{O}|\theta)}{\partial \theta} = 0 \tag{5.3}$$

When the prior distribution is *informative*, i.e. the distribution $p(\theta)$ is different than a constant, then the adapted parameters are obtained by solving the equation

$$\frac{\partial \left(p(\mathbf{O}|\theta)p(\theta)\right)}{\partial \theta} = 0 \tag{5.4}$$

This corresponds to a Maximum A Posteriori (MAP) estimation of θ_{ad}.
When the adaptation is performed with ML, θ_{ad} is estimated so that the probability of the adapted models generating the adaptation data is maximized. When the adaptation is performed with MAP, θ_{ad} is such that the Bayes risk [46] over the adaptation set is minimized (hence the name Bayesian Adaptation).
In the experiments, ML will be used to estimate the parameters of a linear regression which transforms WI parameters into WD parameters (Maximum Likelihood Linear Regression, MLLR). The method will be shown to be effective for a limited amount of adaptation data, but it quickly converges to a saturation performance that cannot be improved any further. On the other hand, the MAP estimation needs, in order to be effective, more adaptation data, but the performance of the adapted models converges to that of the WD models. A third possibility is given by the combination of the two approaches. A model is first adapted with MLLR. Then its parameters are used to obtain a prior distribution information and to perform a MAP adaptation.
In this work, the adaptation is applied to continuous density HMMs: the probability of emitting an

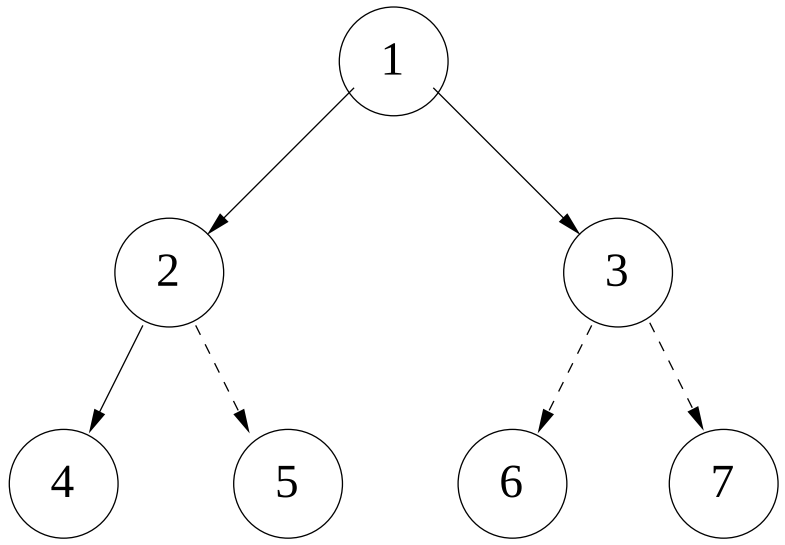


Figure 5.1: Regression tree with four base classes. The dashed line means that the child node does not contain sufficient data. In the case of the figure, there are three regression classes. The first one is composed of data clustered at node 4. Such data is transformed by a matrix estimated with the data themselves. The second one is composed of data clustered at node 5. The transform matrix of such data is estimated using data clustered at node 2 because the data belonging to node 5 are not enough. The last class is composed of data clustered at nodes 6 and 7. Since the data are insufficient for both nodes, they share a common transform matrix estimated with the data clustered at node 3 and then form a single class.

observation $\mathbf{o}$ when being in a given state of the HMM is modeled by a mixture of Gaussians [121]. The parameters to adapt are thus the means, variances and weights of these mixtures of Gaussians. We can furthermore simplify the adaptation techniques by making the hypothesis that the WD information is carried essentially by the means of the Gaussians. Hence only such parameters will be adapted. The parameter vector θ corresponds to the vector $\mu = (\mu_1, \mu_2, \ldots, \mu_G)$, where G is the total number of Gaussians.
The MLLR technique is conceived to be effective with little adaptation data. Because of this, most Gaussians might be poorly or not at all affected by the adaptation process. This can be overcome by clustering the Gaussians, i.e. by grouping them so that, during the adaptation process, most of them can be updated [57].
The MAP technique adapts each Gaussian separately. This makes it necessary to have more adaptation material, but this gives better results as the adaptation set size increases.
Subsection 5.2.1 explains how the Gaussians are clustered for the MLLR technique. Subsections 5.2.2 and 5.2.3 give, respectively, more details about ML and MAP techniques.

5.2.1 Gaussian clustering

The first step in MLLR adaptation is the Gaussian clustering performed by grouping the Gaussians into *regression classes*. All the Gaussians of a cluster share the parameters of a linear transform leading from the WI means to the WD ones (see section 5.2.2). This allows the adaption as well of the Gaussians for which there is not enough data in the adaptation set.
The regression classes are determined dynamically according to the amount of data available (in the WI data set) using a binary *regression class tree* [57]. This is grown using a centroid splitting algorithm based on a Euclidean distance measure. Mean and variance from the Gaussians clustered at a given

node to be split are calculated. Two children are created and their means are initalized to the mean of the parent perturbed in opposite directions by a fraction of the variance. Each Gaussian clustered at the parent node is assigned to one of the child nodes (depending on the distance); and, when all the Gaussians are assigned, the mean and variance for the children nodes are calculated. The process is repeated until the predetermined number of leaf nodes is reached.
Once the tree is grown, the regression classes can be determined following the scheme illustrated in figure 5.1. A solid arrow means that the child node contains a number of observations (each one being attributed to the most likely Gaussian) above an experimentally determined threshold, and can thus form a regression class. In the other cases the data is insufficient and its Gaussians belong to the regression class of the parent node.
Since the tree is grown using the WI data, it is independent of the writer and can be used to perform the adaptation to any writer.

5.2.2 Maximum Likelihood Linear Regression

At the beginning of the adaptation process, the Gaussians are grouped into clusters. When a Gaussian of the cluster is hit by an observation of the adaptation set, all the Gaussians of the same cluster are updated.
Consider the Gaussian:

$$N(\mathbf{o},\mu_{i_g},\Sigma_{i_g}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_{i_g}|^{\frac{1}{2}}} e^{\frac{1}{2}(\mathbf{o}-\mu_{i_g})'\Sigma_{i_g}^{-1}(\mathbf{o}-\mu_{i_g})} \tag{5.5}$$

where Σ is the covariance matrix, μ the mean and d the dimension of the observation space. The index i_g states that this is Gaussian i of cluster g.
Following the hypothesis proposed in [40][94][95], we adapt only the WI means μ using the linear transform:

$$\hat{\mu}_{i_g} = W_g \xi_{i_g} \tag{5.6}$$

where W_g is a $d \times (d+1)$ matrix, and $\xi_{i_g} = (\omega_g, \mu_{i_g1}, \ldots, \mu_{i_gd})$, where ω_g is an offset. Since ML is used to estimate the parameters of a linear transform, the method is called Maximum Likelihood Linear Regression.
After the transform, the Gaussians become:

$$\hat{N}(\mathbf{o},\mu_{i_g},\Sigma_{i_g}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_{i_g}|^{\frac{1}{2}}} e^{\frac{1}{2}(\mathbf{o}-W_g\xi_{i_g})'\Sigma_{i_g}^{-1}(\mathbf{o}-W_g\xi_{i_g})} \tag{5.7}$$

The W_g (the index is g because all of the Gaussians in the cluster share the same matrix transform) elements are selected to maximize the likelihood of the adapted models generating the adaptation data. The ML estimation is performed with the Expectation-Maximization algorithm [38]. By formulating the standard auxiliary function and then maximizing it with respect to the transformed mean, the following equation is obtained:

$$\sum_{t=1}^{T}\sum_{r_g=1}^{R_g} \gamma_{r_g}(t)\Sigma_{r_g}^{-1}\mathbf{o}_t\xi'_{r_g} = \sum_{t=1}^{T}\sum_{r_g=1}^{R_g} \gamma_{r_g}(t)\Sigma_{r_g}^{-1}\bar{W}_g\xi_{r_g}\xi'_{r_g} \tag{5.8}$$

where $\bar{W}_g$ indicates the estimated parameters at a certain iteration, and $\gamma_{r_g}(t)$ the probability of being in state r_g at sample t of the adaptation set. The sum over t involves all the observations $\mathbf{o}_t$ belonging to the adaptation set $\mathbf{O}$; the sum over r_g involves all the Gaussians belonging to the cluster g. Once the elements of $\bar{W}_g$ are estimated at a certain iteration n, they are used to modify the means μ_{r_g} and, correspondingly, the probabilities γ_{r_g}. These can be used again in equation 5.8 to obtain the estimation at iteration $n+1$ of W_g.
The matrices W_g are randomly initialized. Then by repeating the process (until the change in likelihood of the data between two iterations falls below a predefinite threshold), they are iteratively

optimized to approximate ML estimations.
Note that the μ adapted by ML will always stay a linear function of the WI μ, which explains why it adapts quickly, but cannot benefit from a lot of adaptation material.

5.2.3 Bayesian adaptation

In the Bayesian framework, the first problem to be solved is finding an appropriate prior distribution $p(\theta)$.
Following [58][59][93], we consider a mixture of Gaussians as the marginal pdf of $p(\theta)$:

$$p(\mathbf{o}|\theta) = \int_\Theta p(\mathbf{o}, \theta)p(\theta)d\theta \tag{5.9}$$

This leads to expressing $p(\theta)$ as a product of a Dirichlet density [79], accounting for the mixture weights ω_i, and a normal-Wishart density [37], accounting for the other parameters.
Such a prior distribution has the important property of belonging to the conjugate family of the complete data density in the HMM case. This allows us to apply the Expectation-Maximization technique to obtain a MAP estimate of the adapted parameters. We make the same assumption as in subsection 5.2.2 and we thus adapt only the means. The use of EM then leads to the following equation for the adaptation of the means:

$$\hat{\mu}_{jm} = \frac{N_{jm}}{N_{jm} + \tau}\bar{\mu}_{jm} + \frac{\tau}{N_{jm} + \tau}\mu_{jm} \tag{5.10}$$

where N_{jm} is the occupation likelihood of the adaptation data, i.e. the sum of the probabilities of each observation in the adaptation set being emitted by the Gaussian m in state j (the index of the model is omitted for simplicity), τ is a parameter related to the prior distribution (the value is set empirically), $\bar{\mu}_{jm}$ is the mean of observed adaptation data and μ_{jm} is the WI data mean. When N_{jm} is small, the adapted mean is close to the WI mean. Otherwise the adapted values shift toward the mean of the adaptation data. At each iteration of the EM, all of the $\hat{\mu}_{jm}$ estimated at the previous iteration are used in the Gaussians of the cluster g. This leads to new values for N_{jm} and $\bar{\mu}_{jm}$ and through equation 5.10, a new value for $\hat{\mu}_{jm}$. This iterative procedure is repeated until the change in the parameters between two successive iterations falls below a predefined threshold.

5.3 Experiments and results

The experiments were performed using the system and the data described in chapter 3. The experimental setup is different from the one used commonly in recognition experiments. The Bern database is used to train a WI system, while the Cambridge database is used for both training a WD system and adapting the WI system.
The Bern database was split into two parts, a training set (8160 words) and a test set (4018 words). In this specific experiment, the test set could be used as a validation set (to find the optimal number of states and Gaussians in the models) because the aim of the experiment is not to measure its performance, but to obtain a good WI system. Systems with number of states S between 10 and 20 and number of Gaussians G between 1 and 10 were trained over the training set and tested over the test set. The best results were obtained with a system having S=14 and G=7. This system was retrained over the whole Bern database and it was used as a WI system in the adaptation experiments.
The WD system is obtained using the Cambridge database. The data set was split into two parts, a training set (2700 words) and a test set (1353 words). The partition is not performed, as usual, with a random process. The words are stored in the same order as they were written and the training set is composed of the first 2700 words. The reason is that the same data set will be used for the adaptation of the WI system and, in a realistic situation, such a process is performed over the first n samples in order to improve the performance over the following ones.

The size of the training set is progressively increased (while keeping the test set unchanged) in order to show the dependence of the system performance upon the number of samples used to train (adapt) the WD (WI) models. For the first 200 words, the training (adaptation) set size increasing step is 10; afterwards, it becomes 100.
A WD (WI) system is then obtained by training (adapting) the WD (WI) models over the first 10, 20, 30,..., 200 words as well as over the first 300, 400,..., 2700 words of the WD database.
The training technique used for the WD models is the same as the one described above for the WI models. Models with $6 \leq S \leq 16$ and $1 \leq G \leq 5$ were trained and tested.

5.3.1 Adaptation Results

The results obtained with training/adaptation set sizes between 10 and 200 are shown in figure 5.2. The performance is measured in terms of recognition rate with a lexicon of 100 words [1]. The upper plot shows that, in this range of training/adaptation set size, the adapted models perform significantly better than the WD ones. The lower plot shows the performance of the adapted models more clearly. The models adapted with ML increase their performance slowly with respect to the others and their accuracy falls below that of the WD models more quickly than the models adapted with other techniques. The MAP method produces models which are slightly more accurate than the combination of ML and MAP (the differences in performance between the various adaptation methods are mostly due to statistical fluctuations).
The results obtained with training (adaptation) set sizes between 300 and 2700 are shown in figure 5.3. When more than 200 words are available in the training set, the WD models become better than the adapted ones. This limit must be considered a lower bound because in our experiments the WD system was trained using the test set for cross validation, therefore its performance is biased towards it. This leads to an overestimation of the recognition rate of the WD models and it is more correct to say that they need then *at least* 200 words in order to perform better than the adapted models.
The models adapted with ML do not take any advantage of the increase of the adaptation set, while the models adapted with other techniques show some improvements in accuracy. The difference in performance of the three adaptation techniques is due with high probability (more than 85%) to statistical fluctuations.
As mentioned above, the adaptation set is composed of the first n samples produced by the writer. This corresponds to a realistic condition where the WD samples available at a certain moment are used to improve the performance over the data written afterwards.
If the experimental conditions allow one to select arbitrarily the samples (e.g. by asking the writer to write some predetermined words), the system can be made effective with fewer data by using words containing all the characters. In this way, all the letter models are affected by the adaptation process and the system is adapted more quickly.
The selection of the samples can have influence over the speed of the adaptation, but cannot in any case improve the performance of the adapted models. The HMMs adapted with MLLR are limited by the fact that the new parameters are a linear fuction of the old ones. The models adapted through the Bayesian approach cannot in any case perform better than models trained directly (once enough training data are available) with WD data.
A different technique to adapt a CSR system to a single writer is presented in [91]. In this work, the CSR system segments the words into letters and then recognizes them separately with a Neural Network. A contextual analysis of the character recognizer results allows the correction of eventual misclassifications.
The adaptation consists of retraining the neural network using only characters of the specific writer extracted from the WD samples using a WI system. From this point of view, the system adapted with this technique corresponds to what we call a WD system, i.e. a system trained directly over the

[1]The lexicon of the WD database is composed of 1370 words. In order to perform the recognition against 100 words, each sample uses a specific lexicon composed of the correct transcription and 99 words selected randomly from the original 1370 word lexicon.

samples of the specific writer.
To train with WD data is the optimal solution, but requires, to be effective, a sufficient amount of data. The aim of the HMM adaptation is not to improve the system performance (the best accuracy can be achieved in any case only by training directly with WD data), but the possibility of having an effective system when the available WD data is not sufficient for reliable training.
Moreover, in order to retrain the network it is necessary to have samples of all the characters, while in adapting the letter models, it is possible to selectively adapt only the models related to letters presented in the adaptation set (leaving the others unchanged). This allows the adapted system to be effective with very small amounts of adaptation data.

5.4 Conclusions

We have presented in this chapter an application of HMM adaptation techniques to the problem of Off-Line Cursive Script Recognition. A CSR system trained over a database of samples produced by many writers was adapted to the words of a single writer data set.
The experiments showed that, for our database, the models trained over sets containing less than ~200 words have an accuracy inferior to that of models obtained adapting WI models to the same data. This estimated amount must be considered a lower bound since the WD models are trained using the test set for cross-validation and this results in an overestimation of their performance.
For very small training sets (less than 30 words) it was not even possible to train WD models, while there was no problem in adapting WI models. For a 30 word set, the WD system had an accuracy of 34.6% while a WI system adapted with the combination of ML and MAP techniques had an accuracy of 87.3%.
Both adaptation techniques and training methods for continuous density HMMs have been described in detail. The effect of increasing the number of Gaussians in the mixtures has also been shown.
Some improvements can be obtained by using more advanced adaptation techniques and better WI models. Moreover, by studying an optimal composition of the adaptation set, it will be possible to minimize the number of necessary samples for an effective adaptation. The absence of some letters in the first words is due to the fact that our database is a transcription of a text, but it is possible to create smaller sets of words containing all the letters by asking the writer to produce them.
The adaptation techniques are a natural solution for all the applications where it is necessary to recognize samples produced by a single writer without having enough samples for a reliable training.

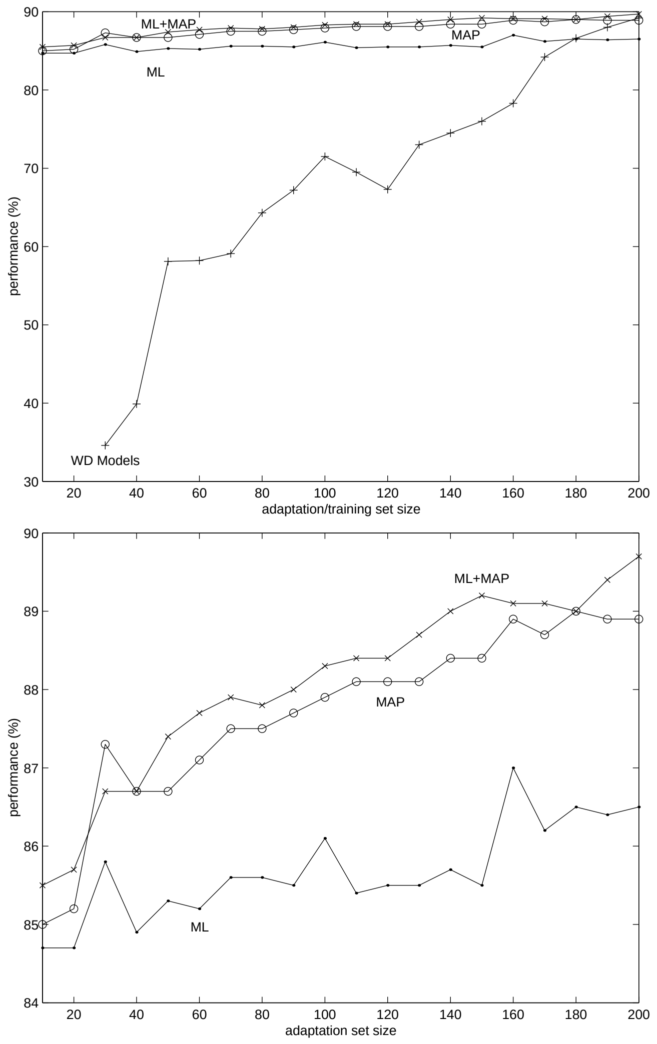


Figure 5.2: The upper figure reports the accuracy vs the training/adaptation set size for the WD models and the WI models adapted with ML, MAP and ML+MAP techniques. In the lower plot, the curves of the adapted models are shown in more details. The values are reported for training/adaptation set sizes less than or equal to 200. When the training/adaptation set size is less than 30, no WD system can be trained.

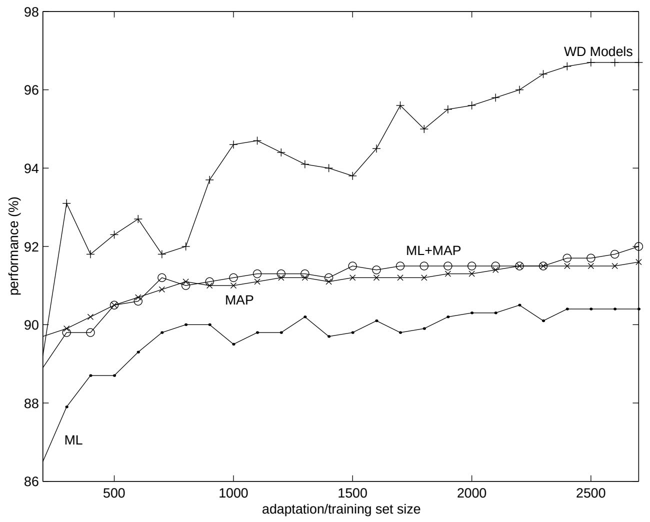


Figure 5.3: The figure shows the accuracy of WD and adapted models for sizes of the training/adaptation set ranging from 300 to 2700.

Chapter 6

Text Recognition

6.1 Introduction

The offline cursive handwriting recognition systems presented in the literature deal, almost without exception, with single words [120][136][139]. This happened, in our opinion, because the research focused on application domains (handwritten address recognition and bank check reading) where only single words are involved. Moreover, datasets allowing experiments on the recognition of texts have only recently been made available [131][151][109].

This chapter shows how the system presented previously deals with unconstrained texts segmented into lines. The recognition of word sequences has rarely been addressed in the literature. The few works dedicated to this problem (see below for a quick survey) are based on significantly different approaches than ours, or impose severe constraints to make the problem easier. In this chapter, on the contrary, the texts to be recognized are unconstrained and no information about their content is used in order to build the system. The texts we recognize are the transcription of documents belonging to a corpus assumed to reproduce the statistics of average English. This allows us to apply Statistical Language Models (SLM) in order to improve the performance of our system [125]. We used N-gram models (the most successful SLM applied until now [78]) of order 1, 2 and 3 (called unigrams, bigrams and trigrams respectively). Previous works typically preferred the application of syntax based postprocessing in order to include linguistic knowledge in the recognition. The trigrams in particular were never used, to our knowledge, in offline cursive handwriting recognition.

The fact that a legal amount can be interpreted as a word sequence (e.g. *three thousand five hundred and fifty seven*) was used in bank check reading. In this application, the lexicon is small (20-30 words) and only a limited range of amounts (e.g. between 1 and 10'000 dollars) is processed automatically. Such constraints limit significantly the variability of the data. In [117] the output of a recognizer (that segments the amount into single words and transcribes them separately) is analyzed with a parser and only the solutions which are plausible from a syntactical point of view are retained. In [63], the sentence level is used only to help the segmentation into words, reducing the problem to single word recognition.

Sentence based approaches have also been tried in postal address recognition [27][98][100]. In this case, the constraint is determined by the zip code (which limits the lexicon to 10-1'000 words) and by the fact that the structure in an address line is not very variable. Also in this case, the sentence level is used to help the segmentation rather than the recognition. In [27], the sentence level is used to locate the different information in an address (street, town, recipient name, etc.). In [98][100], features extracted from the whole sentence are used to accept or reject the output of a single word classifier recognizing each word separately.

Some works were finally dedicated to the recognition of texts. A first approach [85][134][133] segmented a sentence into single words, recognized these words separately and then applied a syntax-based postprocessing, improving the recognition rate of the system. The segmentation of a sentence into single words is as difficult and error prone as the segmentation of a word into letters. This required the use of heuristics in order to recover segmentation errors and the processing of many different hypotheses. An approach based on Hidden Markov Models and N-grams (unigrams and bigrams) was proposed in [104][108]. The use of HMMs avoided the segmentation and the N-grams were shown to improve the performance of the system. A limitation of the proposed approach is the use of a lexicon extracted from the test set. This determines an overfitting of the system to the data to be recognized and makes the system unable to work on unconstrained texts (if the test set is changed, most of the words will no longer be represented in the lexicon).

When passing from single word to text line recognition, several changes in the experimental setup are necessary. The most important one concerns the selection of the lexicon. In order to deal with unconstrained texts, the lexicon must be large enough to have a high probability of containing the words appearing in the text. On the other hand, if the dictionary entries are too many, the recognition problem becomes too difficult. A good trade-off between these two effects must be found. This leads to lexica that do not have a full coverage of the data, i.e. not all of the words to be recognized are actually represented in the lexicon. This is an important difference with respect to single word recognition experiments where the lexicon is provided by information related to the handwritten samples (e.g. the zip code in postal address reading) and has a full coverage of the data.

Another important difference is in the way the performance of the system is measured. In single word recognition, a sample is correctly or incorrectly recognized (there is a single kind of error). In text recognition, there are several kinds of error. A word can be not only misclassified, but also deleted. Moreover, words not appearing in the text can be inserted during the decoding. This leads to different performance metrics that must be selected depending on the task the system is used for. Several experiments (changing the size of the lexicon from 1'000 to 50'000 and the order of the language model from 1 to 3) were performed over single writer data. The data set used is publicly available in order to allow a comparison with other systems.

The rest of the chapter is organized as follows: Section 6.2 provides the statistical foundations of our approach, Section 6.3 presents SLMs and N-gram models, Section 6.4 reports experiments and results obtained and the Section 6.5 draws some conclusions. This chapter corresponds to reference [144].

6.2 Statistical Foundations

This section describes the problem of handwritten text recognition from a statistical point of view. The handwritten pages are split into lines before the recognition and each line is decoded separately. For this reason, the recognition process is not applied to sentences, but rather to fragments of longer phrases. Since the system is supposed to recognize unconstrained texts, no information is available about the content of the data. The only hypothesis made about the lines is that they are written in English. They are then supposed to respect, on average, the statistics of any fragment of english text. The above aspects will be shown to have an important influence on the recognition results.

When the recognition is performed offline, only the image of the handwritten data is available. The image is converted into a sequence $O = (o_1, o_2, \ldots, o_m)$ of observation vectors and the recognition task can be thought of as finding a word sequence $\hat{W}$ maximizing the a-posteriori probability:

$$\hat{W} = \arg\max_{W} p(W|O) \tag{6.1}$$

where $W = (w_1, w_2, \ldots, w_n)$ is a sequence of words belonging to a fixed vocabulary V. By applying the Bayes theorem, Equation (6.1) can be rewritten as follows:

$$\hat{W} = \arg\max_{W} \frac{p(O|W)p(W)}{p(O)} \tag{6.2}$$

and since O is constant during recognition:

$$\hat{W} = \arg\max_{W} p(O|W)p(W). \tag{6.3}$$

The right side of Equation (6.3) shows the role of the different sources of information in the recognition problem. The term $p(O|W)$ is the probability of the observation sequence O being generated given the sentence W. This probability is estimated with HMMs.
If W is composed of n words and the size of the dictionary is $|V|$, then the number of possible word sequences is $|V|^n$. Even for small values of $|V|$ and n, this amounts to a huge number, making the task of the recognizer difficult. The term $p(W)$ provides an a-priori probability of the word sequence W being written and it is often estimated using a *Statistical Language Model.* A good SLM can significantly constrain the search space so that all the sentences that are unlikely to be written (from a linguistic point of view) have a low probability.
In the single word recognition problem, $p(W)$ is typically a uniform distribution over the words of the lexicon. This means that the recognition relies only on the HMMs and Equation (6.3) corresponds to:

$$\hat{w} = \arg\max_{w} p(O|w) \tag{6.4}$$

where w is a word belonging to the lexicon. In the next section, SLMs and N-gram models in particuar are described in detail.

6.3 Statistical Language Models

Statistical Language Modeling involves attempts to capture regularities of natural language in order to improve the performance of various natural language applications, e.g. Information Retrieval, Machine Translation and Document Classification [125]. This section is focused on the use of SLMs in our specific problem, i.e. the decoding of handwritten texts. As shown in Equation (6.3), the SLM is supposed to give the a priori probability of a certain sentence being written [76][78]. If W contains n words, $p(W)$ can be decomposed as follows:

$$p(W) = \prod_{i=1}^{n} p(w_i|w_1^{i-1}) = \prod_{i=1}^{n} p(w_i|h_i) \tag{6.5}$$

where $w_1^{i-1} = (w_1, \ldots, w_{i-1})$ and h_i is referred to as *history* of word i.
The fact that the probability of word w_i being written depends only on the previous words of the sentence makes decomposition in Equation (6.5) especially suitable for Viterbi decoding [147]. Using Viterbi, the observation sequence extracted from the handwritten data is decoded sequentially and it is necessary to take intermediate decisions before reaching its end. Equation (6.5) can be easily integrated in such a context. However, Equation (6.5) poses an important problem: for a vocabulary of reasonable dimension (in the order of tens of thousands), most of the possible histories appear too few times or even never. This does not allow the application of a statistical approach. The solution to such problem is to group the histories into a tractable number of equivalence classes. Equation (6.5) can then be rewritten as follows:

$$p(W) = \prod_{i=1}^{n} p(w_i|w_1^{i-1}) = \prod_{i=1}^{n} p(w_i|\Phi(h_i)) \tag{6.6}$$

where $\Phi : \{h\} \rightarrow C$ associates a history h to an equivalence class belonging to a finite set C.
The nature of $\Phi(h)$ allows one to distinguish between the different SLM techniques presented in the literature (see [125] for an extensive survey). In most cases, $\Phi(h)$ incorporates some linguistic knowledge. Attempts were made to replace the words w_i with corresponding classes $C(w_i)$ obtained through word clustering. Decision Trees, Classification and Regression Trees and different kinds of grammars were applied. However, such approaches are only moderately successful with respect to N-grams that are not specifically conceived for language modeling and could be used to model any sequence of symbols belonging to a finite alphabet [78][114][125].
In the next three subsections, N-gram models, smoothing techniques and an SLM performance measure are analyzed.

6.3.1 N-gram Language Models

An N-gram model makes an equivalence class out of all the histories ending with the same $N-1$ words:

$$p(W)=\prod_{i=1}^{n}p(w_i|w_{i-N+1}^{i-1}) \tag{6.7}$$

where N is called the *order* of the model. Even for low orders, the number of equivalence classes quickly becomes intractable. In practice, only unigrams, bigrams and trigrams are used. The probabilities $p(w_i|w_{i-N+1}^{i-1})$ are estimated by simply counting the number of times a certain sequence of N words appears in a corpus of training texts:

$$p(w_i|w_{i-N+1}^{i-1})=\frac{C(w_{i-N+1}^{i})}{C(w_{i-N+1}^{i-1})} \tag{6.8}$$

where $C(.)$ is the number of times the argument is counted.
This corresponds to a Maximum Likelihood (ML) estimation of the probabilities $p(w_i|w_{i-N+1}^{i-1})$. In other words, the model resulting from Equation (6.8) maximizes the likelihood over the training set. This leaves open an important problem: all the sequences of N words not appearing in the training text have zero probability. This is not correct because, in the case of unconstrained texts, no corpus is wide enough to contain all possible N-grams when N is big. Evaluations performed over a patent description dataset (around 1.8 million words) showed that, when splitting the data into training (1.5 million words) and test sets (0.3 million words), only 77% of the trigrams represented in the training set were also represented in the test set [76] (although the database is homogeneous). In order to solve this problem, the models are smoothed, i.e. the probability mass estimated with eq. (6.8) is redistributed across all possible sequences of N words. In such a way a model trained over a certain corpus can be used for any other text, but N-grams that are not possible from a linguistic point of view have non-zero probability. This is the main limitation of the N-gram models.

6.3.2 Smoothing

Smoothing is supposed to redistribute the probability mass estimated with ML across all possible sequences of N words. Many smoothing techniques have been proposed in the literature, but none of them seems to be systematically superior to the others [24]. In this work we selected a combination of *discounting* and *backing-off* following the schema proposed in [80].
The discounting is necessary because ML overestimates the frequences of the represented N-grams and correspondingly underestimates the frequences of the non-represented ones. After the discounting is applied, the frequence r of a certain word sequence is changed to r^* according to:

$$r^*=\frac{n_{r+1}}{n_r}(r+1) \tag{6.9}$$

where n_r is the number of events with frequence r. This is called Good-Turing discounting (originally proposed in [124]) and is based on the inverse proportionality between r and n_r. This property of the natural phenomena is known as Zipf Law and, in the case of texts, means that words appearing few times are more numerous than those appearing many times. The Good-Turing algorithm was selected because, being based on a natural law, it makes the discounting strategy more robust with respect to a change of data. Several discounting strategies are available in the literature and their performance is essentially equivalent [24].
The *amount of frequence* $r-r^*$ (summed up over all events) is redistributed across all possible N-grams not detected in the training text. In the most simple case, the redistribution is performed uniformly. Such a solution is not optimal because it does not take into account information present in the training corpus. For this reason, the redistribution is typically made through the so called

back-off. The trigram version of the back-off can be represented as follows:

$$p(w_i|w_{i-2}^{i-1}) = \begin{cases} p(w_i|w_{i-2}^{i-1}) & \text{if} \quad C(w_{i-2}^{i}) > 0 \\ \alpha_1 p(w_i|w_{i-1}) & \text{if} \quad C(w_{i-2}^{i}) = 0 \\ & \text{and} \quad C(w_{i-1}^{i}) > 0 \\ \alpha_2 p(w_i) & \text{otherwise.} \end{cases} \tag{6.10}$$

This means that, when an N-gram is not represented, lower order models are used. The coefficients α ensure that the probabilities of the non represented trigrams sum up to the amount of frequence to be redistributed. By using bigram and trigram statistics, the probability is no longer redistributed uniformly, but accordingly to the information extracted from the corpus.

6.3.3 The Perplexity

The perplexity (PP) is the measure most commonly used to evaluate the performance of a language model. The PP is estimated as follows:

$$PP = 2^H \tag{6.11}$$

where $H = \frac{1}{n}\sum_i p(w_i|h_i)$ is an estimate of the entropy of the model (measured over a text). The PP is the average branching factor of the model, i.e. the average number of words having a probability significantly higher than zero at each step of the decoding [78]. In a problem where all the words have the same probability of being written, the PP corresponds to the size of the lexicon. For this reason, the PP is often interpreted as the dictionary size in the case of single word recognition. Such interpretations show that an improvement of the model corresponds to a decrease of the perplexity.
The relationship between the PP of an SLM and the recognition rate of the system using it cannot be modeled clearly [87]. A decrease of the PP does not necessarily lead to better recognition rate for a system (it can even have negative effects) and vice-versa.

6.4 Experiments and Results

Experiments were performed using the system described in chapter 3. Although the system was developed to work on single words, no modifications were necessary to deal with texts. The data set used is composed of a text written by a single person and is publicly available on the web [1]. It will be referred to as *Cambridge* database. The Cambridge database was originally presented in [131] and contains 353 handwritten text lines split into training (153 lines), validation (83 lines) and test sets (117 lines). The lines are kept in the same order as they were written. This reproduces a realistic situation where the data available at a certain time is used to obtain a system capable of recognizing the data that will be written in the future.
The language models were obtained using the TDT-2 Corpus [61], a collection of news transcriptions obtained from several journals and broadcasting companies. The corpus is completely independent of the texts in the handwriting data sets. In this way, the language models are not fitted to the specific texts they have to model and the experimental setup reflects the actual condition of unconstrained text recognition.
In the next subsections we show in more detail how the lexicon was selected (section 6.4.1), how the language models were trained and tested (section 6.4.2) and the results obtained over the Cambridge (section 6.4.3) database.

6.4.1 Lexicon Selection

In single word recognition, the lexicon is always implicitly assumed to cover 100% of the data. Every handwritten word is supposed to be in the dictionary and this happens because the lexicon is determined from information coming from the application environment (e.g. the zip code in postal address

[1]The data can be downloaded at the following ftp address: `ftp.eng.cam.ac.uk/pub/data`.

recognition). This is not the case for the recognition of unconstrained texts. The presence of proper names, technical terms and morphological variations of a single stem makes it impossible to define a *universal* lexicon.

The only available source of information at the linguistic level is the text corpus we use to train the language models. It is therefore reasonable to extract the lexicon from it. Our corpus is 20'407'827 words in length. It is composed using 196'209 unique words. In order to build a lexicon of size M, we simply selected the M most frequent words in the lexicon. This approach is based on the hypothesis that the most frequent words in a text are typically *functional words* (prepositions, articles, conjunctions, verbs of common use, etc.), hence we can expect to find them in any other text. Moreover, the use of the most frequent words allows to obtain more reliable estimates of the N-gram probabilities. The plot in figure 6.1 shows the coverage (percentage of text words actually represented in the dictionary) of the lexica obtained with the above mentioned criterion in function of their size. The coverage is measured on the test set of the Cambridge database. The coverage represents, for a given lexicon size, the upper limit of system recognition performance.

6.4.2 N-gram Models Training

The N-gram models used in this work were trained over the TDT-2 corpus [61], a collection of transcriptions from several broadcast and newswire sources (ABC, CNN, NBC, MSNBC, Associated Press, New York Times, Voice of America, Public Radio International). For each lexicon described in section 6.4.1, three models (based on unigrams, bigrams and trigrams respectively) were created. The plots in figure 6.2 show the perplexities of the SLMs as a function of the lexicon size. The perplexity is estimated over the part of the text covered by the lexicon, without taking into account the out-of-vocabulary words.

A big improvement is obtained when passing from unigrams to bigrams, but no further improvement is obtained when applying trigrams. This happens for several reasons. The first is that the handwritten text is split into lines and only the words after the third one can take some advantages from the trigram model. Since a line contains in average 10 words, this means that only ~80% of the data can actually benefit from the trigram model (while 90% of the data can be modeled with bigrams).

A second problem, is that the percentage of trigrams covered by the corpus in the test set is ~40%. This further reduces the number of words where the trigram model can have a positive effect. The coverage in terms of bigrams is much higher (around 70%) and the percentage of words over which the model can have an effect is around 90%. For these reasons, the bigram and trigram models have a similar perplexity.

6.4.3 Cambridge Database Results

This section reports the results obtained on the Cambridge database. Once the language models are available (see previous section), it is necessary to train the HMMs. Since it is not possible to set a priori the number of states S and the number of Gaussians G in the models, a validation phase is necessary. Models with $10 \leq S \leq 14$ and $10 \leq G \leq 15$ are trained over the training set and tested, without using SLMs, over the validation set (The range of S and G is determined by the amount of available training material). The system corresponding to the couple (S, G) giving the best results (over the validation set) is selected as optimal. The system selected in the validation phase ($S = 12$ and $G = 12$) is retrained over the union of training and validation set and the resulting system is used in the actual recognition experiments.

For each lexicon, four versions of the system are tested over the test set. The first version (called *baseline*) makes no use of SLMs, the other ones use unigram, bigram and trigram models corresponding to the lexicon under consideration. The performance is measured using two different metrics: *accuracy* and *recognition rate*. In the recognition of a text, there are several sources of errors. A word can be misclassified and the corresponding error is called a *substitution*, but also split into two parts leading not only to an incorrect transcription, but also to the introduction of a word that does not exist in the original text. In this case the error is referred to as *insertion*. Finally, when the space between two words is missed, they are joined together giving rise to a substitution error and to the disappearance of a word existing in the original text. The error in this case is called a *deletion*. The different performance metrics depend on the sources of error that are taken into account. The accuracy is obtained as $100 - s$ (where s is the substitution rate). The accuracy is then the percentage of words correctly classified. The recognition rate is estimated as $100 - d - s - i$ where d and i are the deletion and insertion rates respectively. The recognition rate is not a percentage and can be negative. Its highest value is 100. When the task is a correct transcription of the text, the recognition rate is the most appropriate performance measure. It takes in fact into account insertions and deletions that represent important deviations with respect to the original text. When the task is related to content modeling (e.g. indexing), then the accuracy is a good metric since the only important factor is how many words are correctly classified.

In our experiments we used both metrics: figure 6.3 shows the recognition rate (left plot) and the accuracy (right plot) respectively as a function of the lexicon size. The plots also report the lexicon coverage that is the upper bound of both the recognition rate and the accuracy. In both cases, the performance is the result of a tradeoff between the improvement of the test set coverage and the increase of the lexicon size. The application of the N-gram models has a significantly positive effect on both recognition rate and accuracy (the statistical confidence is more than 90%). Moreover, the SLMs make the system more robust with respect to the increase of the lexicon size so that it is possible to maximize the benefit of the improved coverage.

For small lexica (less than 10'000 words) the systems using N-grams have a recognition rate lower than the baseline system (in the case of the 5'000 word lexicon, the difference is most probably due to statistical fluctuations). This happens because, when the coverage is low, the words not represented in the lexicon are easily split into shorter words giving rise to a high insertion error. For this reason, this phenomenon is not observed in the accuracy plot (the accuracy does not take the insertion error into account). The insertions are the reason for the difference between accuracy and recognition rate. Sometimes, when part of a word corresponds to a entry in the lexicon (e.g. *unmentionable* is composed of the entries *un*, *mention* and *able*) the decoder favours the transcription splitting the bigger word, especially when the shorter words are more frequently represented in the training corpus. No deletion error is observed. This is due to the fact that the spaces between neighboring words are typically evident, and are never missed (condition necessary to observe a deletion).

The systems using unigrams, bigrams and trigrams are equivalent in terms of performance. This is due, in our opinion, to the fact that the text lines are decoded separately, making it more difficult to take full advantage of the use of bigrams and trigrams (see end of section 6.4.2).

6.5 Conclusions

This chapter presented a system for the offline recognition of handwritten texts. The recognizer is based on continuous density Hidden Markov Models and Statistical Language Models (unigrams, bigrams and trigrams). Several experiments were performed using single writer data. The experimetal setup reproduces the conditions of unconstrained text recognition: no information about the content of the documents to be transcribed is used, except for the fact that they are written in English. The differences with respect to the case of single word recognition are highlighted. The recognition was performed over single writer data. Lexica of different sizes, (from 1'000 to 50'000 words) extracted on a statistical basis from the corpus used to train the language models, were used. The performance of the system was measured in terms of accuracy and recognition rate.

The results show that the application of SLMs significantly improves the performance of the system, independent of the metric used. However, the performance is not improved when increasing the order of the N-gram models. This depends, in our opinion, on our experimental setup which is difficult in two important aspects. The first is that the lines are decoded separately, so the models of higher order are effective on a smaller part of the data (see end of section 6.4.2). The second is that there is no alignment between the corpus used to train the SLMs and the texts of the training set. A better alignment can be obtained by using corpora more compatible with the texts to be recognized (e.g. news collections should be used to transcribe other news). This makes the system more constrained, but more effective. Moreover, a better alignment allows higher coverage to be obtained, given a certain lexicon size. The solution of the above mentioned problems can be an interesting direction for further work.

In its current configuration, the system appears to be more suitable for the application of content modeling techniques (Indexing, Information Retrieval, Topic Detection, etc.) than for the actual transcription of texts. All symbols other than letters are in fact not modeled and treated like noise. This is due to the fact that only the letters are represented sufficiently for a reliable training. In order to obtain a system more oriented toward the exact transcription of documents, it is necessary to increase the amount of training material (especially for the less represented symbols).

At the present state of the art in language modeling, it seems unlikely the possibility of having models performing better than N-grams. The application of other language modeling techniques (e.g. grammars or decision trees) seems to be limited to very constrained problems, like bank check reading, date processing or postal address recognition. The possibility of adapting generic N-gram models to specific problems can be explored in the same way that writer independent HMMs can be adapted to writer dependent data.

Since very few works have been dedicated to the offline recognition of handwritten texts, the above list of possible future directions is far from being exhaustive. The availability of systems for text recognition can give rise to new applications that were not considered until now. Especially from this point of view, the recognition of handwritten texts represents an interesting and promising research domain.

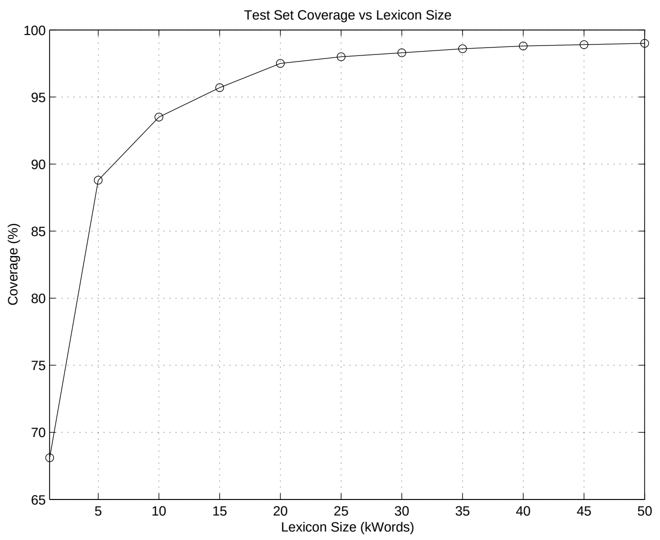


Figure 6.1: Test set coverage vs lexicon size. The plot shows the percentage of words covered by a lexicon in the test set as a function of the lexicon size.

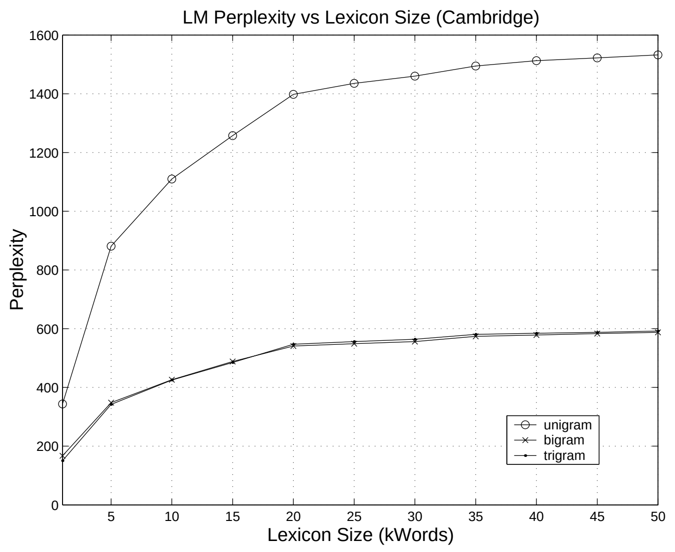
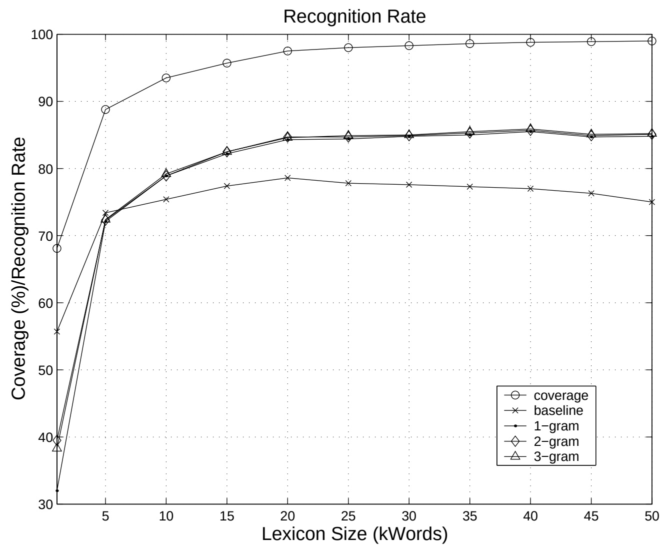


Figure 6.2: Language Model perplexity. The plot shows the perplexity of the language models over the test set of the Cambridge database as a function of the dictionary size.



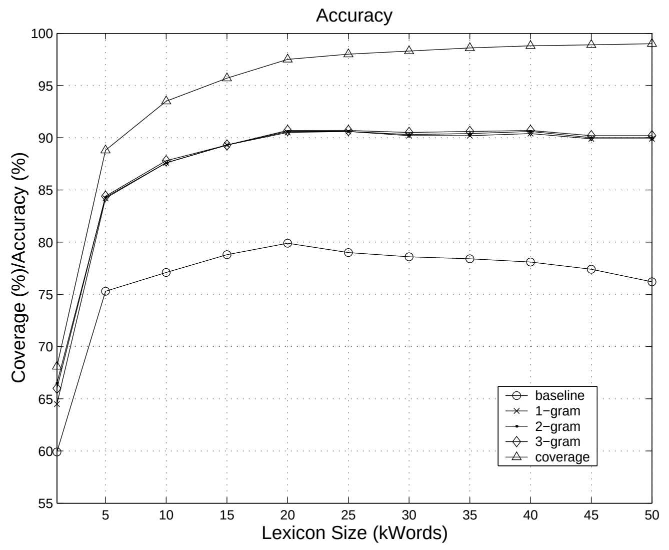


Figure 6.3: System performance. The upper (lower) plot shows the recognition rate (accuracy) of the system. The performance is measured over the test set.

Chapter 7

Conclusions

7.1 Conclusions and Future Work

This thesis presented the development of a system for offline cursive handwriting recognition. The effectiveness of the approaches, selected to solve the various encountered problems, has been shown through several experiments. In the following, the main contributions of this thesis and the related publications are highlighted.
The first step was an extensive survey of the domain literature [139]. This allowed us to establish the state of the art and to identify open issues still to be addressed.
The first problem we individuated is the use of many heuristic algorithms. These are often data dependent and typically use parameters that must be tuned empirically leading to a heavy experimental effort. Two steps of the processing are especially prone to this problem: segmentation and normalization. In the first version of our system (presented in [145]), we took care of this aspect.
In order to avoid the problems related to the segmentation, we applied a sliding window approach. In such architecture, there is no attempt to extract specific fragments of the word. The handwritten data is sampled at regular steps (in our case, at each column) and the fragments are isolated blindly. At the normalization level, we developed techniques for slope and slant removal based on statistics and fully adaptive. A description can be found in [146] where the new techniques are also shown to improve the recognition rate with respect to more traditional methods. Moreover, since our slope and slant removal algorithms make no use of hyperparameters, the heavy experimental effort needed to find an optimal configuration of the system is avoided.
The second aspect we considered is modeling. There is limited literature which discusses work where improvements of the performance are obtained through a better use of the HMMs. We explored two possibilities in this direction: the use of feature vector transforms making the data more suitable for the modeling and the application of HMM adaptation techniques.
We performed a comparison between several data transforms (linear and nonlinear Principal Component Analysis, Independent Component Analysis) showing that their application can significantly improve the performance of a system [141][142] over both single and multiple writer data.
The adaptation techniques (Maximum Likelihood Linear Regression, Maximum A Posteriori and their combination) were shown to be very effective when a writer dependent system is needed, but not enough writer dependent data is available for reliable training [140][143]. When less than 200 writer dependent samples are available, the systems obtained through adaptation perform better than those obtained by training directly over the data at disposition.
As a third direction to explore, we selected the extension from single word to text line recognition. Almost no work had been presented about this subject in the literature. The recognition of lines involves the use of language models (we used unigrams, bigrams and trigrams) and several changes in the experimental setup with respect to the recognition of single words.
The fourth and most important aspect we considered in all of the above work is the experimental

setup. This is very important in order to obtain realistic measures of the performance. It is frequent to find in the literature works where the separation between training and test set is not rigorous. In our experiments, we were very careful about this point. All the hyperparameters were set using a validation set and the information in the test set was never used to build the system.
Several interesting issues remain open and can be the subject of further work. The list proposed here is not exhaustive but aims to show possible research directions. In future, the focus will be on the recognition of texts which has been developed less than the recognition of single words. The limit due to the fact that the text is split into lines should be overcome. The last words of a line can be used to help the bigram and trigram modeling of the following. Another problem is the alignment between the language model and the text to be recognized. There are several ways that are worth studying to obtain such an effect. The first one is to perform an adaptation in the same way as in the case of the writer adaptation. A small amount of data is used to fit a generic language model to a specific problem. Another way, suitable when the text to be transcribed is long, is to recognize part of the data, analyze its content and then route the data to a topic dependent system. In conclusion, the use of language modeling, that has been completely neglected until now in offline handwriting recognition, still leaves open many possibilities for further improvement.

Bibliography

[1] N. Arica and F.T. Yarman-Vural. Optical character recognition for cursive handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):801–813, 2002.

[2] P. Baldi and K. Hornik. Learning in linear neural networks: a survey. *IEEE Transactions on Neural Networks*, 6(4):837–858, 1995.

[3] C. Barriere and R. Plamondon. Human identification of letters in mixed script handwriting: An upper bound on recognition rates. *IEEE Transactions on Systems, Man and Cybernetics B*, 28(1):78–82, february 1998.

[4] L.E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[5] L.E. Baum and J.E. Egon. An inequality with applications to statistical estimation for probabilistic function of a Markov process and to a model for ecology. *Bulletin of American Metheorological Society*, 73:360–363, 1967.

[6] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[7] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[8] L.E. Baum and G.R. Sell. Growth functions for transformation on manifolds. *Pacific Journal Mathematics*, 27(2):211–227, 1968.

[9] R.E. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

[10] R.E. Bellman and S.E. Dreyfus. *Applied Dynamic Programming.* Princeton University Press, 1962.

[11] Y. Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 2:129–162, 1999.

[12] C. Bishop. *Neural Networks for Pattern Recognition.* Cambridge University Press, Cambridge, UK, 1995.

[13] V. Bouletreau, N. Vincent, N. Sabourin, and H. Emptoz. Synthetic parameters for handwriting classification. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 102–106, Ulm, 1997.

[14] H. Bourlard and S. Bengio. Hidden markov models and other finite state automata for sequence processing. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks.* MIT Press, Cambridge (MA), USA, 2002.

[15] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[16] Radmilo M. Bozinovic and Sargur N. Srihari. Off-line cursive script word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):69–83, January 1989.

[17] H. Bunke, M. Roth, and E.G. Schukat-Talamazzini. Off-line recognition of cursive script produced by a cooperative writer. In *Proceedings of International Conference on Pattern Recognition*, pages 383–386, 1994.

[18] H. Bunke, M. Roth, and E.G. Schukat-Talamazzini. Off-line cursive handwriting recognition using Hidden Markov Models. *Pattern Recognition*, 28(9):1399–1413, September 1995.

[19] J. Cai and Z.Q. Liu. Off-line unconstrained handwritten word recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(3):259–280, 2000.

[20] F. Camastra and A. Vinciarelli. Cursive character recognition by Learning Vector Quantization. *Pattern Recognition Letters*, 22(6-7):625–629, 2001.

[21] G. Casey and E. Lecolinet. Strategies in character segmentation: A survey. In *Proceedingsof International Conference on Document Analysis and Recognition*, volume 1, pages 1028–1033, Montreal, 1995.

[22] M.Y. Chen and A. Kundu. An alternative to variable duration HMM in handwritten word recognition. In *Proceedings of International Workshop on Frontiers in Handwriting Recognition*, 1993.

[23] M.Y. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition using a Hidden Markov Model type stochastic network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):481–496, May 1994.

[24] S. Chen and R. Rosenfeld. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50, january 2000.

[25] W. Chen, P. Gader, and H. Shi. Lexicon-driven handwritten word recognition using optimal linear combinations of order statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):77–82, January 1999.

[26] W.T. Cheng and P. Gader. Word level discriminative training for handwritten word recognition. In *Proceedings of International Workshop on Frontiers in Handwriting Recognition*, pages 393–402, Amsterdam, 2000.

[27] E. Cohen, J.J. Hull, and S.N. Srihari. Control structure for interpreting handwritten addresses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10), 1994.

[28] F.S. Cohen. Markov Random Fields for image modelling e analysis. In *Modelling and Applications of Stochastic Processes*, pages 243–272. Kluwer Academic Press, 1986.

[29] M. Coltheart and K. Rastle. Serial processing in reading aloud: evidence for dual-route models in reading. *Journal of Experimental Psychology*, 20:1197–1211, 1994.

[30] P. Comon. Independent Component Analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

[31] M. Coté. *Utilisation d'un modéle d'accés lexical et de concepts perceptifs pour la reconnaissance d'images de mots cursifs.* PhD thesis, Ecole Nationale Supérieure des télécommunications de Paris, France, 1997.

[32] M. Coté, M. Cheriet, E. Lecolinet, and C.Y. Suen. Automatic reading of cursive scripts using human knowledge. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 107–111, Ulm, 1997.

[33] M. Côté, E. Lecolinet, M. Cheriet, and C.Y. Suen. Building a perception based model for reading cursive script. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 898–901, Montreal, 1995.

[34] M. Cote', E. Lecolinet, M. Cheriet, and C.Y. Suen. Automatic reading of cursive scripts using a reading model and perceptual concepts - the PERCEPTO system. *International Journal of Document Analysis and Recognition*, 1(1):3–17, january 1998.

[35] J.P. Crettez. A set of handwriting families: Style recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 489–494, Montreal, 1995.

[36] D.P. D'Amato, E.J. Kuebert, and A. Lawson. Results from a performance evaluation of handwritten address recognition systems for the United States Postal Service. In *Proceedings of International Workshop on Frontiers in Handwriting Recognition*, pages 189–198, Amsterdam, 2000.

[37] M. De Groot. *Optimal statistical Decisions.* McGraw-Hill, New York, 1970.

[38] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood estimation from incomplete data. *Journal of the Royal Statistical Society (B)*, 39(1):1–38, 1977.

[39] V. Di Lecce, A. Dimauro, Guerriero, S. Impedovo, G. Pirlo, and A. Salzo. A new hybrid approach for legal amount recognition. In *Proceedings of International Workshop on Frontiers in Handwriting Recognition*, pages 199–208, Amsterdam, 2000.

[40] V. Digalakis, D. Rtischev, and L. Neumayer. Speaker adaptation using constrained estimation of gaussian mixtures. *IEEE Transactions on Speech and Audio Processing*, 3(5):357–366, September 1995.

[41] G. Dimauro, S. Impedovo, and G. Pirlo. Automatic recognition of cursive amounts on italian bank-checks. In S. Impedovo, editor, *Progress in Image Analysis and Processing III*, pages 323–330. World Scientific, 1994.

[42] G. Dimauro, S. Impedovo, G. Pirlo, and A. Salzo. Bankcheck recognition systems: re-engineering the design process. In A. Downton and S. Impedovo, editors, *Progress in Handwriting Recognition*, pages 419–425.

[43] G. Dimauro, S. Impedovo, G. Pirlo, and A. Salzo. Automatic bankcheck processing: A new engineered system. In *Automatic Bankcheck Processing*, pages 5–42. World Scientific Publishing, 1997.

[44] Y. Ding, F. Kimura, Y. Miyake, and M. Shridar. Slant estimation for handwritten words by directionally refined chain code. In *Proceedings of International Workshop on Frontiers in Handwriting Recognition*, pages 53–62, Amsterdam, 2000.

[45] J.P. Dodel and R. Shinghal. Symbolic/neural recognition of cursive amounts on bank cheques. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 15–18, Montreal, 1995.

[46] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* John Wiley & Sons, New York, NY, 2000.

[47] S. Edelman, T. Flash, and S. Ullman. Reading cursive handwriting by alignment of letter prototypes. *International Journal of Computer Vision*, 5(3):303–331, March 1990.

[48] A. El Yacoubi, J.M. Bertille, and M. Gilloux. Conjoined location and recognition of street names within a postal address delivery line. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 1024–1027, Montreal, 1995.

[49] A. El-Yacoubi, M. Gilloux, R. Sabourin, and C.Y. Suen. An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):752–760, August 1999.

[50] J.T. Favata. General word recognition using approximate segment-string matching. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 92–96, Ulm, 1997.

[51] G.D. Forney. The Viterbi algorithm. *Proceedings of IEEE*, 61(3):268–278, 1973.

[52] D. Foteringhame and R. Baddeley. Nonlinear principal components analysis of neuronal spike train data. *Biological Cybernetics*, 77(4):282–288, 1997.

[53] T. Freche and N. Vincent. Local and global approaches to achieve quantitative measurement of handwritings. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 593–596, Bangalore, 1999.

[54] K. Fukunaga. *Statistical Pattern Recognition.* Academic Press, 1990.

[55] P.D. Gader, M. Mohamed, and J.H. Chiang. Handwritten word recognition with character and inter-character neural networks. *IEEE Transactions on Systems, Man and Cybernetics B*, 27(1):158–164, february 1997.

[56] P.D. Gader, M.A. Mohamed, and J. Chiang. Comparison of crisp and fuzzy character neural networks in handwritten word recognition. *IEEE Transactions on Fuzzy Systems*, 3(3):357–364, 1995.

[57] M.J.F. Gales. The generation and use of regression class trees for MLLR adaptation. Technical Report TR263, Cambridge University Engineering Department, 1996.

[58] J.L. Gauvain and C.H. Lee. MAP estimation of Continuous Density HMM: theory and applications. In *Proceedings of DARPA Speech and Natural Language Workshop*. Morgan Kaufmann, 1992.

[59] J.L. Gauvain and C.H. Lee. Maximum a Posteriori estimation for multivariate gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, april 1994.

[60] N. Gorski, V. Anisimov, E. Augustin, O. Baret, D. Price, and J.C. Simon. A2iA check reader: A family of bank check recognition systems. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 523–526, Bangalore, 1999.

[61] D. Graff, C. Cieri, S. Strassel, and N. Martey. The TDT-3 text and speech corpus. In *Proceedings of Topic Detection and Tracking Workshop*, 2000.

[62] D. Guillevic. *Unconstrained Handwriting Recognition Applied to the Processing of Bank Cheques.* PhD thesis, Concordia University, Montreal, Canada, 1995.

[63] D. Guillevic and C.Y. Suen. Cursive script recognition: A sentence level recognition scheme. In *Proceedins of 4^{th} International Workshop on Frontiers in Handwriting Recognition*, pages 216–223, 1994.

[64] D. Guillevic and C.Y. Suen. Cursive script recognition applied to the processing of bank cheques. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 11–14, Montreal, 1995.

[65] D. Guillevic and C.Y. Suen. HMM word engine recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 2, pages 544–547, Ulm, 1997.

[66] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision*. Addison Wesley, USA, 1992.

[67] J.J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

[68] A. Hyvärinen. A fast fixed-point algorithm for Independent Component Analysis. *Neural Computation*, 9(7):1483–1492, 1997.

[69] A. Hyvärinen. Fast and robust fixed-point algorithms for Independent Component Analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.

[70] A. Hyvärinen. Survey on Independent Component Analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[71] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley and Sons, New York, NY, 2001.

[72] A. Hyvärinen and E. Oja. Independent Component Analysis: A tutorial. *Neural Networks*, 13(4-5):411–430, 2000.

[73] S. Impedovo, editor. *Fundamentals in Handwriting Recognition*, chapter Hidden Markov Models in Handwriting Recognition. Springer, 1994.

[74] A.M. Jacobs and J. Grainger. Models of visual word recognition - sampling the state of the art. *Journal of Experimental Psychology*, 20(6):1311–1334, 1994.

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

[76] F. Jelinek. Self-organized language modeling for speech recognition. In A. Waibel and L. Kai-Fu, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann, Palo Alto, CA, 1989.

[77] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[78] F. Jelinek. *Statistical Aspects of Speech Recognition*. MIT Press, 1998.

[79] N.L. Johnson and S. Kotz. *Distribution in Statistics*. John Wiley & Sons, New York, NY, 1972.

[80] S.M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, 1987.

[81] G. Kaufmann and H. Bunke. Automated reading of cheque amounts. *Pattern Analysis and Applications*, 3:132–141, march 2000.

[82] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis. A slant removal algorithm. *Pattern Recognition*, 33(7):1261–1262, july 2000.

[83] G. Kim and V. Govindaraju. Handwritten word recognition for real time applications. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 24–27, Montreal, 1995.

[84] G. Kim and V. Govindaraju. A lexicon driven approach to handwritten word recognition for real time application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):366–379, april 1997.

[85] G. Kim, V. Govindaraju, and S.N. Rihari. An architecture for handwritten text recognition systems. *International Journal of Document Analysis and Recognition*, 2(1):37–44, 2000.

[86] M. Kirby. *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns.* John Wiley and Sons, New York, NY, 2001.

[87] D. Klakow and J. Peters. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38:19–28, 2002.

[88] S. Knerr, V. Anisimov, O. Baret, N. Gorski, D. Price, and J.C. Simon. The A2iA Interchecque system: courtesy amount and legal amount recognition for french checks. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(4), 1997.

[89] S. Knerr, E. Augustin, O. Baret, and D. Price. Hidden Markov Model based word recognition and its application to legal amount reading on french checks. *Computer Vision and Image Understanding*, 70(3):404–419, June 1998.

[90] A. Kundu, Y. He, and M.Y. Che. Alternatives to variable duration HMM in handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1275–1280, November 1998.

[91] B. Lazzerini, F. Marcelloni, and L.M. Reyneri. Beatrix: a self-learining system for off-line recognition of handwritten texts. *Pattern Recognition Letters*, 18:583–594, 1997.

[92] Y. Le Cun and Y. Bengio. Pattern recognition and neural networks. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks.* MIT Press, 1995.

[93] C.H. Lee and J.L. Gauvain. Speaker adaptation based on MAP estimation of HMM parameters. In *Proceedings of IEEE Conference on Audio Speech and Signal Processing*, volume 2, pages 558–561, 1993.

[94] C.J. Leggetter and P.C. Woodland. Flexible speaker adaptation for large vocabulary speech recognition. In *Proceedings of 4^{th} European Conference on Speech Communication and Technology*, pages 1155–1158, 1995.

[95] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density HMMs. *Computer Speech and Language*, 9:171–185, 1995.

[96] B. Lerner, H. Guterman, M. Aladjem, and I. Dinstein. A comparative study of neural network based feature extraction paradigms. *Pattern Recognition Letters*, 20(1):1999, 1999.

[97] Y. Lu and M. Shridar. Character segmentation in handwritten words - an overview. *Pattern Recognition*, 1996.

[98] S. Madhvanath, G. Kim, and V. Govindaraju. Chaincode contour processing for handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):928–932, September 1999.

[99] S. Madhvanath, E. Kleinberg, and V. Govindaraju. Holistic verification of handwritten phrases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999.

[100] S. Madhvanath, E. Kleinberg, V. Govindaraju, and S.N. Srihari. The HOVER system for rapid holistic verification of off-line handwritten phrases. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 2, pages 855–859, Ulm, 1997.

[101] S. Madhvanath and V. Kripasundar. Pruning large lexicons using generalized word shape descriptors. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 2, pages 552–555, Ulm, 1997.

[102] E.C. Malthouse. Limitations of nonlinear PCA as performed with generic neural networks. *IEEE Transactions on Neural Networks*, 9(1):165–173, january 1998.

[103] J. Mao and Jain A.K. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, 1995.

[104] U. Marti and H. Bunke. Towards general cursive script recognition. In *Proceedings of International workshop on Frontiers in Handwriting Recognition*, pages 379–388, Korea, 1998.

[105] U. Marti, G. Kaufmann, and Bunke H. Cursive script recognition with time delay neural networks using learning hints. In W. Gerstner, A. Gernoud, M. Hasler, and JournalD. Nicoud, editors, *Artificial Neural Networks - ICANN97*, pages 973–979. Springer Verlag, october 1997.

[106] U.V. Marti and H. Bunke. A full english sentence database for off-line handwriting recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 705–708, Bangalore, 1999.

[107] U.V. Marti and H. Bunke. Handwritten sentence recognition. In *Proceedings of International Conference on Pattern Recognition*, volume 3, pages 467–470, Barcelona, 2000.

[108] U.V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):65–90, 2001.

[109] U.V. Marti and H. Bunke. The IAM-database: an english sentence database for offline handwriting recognition. *International Journal of Document Analysis and Recognition*, 5(1):39–46, january 2002.

[110] J.L. McClelland and D.E. Rumelhart. An interactive activation model of context effects in letter perception. *Psychological Review*, 88:375–407, 1981.

[111] M. Mohamed and P. Gader. Handwritten word recognition using segmentation-free Hidden Markov Modeling and segmentation-based Dynamic Programming techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):548–554, May 1996.

[112] J. Moreau. A new system for automatic reading of postal checks. In *Proceedings of International Workshop on Frontiers in Handwriting Recognition*, pages 121–132, 1991.

[113] M. Morita, J. Facon, F. Bortolozzi, S. Garnes, and R. Sabourin. Mathematical morphology and weighted least squares to correct handwriting baseline skew. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 430–433, Bangalore, 1999.

[114] H. Ney. Corpus-based statistical methods in speech and language processing. In S. Young and G. Bloothoft, editors, *Corpus-based methods in language and speech processing*, pages 1–26. Kluwer Academic Publishers, The Netherlands, 1997.

[115] C. Olivier, T. Paquet, M. Avila, and Y. Lecourtier. Recognition of handwritten words using stochastic models. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 19–23, Montreal, 1995.

[116] K. Paap, S.L. Newsome, J.E. McDonald, and R.W. Schvaneveldt. An activation-verification model for letter and word recognition: the word superiority effect. *Psychological Review*, 89:573–594, 1982.

[117] T. Paquet and Y. Lecourtier. Recognition of handwritten sentences using a restricted lexicon. *Pattern Recognition*, 26(3):391–407, 1993.

[118] C. Parisse. Global word shape processing in off-line recognition of handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):460–464, April 1996.

[119] J. Park, V. Govindaraju, and S.N. Srihari. Efficient word segmentation driven by unconstrained handwritten phrase recognition. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 605–608, Bangalore, 1999.

[120] R. Plamondon and S.N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[121] L. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In A. Waibel and L. Kai-Fu, editors, *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann, Palo Alto, CA, 1989.

[122] L. Rabiner and B.H. Huang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[123] B.D. Ripley. Statistical aspects of neural networks. In Journal Bornndorff-Nielsen, Journal Jensen, and W. Kendal, editors, *Networks on Chaos: Statistical and Probabilistic Aspects.* Chapman and Hall, 1993.

[124] R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228, 1996.

[125] R. Rosenfeld. Two decades of statistical language modeling: where do we go from here? *Proceedings of IEEE*, 88(8):1270–1278, august 2000.

[126] J.C. Salome, M. Leroux, and J. Badard. Recognition of cursive script words in a small lexicon. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 774–782, 1991.

[127] G. Saon. Cursive word recognition using a random field based Hidden Markov Model. *International Journal of Document Analysis and Recognition*, 1(1):199–208, 1999.

[128] K.M. Sayre. Machine recognition of handwritten words: a project report. *Pattern Recognition*, 5(3):213–228, 1973.

[129] G. Seni, V. Kripasundar, and R.K. Srihari. Generalizing edit distance to incorporate domain information: Handwritten text recognition as a case study. *Pattern Recognition*, 1996.

[130] A. W. Senior. *Off-Line Cursive Handwriting Recognition Using Recurrent Neural Network.* PhD thesis, University of Cambridge, UK, 1994.

[131] A.W. Senior and A.J. Robinson. An off-line cursive handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):309–321, March 1998.

[132] M. Shridar, G. Houle, and Kimura F. Handwritten word recognition using lexicon free and lexicon directed word recognition algorithms. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 2, pages 861–865, Ulm, 1997.

[133] R.K. Srihari. Use of lexical and syntactic techniques in recognizing handwritten text. In *Proceedingsof ARPA workshop on Human Language Technology*, pages 403–407, Princeton, 1994.

[134] R.K. Srihari and C. Baltus. Incorporating syntactic constraints in recognizing handwritten sentences. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1262–1267, Chambery, 1993.

[135] S.N. Srihari. Handwritten address interpretation: a task of many pattern recognition problems. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(5):663–674, 2000.

[136] T. Steinherz, E. Rivlin, and N. Intrator. Off-line cursive script word recognition - a survey. *International Journal on Document Analysis and Recognition*, 2(2):1–33, February 1999.

[137] Y. Tay, P. Lallican, M. Khalid, C. Viard-Gaudin, and S. Knerr. An offline cursive handwritten word recognition system. In *Proceedings of IEEE Region 10 Conference*, 2001.

[138] C. Viard-Gaudin, P.M. Lallican, S. Knerr, and P. Binter. The IRESTE on/off (IRONOFF) dual handwriting database. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, pages 455–458, Bangalore, 1999.

[139] A. Vinciarelli. A survey on off-line cursive word recognition. *Pattern Recognition*, 35(7):1433–1446, june 2002.

[140] A. Vinciarelli and S. Bengio. Writer adaptation techniques in HMM based off-line cursive script recognition. *Pattern Recognition Letters*, 23(8):905–916, 2001.

[141] A. Vinciarelli and S. Bengio. Off-line cursive word recognition using continuous density HMMs trained with PCA ir ICA features. In *Proceedings of 16^{t}h International Conference on Pattern Recognition*, volume III, pages 81–84, 2002.

[142] A. Vinciarelli and S. Bengio. Transforming feature vectors to improve HMM based offline cursive word recognition systems. Technical Report 32, IDIAP, 2002.

[143] A. Vinciarelli and S. Bengio. Writer adaptation techniques in HMM based off-line cursive script recognition. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, pages 287–291, 2002.

[144] A. Vinciarelli, S. Bengio, and H. Bunke. Offline recognition of large vocabulary cursive handwritten text. Technical Report 01, IDIAP, 2003.

[145] A. Vinciarelli and J. Lüttin. Off-line cursive script recognition based on continuous density HMM. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, pages 493–498. World Publishing, 2000.

[146] A. Vinciarelli and J. Lüttin. A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22(9):1043–1050, 2001.

[147] A.J. Viterbi. Error bounds for convolutional codes and an asimptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–269, 1967.

[148] W. Wang, A. Brakensiek, A. Kosmala, and G. Rigoll. HMM based high accuracy off-line cursive handwriting recognition by a baseline detection error tolerant feature extraction approach. In *Proceedings of International Workshop on Frontiers in Handwriting Recognition*, pages 209–218, Amsterdam, 2000.

[149] B.A. Yanikoglu and P.A. Sandon. Off line cursive handwriting recognition using neural networks. In *Proceedings of SPIE Conference on Applications of Artificial Neural Networks*, 1993.

[150] B.A. Yanikoglu and P.A. Sandon. Off-line cursive handwriting recognition using style parameters. *Technical Report PCS-TR93-192 Dartmouth College*, April 1993.

[151] M. Zimmermann and H. Bunke. Automatic segmentation of the IAM off-line database for handwritten english text. In *Proceedings of 16th International Conference on Pattern Recognition*, 2002.

[152] M. Zimmermann and J. Mao. Lexicon reduction using key characters in cursive handwritten words. *Pattern Recognition Letters*, 20:1297–1304, 1999.