Using a Statistical Language Model to Improve the Performance of an HMM-based Cursive Handwriting Recognition System

U.-V. Marti and H. Bunke
Institut für Informatik und angewandte Mathematik
Universität Bern, Neubrückstrasse 10, CH-3012 Bern
Switzerland
email:{marti,bunke}@iam.unibe.ch

August 25, 2000

Abstract

In this paper a system for the reading of totally unconstrained handwritten text is presented. The kernel of the system is a hidden Markov model (HMM) for handwriting recognition. This HMM is enhanced by a statistical language model. Thus linguistic knowledge beyond the lexicon level is incorporated in the recognition process. Another novel feature of the system is that the HMM is applied in such a way that the difficult problem of segmenting a line of text into individual words is avoided. A number of experiments with various language models and large vocabularies have been conducted. The language models used in the system were also analytically compared based on their perplexity.

Keywords: cursive handwriting recognition, offline handwriting recognition, unconstrained sentence recognition, hidden Markov model, statistical language model, bigram statistics.

1 Introduction

The area of handwriting recognition can be divided into online and offline [1, 2, 3]. In online recognition the writer is connected via an electronic pen or a mouse to the computer and the handwriting is recorded as a function of time. By contrast, in offline recognition the handwriting is captured by means of a scanner and becomes available for processing and recognition in form of an image. Because

of the availability of temporal information, online recognition is often considered the easier problem. In this paper we exclusively focus on offline recognition.
Research in offline handwriting recognition has a history of more than thirty years now. The first systems were restricted to the reading of isolated characters [4]. Thus for languages using Roman characters, the number of classes to be recognized was rather small. Later the recognition of cursively written words became a focus of attention [1, 5, 6, 7, 8]. But for most of the applications considered until now, constraints exist that make the recognition problem easier. For example, in postal address reading [9, 10, 11, 12] the number of possible destination names is limited. Moreover, ZIP code and destination name are redundant to each other. Similar constraints exist for bank check and form reading [13, 14, 15, 16].
It was only recently that research on reading totally unconstrained handwritten text started. In [17, 18] the authors considered the problem of reading whole paragraphs and pages of handwritten text. Applications of this type of problem are automatic reading of historical archives or electronic assistants for the reading of personal notes.
In the present paper we describe a system for the reading of unconstrained handwritten pages. The core of the system is a hidden Markov model (HMM) for text recognition [19, 20, 21]. There are three features in which this system differs from others published in the literature. First the HMM is applied in such a way that the difficult problem of segmenting the text into individual words is avoided, i.e., only complete lines of text have to be extracted from the image of a handwritten page and input to the recognizer. Secondly, a statistical language model is used to improve recognition performance. In most systems known today, a lexicon is the only source of information about the underlying language used in the recognition process, but no knowledge beyond the lexical level is applied. By contrast, we incorporate statistical knowledge about the occurrence and coocurrence probability of the words in the lexicon in order to constrain the search in the HMM. Thirdly, the underlying vocabulary is significantly larger than those used in other systems [17, 18]. The techniques that are actually used in the system described in this paper are similar, to some degree, to those applied in speech recognition [22, 23].
In the next section an overview of the system is given. Then the preprocessing of the text images is described in detail. Feature extraction is the topic of Section 4. Section 5 considers the HMM on which the recognizer is based. The statistical language model is described in Section 6. Experimental results are presented in the Section 7 and conclusions are drawn in the final section.

2 System Overview

The system described in this paper follows the classical architecture. It consists of three main modules: the preprocessing, where noise reduction and normalization

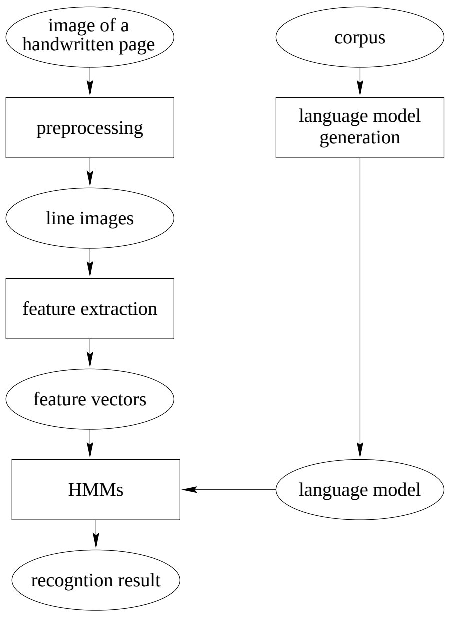


Figure 1: System overview.

take place, the feature extraction, where the image of a handwritten text is transformed into a sequence of numerical feature vectors, and the recognizer, which converts these sequences of feature vectors into word classes. A fourth part, the language model, provides linguistic knowledge about the context in which a word is likely to occur (see Fig. 1).
The first step in the processing chain, the preprocessing, is mainly concerned with text image normalization. The goal of the different normalization steps is to produce a uniform image of the writing with less variations of the same character or word across different writers (see Sec. 3). The aim of feature extraction is to derive a sequence of feature vectors which describe the writing in such a way that different characters and words can be distinguished, at the same time avoiding redundant information as much as possible. In the presented system the features are based on geometrical measurements (see Sec. 4). At the core of the recognition procedure is an HMM. It receives a sequence of feature vectors as input and outputs a sequence of words, applying constraints from the language model. The feature vectors input to the HMM are extracted from a complete line of text. Thus no segmentation of a line of text into individual words is required. Rather, this segmentation is obtained during the recognition process.
In a second, offline processing chain a model of the language is derived. Here we start with a text collection called corpus. The model defines possible word sequences and introduces linguistic knowledge into the recognition task (see Sec. 6).

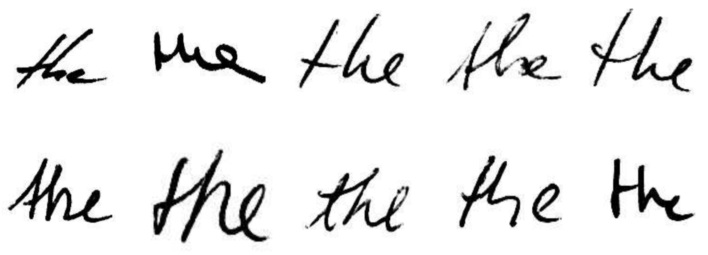


Figure 2: Ten instances of word *the* illustrating different writing styles.

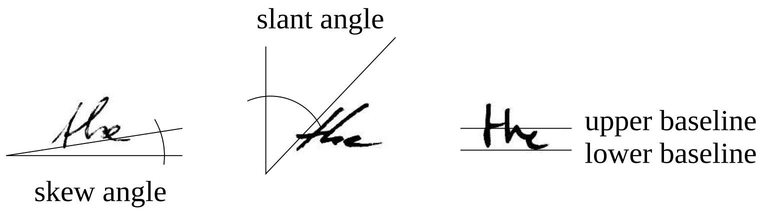


Figure 3: Writing characteristics: skew angle, slant angle and baselines.

3 Preprocessing

Each person has a different writing style with its own characteristics (see Fig. 2). This fact makes the recognition task complicated. To reduce variations in the handwritten texts as much as possible, a number of preprocessing operations are applied. In the presented system the following five preprocessing steps are carried out:

Line Separation: The given page is divided into lines.

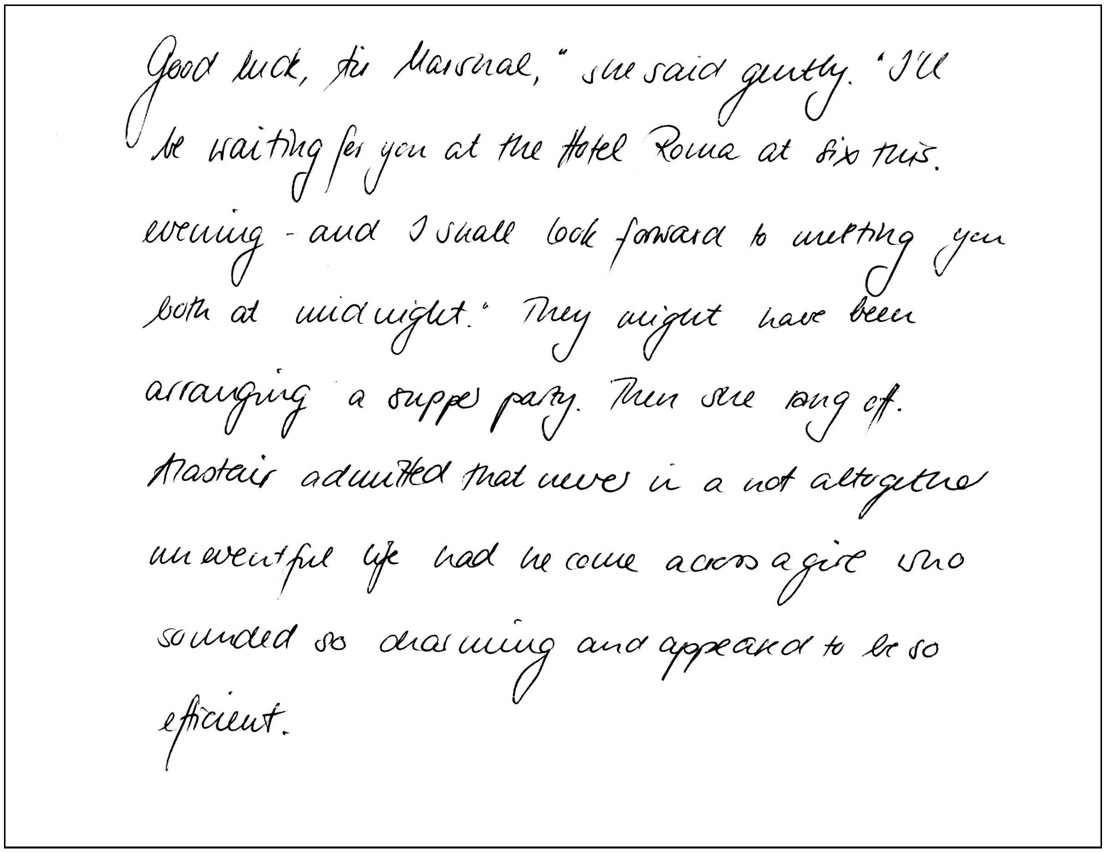
Skew Correction: The text line is horizontally aligned. For this purpose the skew angle needs to be determined (see Fig. 3, left part).

Slant Correction: Applying a shear transformation, the writing's slant is transformed into an upright position (see Fig. 3, middle part).

Line Positioning: The text line's total extent in vertical direction is normalized to a standard value. Moreover, applying a vertical scaling operation the location of the upper and lower baseline (see Fig. 3, right part) is set to a standard position.

Horizontal Scaling: The variations in the width of the writing are normalized.

Note that the order of these operations is not arbitrary. Because the nonlinear scaling in line positioning and horizontal scaling affect angles, skew and slant have to be corrected before positioning and scaling. Correcting the skew of the writing



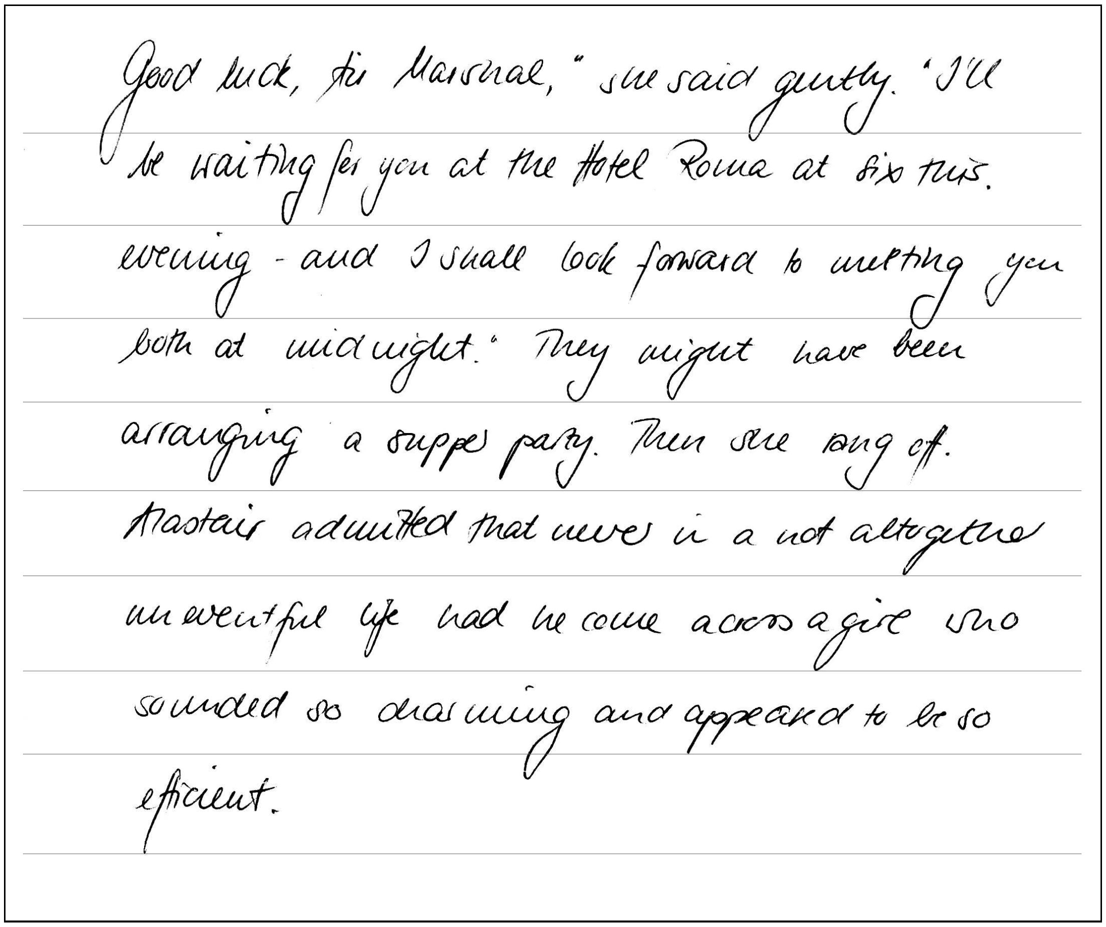
Good luck, Sir Marshal," she said gently. "I'll
be waiting for you at the Hotel Roma at six this
evening - and I shall look forward to meeting you
both at midnight." They might have been
arranging a supper party. Then she rang off.
Alastair admitted that never in a not altogether
uneventful life had he come across a girl who
sounded so charming and appeared to be so
efficient.

Figure 4: Image of a handwritten text paragraph.

means a rotation of the image. This changes also the slant angle. Therefore the skew has to be corrected before the slant.
The following five subsections will describe the preprocessing procedures in detail. They are an important prerequisite for feature extraction and recognition.

3.1 Line Separation

The images of the text to be recognized consist of whole pages and paragraphs of text. An example is given in Fig. 4. The first step in preprocessing aims at segmenting a page into single text lines.
Because the writers had to put rulers under the form while they produced the handwriting, the lines are more or less straight and the characters in two consecutive lines usually don't touch each other. This makes line segmentation easier. The implemented method is based on a histogram where we count the black-white transitions in each horizontal image row. The raw histogram is smoothed by a median filter to eliminate outliers and noise. Local minima are potential cuts between two consecutive text lines. In Fig. 5 these lines are superimposed on the image shown in Fig. 4.



"Good luck, Sir Marshal," she said gently. "I'll
be waiting for you at the Hotel Roma at six this
evening - and I shall look forward to meeting you
both at midnight." They might have been
arranging a supper party. Then she rang off.
Alastair admitted that never in a not altogether
uneventful life had he come across a girl who
sounded so charming and appeared to be so
efficient.

Figure 5: Text paragraph with cutting lines.

If a local minimum in the histogram has a value of zero, it follows that at the corresponding image line no text components are intersected. On the other hand, if the minimum value is greater than zero, segmentation is not possible without cutting the handwritten text. But we know that the chosen position is the one that has the minimum number of intersections in the local neighborhood.
If an intersection is encountered, the center of gravity of the connected component that is cut by the considered horizontal line is determined. There are three possible cases. First, if the center of gravity is above the cutting line at a certain distance $d \geq T$, it is assumed that the touching component belongs to the previous text line. Otherwise, if it is located below the cutting line at distance $d \geq T$ the connected component is assigned to the succeeding text line. In the third case where the center of gravity is located on the cutting line or close to it, the connected component is separated into two parts. This case typically occurs if two characters from different text lines touch each other.
In Figs. 6 and 7 the first two lines extracted from Fig. 5 are shown. There are three instances of cut characters, all of which are correctly assigned to the first

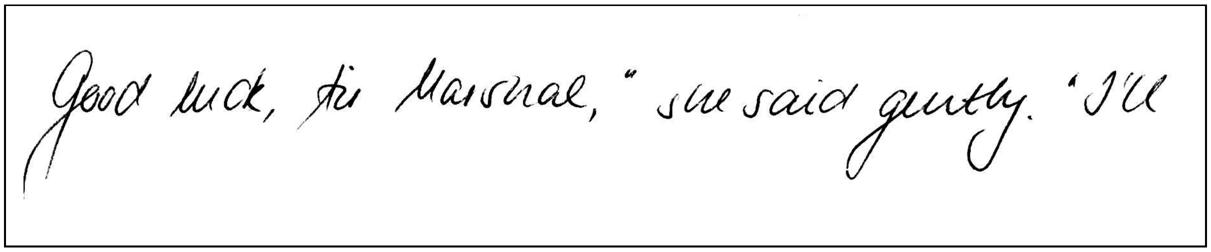

Good luck, Sir Marshal," she said gently. 'I'll

Figure 6: First line of Fig. 4 segmented.

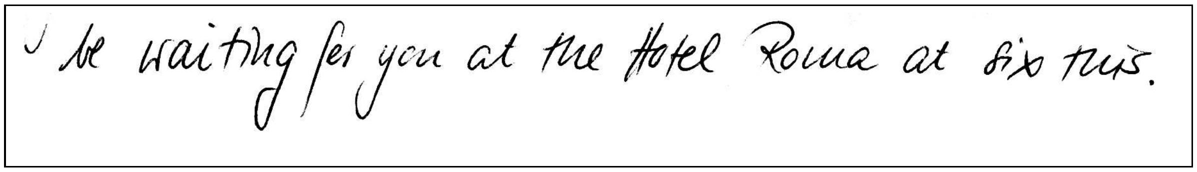

' ll be waiting for you at the Hotel Roma at six this.

Figure 7: Second line of Fig. 4 segmented.

text line. Notice, however, that due to low contrast during binarization, the first character on the first line is split into several connected components, the smaller of which become part of the second text line.
The work described in this paper is based on the image database reported in [24]. Almost all text images from this database could be correctly segmented into text lines. Only in 0.6% of the extracted text lines small errors like the ones in Figs. 6 and 7 occurred.

3.2 Skew Correction

After the text lines have been extracted, skew correction is performed. This operation aligns a line of text with the x-axis of the image and corrects any rotation that may have occurred during the writing or scanning process. Because of the use of rulers, the assumption is made that the lower baseline of the writing can be approximated by a straight line.
In order to determine the skew angle of a text line, we first estimate its lower base line. To do this estimation, the position of the lowest black pixel is computed for each column of the text line image. The set P of pixels obtained from this procedure approximate the lower contour of the handwritten text in the image. Formally, it can be represented as follows:

$$P = \{p_i = (x_i, y_i) | \text{lowest black pixel in column } x_i\} \tag{1}$$

If there are no black pixels in a column, no pixel p_i is added to P. Assume that set P has k entries. On this set of points a linear regression can be applied. The final goal of skew detection is to find the parameters a and b of a straight line expressed by the following equation:

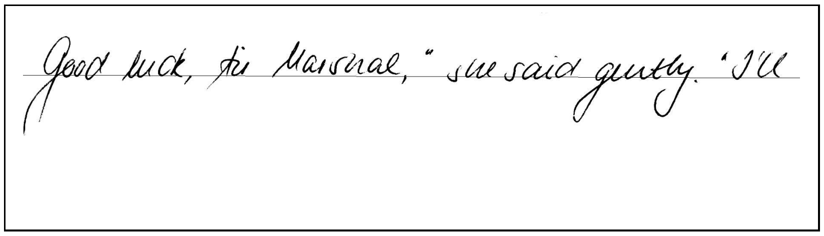


Figure 8: First line of Fig. 4 with lower baseline overlaid.

$$y = ax + b \tag{2}$$

For this purpose the mean values of the two variables x and y have to be computed:

$$\mu_x = \frac{1}{k}\sum_{i=1}^{k} x_i, \qquad \mu_y = \frac{1}{k}\sum_{i=1}^{k} y_i \tag{3}$$

Then, the line parameters a and b can be obtained using the following two formulas:

$$a = \frac{\sum_{i=1}^{k} x_i y_i - k\mu_x\mu_y}{\sum_{i=1}^{k} x_i^2 - k\mu_x^2}, \qquad b = \mu_y - a\mu_x \tag{4}$$

Linear regression minimizes the error between the line and the given set of points. A problem with this approximation is, however, that outliers in the set P may disturb the regression line. For the task of baseline estimation, the character's descenders can be regarded as outliers. To reduce their influence on the regression line, the summed square error between the line and the set P is computed by:

$$e = \sum_{i=1}^{k} (ax_i + b - y_i)^2 \tag{5}$$

If the total error e is larger than a predefined threshold t_e, the point p_i with the largest amount in the sum is eliminated from set P. This procedure is repeated until the error e is smaller than t_e.
With the algorithm described above not only the desired skew angle $\alpha_{skew} = atan(a)$ is determined, but also the lower baseline of the text line under consideration. Applying a rotation by an angle of $-\alpha_{skew}$, the skew is corrected. A similar method has been described in [12].
In Figs. 8 and 9 two examples of baselines computed by our method can be seen. (Note, however, that the skew in these lines is very small. Thus it is nearly invisible.)

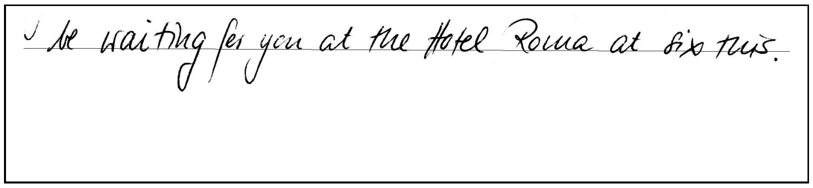


Figure 9: Second line of Fig. 4 with lower baseline overlaid.

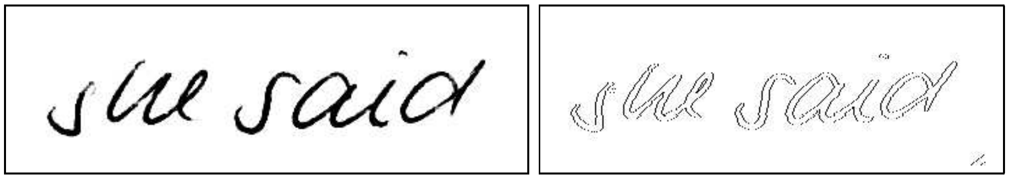


Figure 10: Image of two words: original (left) and vertical (right) contour.

3.3 Slant Correction

Once the skew of the image has been corrected, the slant of the writing can be normalized. This operation has the goal to bring the writing in an upright position. The method used in this work is similar to the one described in [25].
To correct the slant, the angle between the actual, quasi-vertical strokes and the y-axis is measured. For this purpose, the angle distribution of the writing's contour points is accumulated in an angle histogram. To reduce the influence of horizontal contour lines, which disturb the method, only nearly vertical parts are considered. Therefore a vertically oriented contour is computed. This is done by taking only the horizontal black-white and white-black transition pixels into account, ignoring horizontal contour points. Fig. 10 shows a text image and the corresponding vertical contour.
The vertical contour is approximated by a set of straight lines l_i. For this purpose, the contour is traced, until the error between the line l_i and the actual contour points is smaller than a predefined threshold. If the error is larger a new line is started. For each line l_i, the angle λ_i to the vertical axis and the length $|l_i|$ is determined. Based on the list of lines l_i, the angle histogram h_{slant} is computed in the following manner:

$$h_{slant}[\beta] = \sum_{i:\lambda_i=\beta} |l_i|^2, \qquad \beta \in [0, 2\pi] \tag{6}$$

Each entry is weighted by the square of the line's length. This has the effect that long lines like ascenders or descenders have a stronger influence on the slant angle.
The maximum value of histogram h_{slant} is the dominant quasi-vertical direction and gives the slant angle β_{slant}.

$$\beta_{slant} = \text{maxarg}_\beta(h_{slant}[\beta]) \tag{7}$$

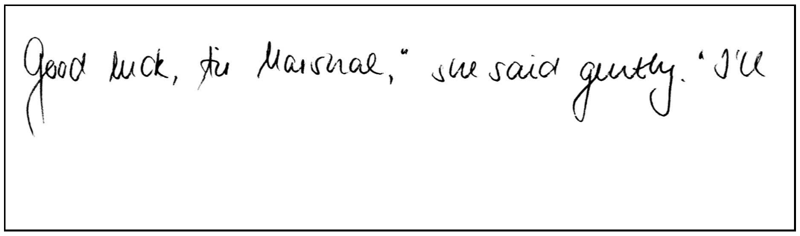

Good luck, Sir Marshal," she said gently. 'I'll

Figure 11: Fig. 6 after slant correction.

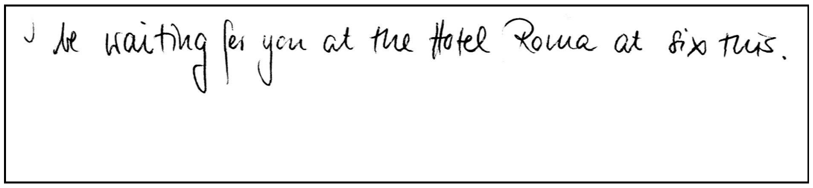

I be waiting for you at the Hotel Roma at six this.

Figure 12: Fig. 7 after slant correction.

Knowing β_{slant}, the writing's slant can be corrected by means of a shear operation. Figs. 11 and 12 show two corrected lines (see also Figs. 8 and 9).

3.4 Positioning of the Text Line

Positioning is a preprocessing operation based on vertical translation and scaling. It transforms the writing into a standard position and normalizes the height of the three main areas of a text line. Because the features used in the HMM are of geometric nature (see Sec. 4), this kind of normalization has a significant impact on the recognition performance.
First the upper and lower baselines, b_u and b_l, of the writing have to be determined. The lower baseline is already known from skew correction. For finding the upper baseline, the histogram of the horizontal projection of the image of the actual text line is used. The real histogram h_{real} is matched with an ideal histogram h_{ideal} (see Fig. 13). Seven points of interest have to be determined. Three of them, the upper bound u, the lower bound l and the maximum max are easy to find. The other four points, b_{u1}, b_{u2}, b_{l1} and b_{u2}, are computed through minimization of the total square of error between the real histogram h_{real} and the ideal histogram h_{ideal}. The splitting of each baseline b_u and b_l into two lines allows a better approximation of the real histogram. The minimization is done in the following manner:

$$e = \min_{b_{u1},b_{u2}} (h_{real} - h_{ideal}(u, b_{u1}, b_{u2}, max))^2 + \min_{b_{l1},b_{l2}} (h_{real} - h_{ideal}(max, b_{l1}, b_{l2}, l))^2 \quad (8)$$

By choosing the position of b_{u1} and b_{u2} the ideal histogram can be fitted to the real one. The minimum is found by exhaustively searching all possible candidates for b_{u1} and b_{u2}. The same procedure is applied for the points b_{l1} and b_{l2}.

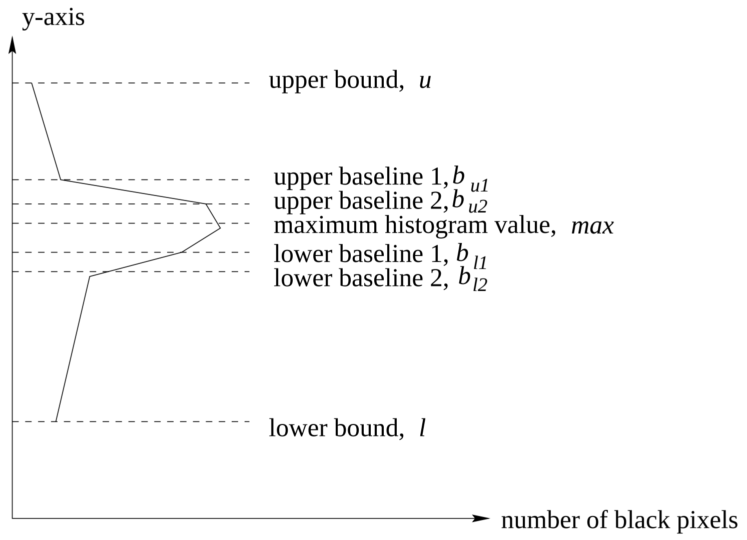

y-axis
upper bound, u
upper baseline 1, b_{u1}
upper baseline 2, b_{u2}
maximum histogram value, max
lower baseline 1, b_{l1}
lower baseline 2, b_{l2}
lower bound, l
number of black pixels

Figure 13: Model of the ideal histogram.

For the text line in Fig. 11, Fig. 14 shows the real histogram h_{real} fitted to the ideal histogram h_{ideal} with minimal square error.
From b_{u1}, b_{u2}, b_{l1} and b_{l2}, the upper and lower baseline is obtained as follows:

$$b_u = \frac{b_{u1} + b_{u2}}{2}, \qquad b_l = \frac{b_{l1} + b_{l2}}{2} \tag{9}$$

Fig. 15 shows the upper and lower baselines and the upper and lower bound of the text line of Fig. 11.
For the positioning it is important to know whether the writing has ascenders or descenders. For this purpose the height of the middle area is considered in relation to the upper and lower area. If a certain threshold is exceeded, it is assumed that there are no ascenders or descenders in the corresponding areas.
Once the baselines are known, the positioning and scaling of the image can be done. The two bounds and the two baselines define three distinguished areas. Each of these areas is scaled separately to a predefined height, which is identical for all three areas. Figs. 16 and 17 show the result of the positioning and vertical scaling.

3.5 Horizontal Scaling

For horizontal scaling, the black-white transitions in horizontal direction are used. For each image line, the number of transitions is determined. From the maximum number of transitions and the image width the mean number of strokes per pixel is estimated. This number can be set in relation to a reference value. The reference value is computed off-line by statistics over the whole available set of text lines in the image database. For example, in [26] it has been reported that there are

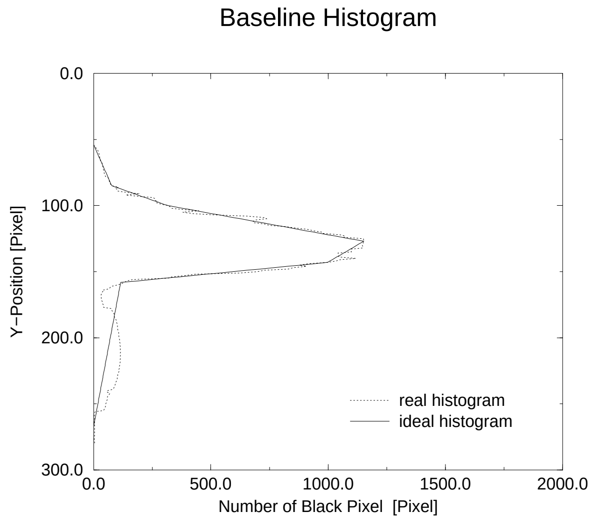


Figure 14: Real histogram fitted to the ideal histogram.

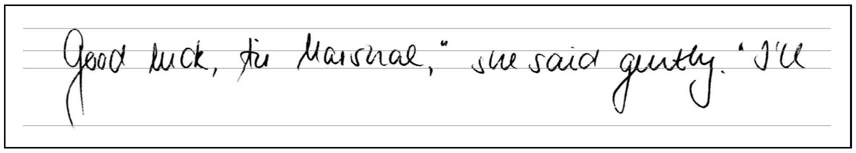


Figure 15: Upper and lower boundaries and baselines

100 pixel per character on the average. In our database [24], we found a value of 53 pixel per character (at 300 dpi resolution).
The relation between the actual and the reference value gives the scaling factor for the horizontal direction. Fig. 18 and Fig. 19 show examples of horizontally scaled images (bottom). For easier comparison the original images, i.e. Figs. 16 and 17, are displayed as well (top).

4 Feature Extraction

To extract a sequence of features from a text line, a sliding window is used. The window is of one column width and the image's height m, and is moved from left to right over each text line. (Thus there is no overlap between two consecutive window positions.) To determine the features at each window position, nine geometrical quantities are computed.
The first three features are the weight of the window (i.e. the number of black pixels), its center of gravity, and the second order moment of the window. This set characterizes the window from the global point of view. It includes information on how many pixels in which region of the window are, and how they are distributed. The following three formulas describe these features:

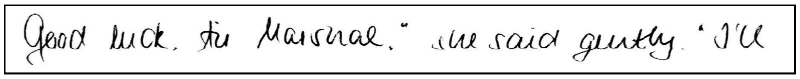

Good luck. Su Marshal." she said gently. 'I'll

Figure 16: Fig. 11 positioned and scaled in vertical direction.

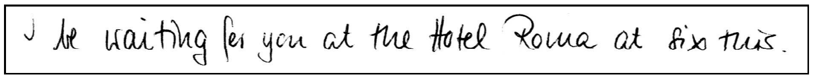

be waiting for you at the Hotel Roma at six this.

Figure 17: Fig. 12 positioned and scaled in vertical direction.

$$f_1(x) = \sum_{y=1}^{m} p(x,y) \tag{10}$$

$$f_2(x) = \frac{1}{m}\sum_{y=1}^{m} y \cdot p(x,y) \tag{11}$$

$$f_3(x) = \frac{1}{m^2}\sum_{y=1}^{m} y^2 \cdot p(x,y) \tag{12}$$

Features four to nine give more details about the writing. Features four and five define the position of the upper and the lower contour in the window. The next two features, number six and seven, give the orientation of the upper and the lower contour in the window by the gradient of the contour at the window's position. As feature number eight the number of black-white transitions in vertical direction is used. Finally, feature number nine gives the number of black pixels between the upper and lower contour.
Notice that all these features can be easily computed from the binary image of a text line. However, to make the features robust against different writing styles, careful preprocessing as described in Section 3 is necessary.

5 Hidden Markov Models

Hidden Markov models (HMMs) are widely used in the field of pattern recognition. Their original application was in speech recognition [22]. But because of the similarities between speech and cursive handwriting recognition, HMMs have become very popular in handwriting recognition as well [19].
In systems with a small vocabulary, it is possible to build an individual HMM for each word. But for large vocabularies this method doesn't work anymore because of the lack of training data. Therefore, in our system an HMM is build for each character. The use of character models allows to share training data. Each instance of a letter in the training set has an impact on the training and leads to a better parameter estimation.

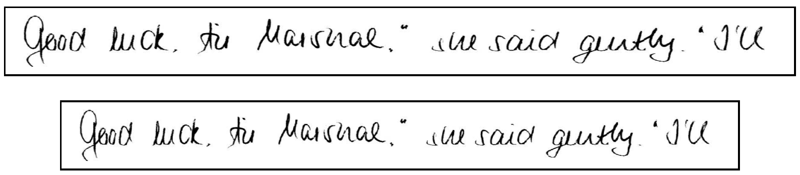

Good luck, Sir Marshal." she said gently. 'I'll
Good luck, Sir Marshal." she said gently. 'I'll

Figure 18: Fig. 16 scaled in horizontal direction.

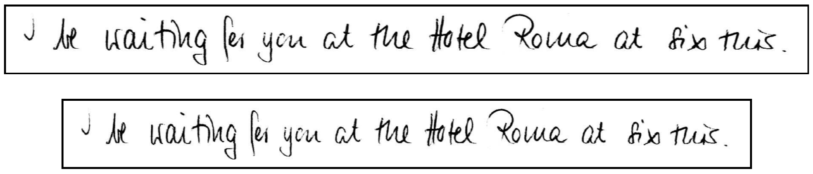

be waiting for you at the Hotel Roma at six this.
be waiting for you at the Hotel Roma at six this.

Figure 19: Fig. 17 scaled in horizontal direction.

To achieve high recognition rates, the character HMMs have to be fitted to the problem. In particular the number of states, the possible transitions and the type of the output probability distributions have to be chosen.
In our system each character model consists of 14 states. This number has been found empirically. The rather high number can be explained by the fact that the sliding window used for feature extraction is only one pixel wide. Because of the left to right direction of writing, a linear transition structure has been choosen for the character models. From each state only the same or the succeeding state can be reached. Because of the continuous nature of the features, the output probability distributions $b(o|s_i)$ are continuous (see Fig. 20). The initialization of the models is done by Viterbi alignment to segment the training observations and recomputing the parameters of the models. To adjust the transition probabilities $p(.)$ and the output probability distributions $b(.)$ during training, the Baum-Welch algorithm [22] is used.
In the recognition phase the character models are concatenated to words, and the words to sentences. Thus a recognition network is obtained (see Fig. 21). Note that this network doesn't include any contextual knowledge on the character level, i.e., the model of a character is independent of its left and right neighbors. In the network the best path is found with the Viterbi algorithm [22]. It corresponds to the desired recognition result, i.e., the best path represents the sequence of words with maximum probability, given the image of the input sentence.
The architecture shown in Fig. 21 makes it possible to avoid the difficult task of segmenting a line of text into individual words. In fact, the basic entities input to the HMM are the feature vectors from a complete line of text. Segmentation of the line into words is delivered by the HMM as a byproduct of the recognition process (see also Figs. 23 and 24). This is an important aspect in which the system described in this paper differs from others reported in the literature [17, 18].

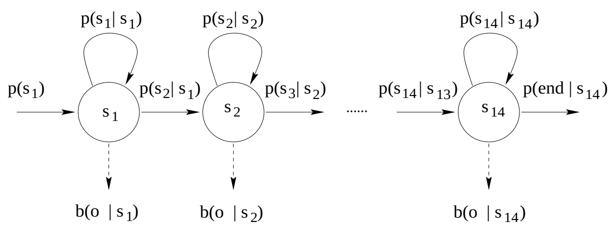


Figure 20: HMM for a single character.

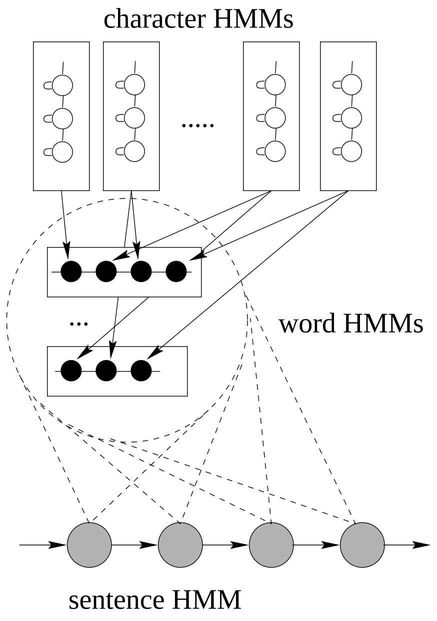


Figure 21: Recognition network.

6 Statistical Language Models

In natural language, the frequencies of words are not equally distributed. Neither are position and sequence of words random. Therefore, additional knowledge about the language can be introduced in the recognition of handwritten sentences. This knowledge constrains both the Viterbi search and the way the sentence HMMs are build from the word HMMs (see Fig. 21).
In this section we first give an introduction to the text collection which is used to estimate the language model parameters. Then a number of particular language models are described. Finally, we introduce and discuss the concept of perplexity which allows to measure the quality of the considered language models.

6.1 Corpus

To build a language model, we have to make assumptions about the underlying language, or we need a representative set of texts (corpus) from which we extract

the characteristic features of the model. For this work, it was decided to use a collection of texts from which the language models are automatically compiled. There are several corpora available. We used the Lancaster - Oslo/Bergen (LOB) corpus [27], the British English pendant to the Brown Corpus [28], which is based on American English. The texts in the corpus are classified into 15 different categories, such as journal papers, scientific texts, popular lore, fiction and so on. Totally there are 500 texts in the collection each containing about 2000 word instances. So the total number of word instances is approximately 1'000'000.
Let V be a vocabulary,

$$V = \{w_1, \ldots, w_n\} \tag{13}$$

where n denotes the vocabulary size and w_i the words. Then the following quantities can be easily derived from the corpus:

- N: the total number of word instances from the vocabulary V found in the corpus.
- $N(w_i)$: the total number of instances of word $w_i \in V$ in the corpus.
- $N(w_i, w_j)$: the total number of instances in the corpus where word $w_j \in V$ immediately follows word $w_i \in V$.

Note that these quantities are only computed for words belonging to the given vocabulary V. Thus unknown words in the corpus are ignored. The numbers N, $N(w_i)$, and $N(w_i, w_j)$ are the basis for computing the different parameters needed in the considered language models.

6.2 Simple Sentence Model

The first language model considered is the most simple one where any word of the vocabulary can follow any other word with the same probability. No further knowledge is introduced in this model. The words of the vocabulary are equally distributed. Each word has the same probability to occur at any position in the sentence. The probability depends only on the size n of the vocabulary, i.e. the probability of word $w_i \in V$ is

$$p(w_i) = \frac{1}{n} \tag{14}$$

Of course this language model doesn't contain any real "knowledge" about the language, but it is interesting as a reference model to get a lower bound on the recognition performance.

6.3 Unigram Sentence Model

In the second language model considered in this paper, the assumption that all words of the underlying vocabulary are equally distributed is replaced by the actual word occurrence distribution evaluated on the LOB corpus. This word distribution is surely more realistic than the simple sentence model. The resulting model is called unigram model in the following.
The probability of the words in the unigram language model can be computed as follows:

$$p(w_i) = \frac{N(w_i)}{N} \tag{15}$$

Because $\sum_{i=1}^{n} N(w_i) = N$ the probability condition $\sum_{i=1}^{n} p(w_i) = 1$ is fulfilled.

6.4 Bigram Sentence Model

The simple and the unigram sentence models have no memory to store the history of the words that were previously recognized. In human reading, however, this kind of memory is very important in recognizing sequences of words. In order to include information on how the occurrence of one word influences the words at the previous or the next position in a sentence, word pairs are considered. This leads to the conditional probability that word w_j occurs immediately after word w_i in the text:

$$p(w_j|w_i) = \frac{N(w_i, w_j)}{N(w_i)} \tag{16}$$

It can be easily seen that the probability condition $\sum_{j=1}^{n} p(w_j|w_i) = 1$ is fulfilled as $N(w_i) = \sum_j N(w_i, w_j)$. The resulting language model is called the 'bigram' sentence model.
In the context of this paper, no language models of higher order were considered. Theoretically it is straightforward to define such language models, using n-grams with $n \geq 3$. But their practical implementation becomes quite involved.

6.5 Back-off Sentence Model

To estimate the parameters of language models reliably, a large amount of data is necessary. However, for certain words it is not possible to get enough training data for reliable parameter estimation. Therefore, the language model has to be smoothed.
For the unigrams in the language model a threshold t_u is introduced, defining the minimum number of occurrences of a word. If the actual number of instances of a word w_i is less than t_u, we set it to t_u to compute the probability $p(w_i)$. Of course, the total number N of words under consideration has to be appropriately

adjusted then. On the other hand, if there are enough word occurrences the probability is computed in the same way as in the normal unigram model:

$$p(w_i) = \begin{cases} \frac{N(w_i)}{N} & \text{if } N(w_i) > t_u \\ \frac{t_u}{N} & \text{if } N(w_i) \leq t_u \end{cases} \tag{17}$$

For the bigram model the same technique can be applied. If there are not enough word pair instances for a reliable estimation, the bigram probability $p(w_j|w_i)$ is approximated by the unigram probability $p(w_j)$. To fulfill the condition that probabilities ad up to one, a scaling factor $b(w_i)$ is used for adjustment.

$$p(w_j|w_i) = \begin{cases} \frac{N(w_i,w_j)}{N(w_i)} & \text{if } N(w_i, w_j) > t_b \\ b(w_i)p(w_j) & \text{if } N(w_i, w_j) \leq t_b \end{cases} \tag{18}$$

$$b(w_i) = \frac{1 - \sum_{w_j \in B} p(w_j|w_i)}{1 - \sum_{w_i \in B} p(w_i)}, \qquad B = \{w_i | N(w_i, w_j) > t_b\} \tag{19}$$

The probabilities $p(w_i)$ and $p(w_j|w_i)$ computed under the different language models are used in the sentence HMM to weight the links between the words (see Fig. 21). Thus these probabilities rate the words during Viterbi decoding.

6.6 Perplexity

In the previous sections, different language models were introduced. The power of each of these models can be determined by means of experiments. But experiments of this kind are often time consuming and require large amounts of training and test data. An alternative way to assess a language model is to compute its perplexity. The advantage of this approach is that the perplexity can be directly computed from the language model.
Consider a sentence s that consists of l words from the dictionary, i.e., $s = w_1 \dots w_l$. The probability of this sentence

$$p(s) = p(w_1 \dots w_l) \tag{20}$$

can be computed in various ways. If we assume that we have complete knowledge of the language we could use it in the following manner:

$$p(s) = p(w_1)p(w_2|w_1) \dots p(w_l|w_1 \dots w_{l-1}) \tag{21}$$

But in general this knowledge is not available, as we only have a language model with a limited memory, or history. So the first possibility is to use the simple sentence model, where all probabilities are equal. For this model the probability defined in eq. (20) turns into:

$$p(s) = p(w_1)p(w_2) \dots p(w_l) = \prod_{i=1}^{l} p(w_i) = n^{-l} \tag{22}$$

As an alternative, we can use the unigram probabilities $p(w_i)$. The probability $p(s)$ for the given sentence s now becomes:

$$p(s) = p(w_1)p(w_2)\dots p(w_l) = \prod_{i=1}^{l} p(w_i) \tag{23}$$

Furthermore using our bigram sentence model, we obtain:

$$p(s) = p(w_1)p(w_2|w_1)\dots p(w_l|w_{l-1}) = p(w_1)\prod_{i=2}^{l} p(w_i|w_{i-1}) \tag{24}$$

Notice that for the back-off sentence model formulas (23) and (24) remain the same.
One disadvantage of the probability $p(s)$ computed according to eqs. (22) - (24) is that it depends on the length of the sentence s, i.e., the longer a sentence gets, the smaller is its probability. To overcome this dependency, we replace $p(s)$ by the perplexity $\mathbf{P}$, defined as follow (for details see [29]):

$$\mathbf{P} = p(s)^{-1/l} = p(w_1 \dots w_l)^{-1/l} \tag{25}$$

Alternatively to using $\mathbf{P}$, its logarithm $\mathbf{LP}$ can be used:

$$\mathbf{LP} = -\frac{1}{l}\log_2(p(s)), \qquad \mathbf{P} = 2^{\mathbf{LP}} \tag{26}$$

From the information theoretic point of view, any language can be seen as an information source. The amount of information from this source can be measured by means of the entropy $\mathbf{H}$:

$$\mathbf{H} = -\lim_{l\to\infty}\frac{1}{l}\log_2(p(s)) = \lim_{l\to\infty}\mathbf{LP} \tag{27}$$

So the perplexity $\mathbf{P} = 2^{\mathbf{H}}$ can be interpreted as the average number of possible successors of a word as sentence length approaches infinity.
From eq. (25) it follows that the perplexity of the language models discussed in the previous section can be computed by using the unigram and bigram probabilities. In the simple sentence model, the perplexity reaches its maximum for a given vocabulary V with n words:

$$\mathbf{P} = p(s)^{-1/l} = \Big(\prod_{i=1}^{l} 1/n\Big)^{-1/l} = \Big(n^{-l}\Big)^{-1/l} = n \tag{28}$$

On the other hand, the minimum perplexity is obtained if and only if the next word w_i is unique, i.e., $p(w_i|w_1 \dots w_{i-1}) = 1$ for a certain history $w_1 \dots w_{i-1}$. In this case the perplexity turns into:

$$\mathbf{P} = p(s)^{-1/l} = \Big(\prod_{i=1}^{l} 1\Big)^{-1/l} = 1^{-1/l} = 1 \tag{29}$$

For the unigram and bigram language model the values lie between these boundaries. For the unigrams the words are not equally distributed any more. Therefore the perplexity is smaller than n. In the bigram model, where more information is used, the perplexity is further decreased.
By computing the perplexity, the power of any chosen language model can be estimated independent of any recognition experiments. The smaller the perplexity is, the more knowledge, or constraints, are included in the model.

7 Experiments and Results

The system described in this paper was tested in a number of experiments. These experiments are described in Section 7.3. The underlying database is presented in Section 7.1. An analysis of the perplexity of the language models is given in Section 7.2.

7.1 Database

A database of handwriting consisting of sentences from the LOB-corpus was acquired. For this purpose, the texts in the corpus (see Sec. 6.1) were split into fragments of about 3 to 6 sentences with at least 50 words each. These text fragments were printed out on forms and different persons were asked to provide handwritten copies of the text on the forms.
As the main focus of this research is the application of high level linguistic knowledge in conjunction with HMMs, we wanted to make image preprocessing as easy as possible. Therefore, we decided that the writers had to use rulers. These guiding lines were printed on a separate sheet of paper which was put under the form. The writers were asked to use their every day writing in order to get the most natural and unconstrained way of writing. We also told the writers to stop to write, if there was not enough space left to write the whole text. This way we wanted to avoid to get pressed and deformed words. There were no constraints regarding the pencil. So all kinds of writing instruments are represented in the database.
The filled forms were scanned with a HP-Scanjet 6100 which is connected to a Sun Ultra 1. The software used to scan the data is *xvscan* version 1.6. It is an add on to the well known image tool *xv*. The resolution was set to 300 dpi at a grey level resolution of 8 bit. The images were saved in TIFF-format with LZW compression. Each form was completely scanned, including the printed and handwritten text. (Thus it is also possible to do experiments with the printed text, for example to distinguish between handwritten and printed text.)
At the moment (November 1999) the database consists of a total of 729 filled forms. There are 5851 lines of handwritten text all together. A total of 50754 word instances out of a vocabulary of 7719 words are included in the database.

	Language model				
Voc. size	Simple	Unigram	Bigram	Unigram normalized	Bigram normalized
2703	2703	365.62	74.72	0.135	0.027
3411	3411	398.66	78.25	0.117	0.022
4409	4409	449.72	82.30	0.102	0.018
7719	7719	564.60	88.22	0.073	0.011

Table 1: Perplexity evaluated on the LOB-corpus

There are between 8 and 9 words per line of text on the average. Ground truth for each form is available. A detailed description of the database can be found in [24].

7.2 Perplexity Analysis

In this subsection the language models are analyzed based on the perplexity introduced in Section 6.6. To compute the word occurrence and coocurrence probabilities $p(w_i)$ and $p(w_j|w_i)$ the LOB-corpus was used.
In Tab. 1 perplexities of the different language models are listed. They were computed for different vocabulary size. The first column in the table corresponds to the vocabulary size. The perplexity for the simple sentence model is given in the second column. Clearly, it has the same value as the vocabulary size as predicted by the theoretical analysis provided in Section 6.6. The next two columns give the perplexity for the unigram and bigram language model. For both, unigrams and bigrams, it can be seen that the perplexity increases with a growing size of the vocabulary. On the other hand, as it is expected, the perplexity is decreasing as we go from the simple to the unigram and bigram language model.
Because the perplexity of a language model depends on the size of the vocabulary, it is hard to judge it on an absolute scale. To reduce the influence of the vocabulary size, we divide the perplexity of the unigram and bigram language models by the simple sentence perplexity, which is the same as the vocabulary size used in the language model. Thus the normalized perplexity is obtained:

$$\mathbf{P}_{\text{norm}} = \frac{\mathbf{P}}{n} \tag{30}$$

Columns five and six of Tab. 1 give this normalized perplexity.
In Fig. 22 the numbers given in columns 5 and 6 of Tab. 1 are plotted on a logarithmic scale. It can be clearly seen that with larger vocabulary size the normalized perplexity decreases. Obviously, for a small vocabulary the number of possible successors of a word is larger in relative terms than for a vocabulary of large size. Furthermore, the normalized bigram perplexity is significantly smaller

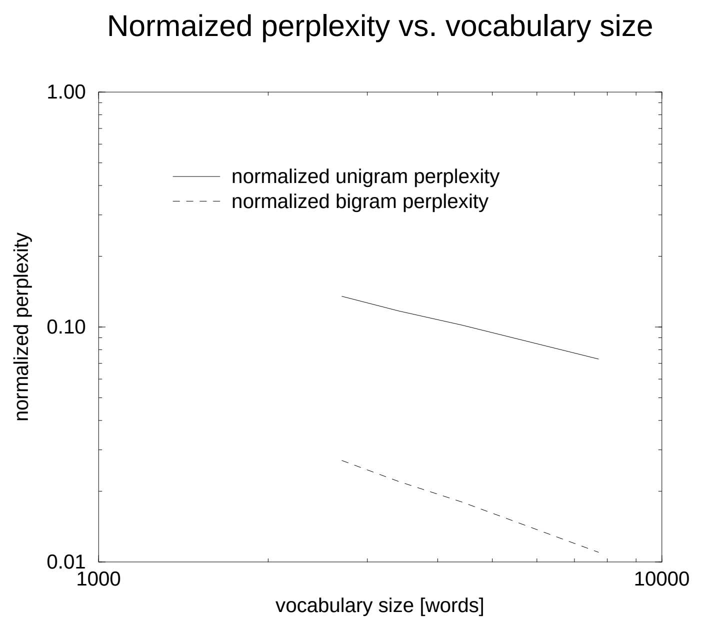


Figure 22: Perplexity graph of the normalized perplexity

than the normalized unigram perplexity. The results shown in Tab. 1 and Fig. 22 provide a theoretical foundation for the intuitive expectation that the recognition rate under the bigram model should be higher than under the unigram model, because the smaller number of successors of a word more severly constrains the search in the HMM.

7.3 Recognition Experiments

As described in Section 5 for each letter an HMM is build. These models are connected to words and sentences. In the training, the ground truth of each line is known. With this information, the Baum-Welch algorithm can be used to learn the parameters of the HMM. In each experiment, the image data set was divided into five parts of approximately equal size. Four of these sets were used for training and the fifth for testing. A cyclic permutation was performed, repeating each experiment five times.
For each of the vocabularies considered in the previous section, three recognition experiments were conducted. The first experiment was based on the simple sentence model. In the second and third experiment the unigram and bigram sentence model was used, respectively. The recognition rates reported in the following are on the word level.
The recognition results we received for different vocabulary sizes and different language models can be seen in Tab. 2. The number of different writers ranges from about 50 for the 2703 word vocabulary to about 300 for the 7719 word

System parameters		Recognition Rate		
Voc. size	Word instances	Simple [%]	Unigram [%]	Bigram [%]
2703	11000	51.44	52.15	61.18
3411	14806	50.72	51.71	63.39
4409	21462	47.58	48.63	61.68
7719	44019	40.47	42.13	60.05

Table 2: Recognition results under different vocabulary size and different language models

vocabulary.
As we go from top to bottom along the last three colums in Table 2, we observe that under each language model, the recognition rate drops with an increasing vocabulary size. This observation matches our intuitive expectation, and is confirmed by the perplexities reported in Table 1: The larger the vocabulary is, the less constraints exist on the possible successors of a word. Going from left to right along the rows and considering the last three columns, we see how the performance is enhanced by more powerful language models. Interestingly, this performance enhancement becomes more pronounced with larger vocabularies, such that the recognition rates for the 2703 and the 7719 word vocabulary are very similar under the bigram language model.

Recognition results for the first two lines of the text in Fig. 4 are shown in Figs. 23 and 24. In these two figures the segmentation of the text line into words, which is obtained as a byproduct of the recognition with the Viterbi algorithm, is displayed. The underlying model was the bigram language model and the size of the considered vocabulary was 7719 words.
The direct comparison of the performance of the considered system with others reported in the literature is not possible, because different databases for learning and testing are involved. Nevertheless, we want to cite a few other performance measures, without commenting on them in detail. In [18] a word recognition rate of 55.6% on a vocabulary of 1600 words was achieved. This system works with a word recognizer for isolated words (based on HMMs). In [30] 42.5% for a 525 word vocabulary and 37% for a vocabulary containing 966 words are reported. All values refer to the case where only the top choice of the system is taken into regard.

8 Conclusions

HMMs have been successfully applied to many handwriting recognition tasks. In this paper, we have considered the novel and difficult problem of recognizing

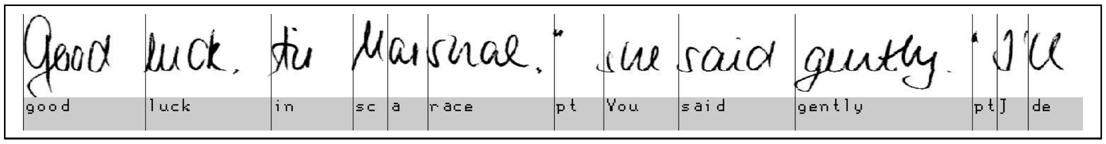


Figure 23: Recognition result with segmentation of Fig. 16

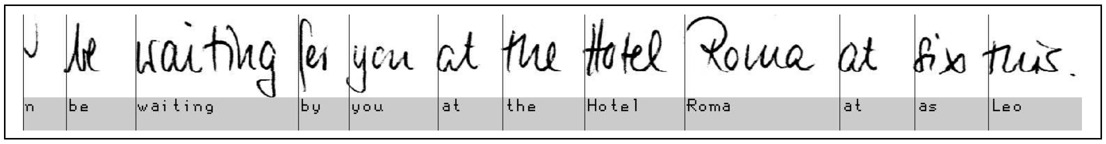


Figure 24: Recognition result with segmentation of Fig. 17

whole paragraphs and pages of unconstrained handwritten text. The HMM developed for this task has a hierarchical structure with character models at the lowest level. These models are concatenated to words and to whole sentences. Under such an architecture, the difficult problem of segmenting a line of text into individual words before recognition can be avoided. The HMM used in this work was furthermore enhanced by a language model incorporating linguistic information beyond the word level. Several language models of different complexity have been used. They were automatically generated from a large text corpus.
The language models were studied analytically using the perplexity as a measure of information contents. Besides, a number of experiments on a large database of handwritten texts were conducted. Generally speaking, very promising results were obtained in these experiments. It can be concluded that language models clearly enhance the recognition capabilities of HMMs. The more power a model possess, the higher is the improvement it implies.
Potential applications of a system like the one described in this paper are in the reading of historical archives or personal notes. Possibilities for further improving the system include the development of more general language models using, for example, n-gram statistics of order $n \geq 3$, larger corpora, or syntactical knowledge. Also a multiple classifier approach seems worth further investigation.

References

[1] J.-C. Simon. Off-line cursive word recognition. *Proc. of the IEEE*, 80(7):1150–1161, July 1992.

[2] I. Guyon, M. Schenkel, and J. Denker. Overview and synthesis of on-line cursive handwriting recognition techniques. In *[?]*, chapter 7, pages 227–258. World Scientific Publ. Co., 1997.

[3] R. Seiler, M. Schenkel, and F. Eggimann. Cursive handwriting: off-line versus on-line recognition. In A.C. Downton and S. Impedovo, editors, *Progress in Handwriting Recognition*, pages 29–36. World Scientific Publ. Co., 1997.

[4] C.Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. *Proc. of the IEEE*, 80(7):1162–1180, 1992.

[5] R. Bozinovic and S.N. Srihari. Off-line cursive script word recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(1):68–83, January 1989.

[6] A. Kaltenmeier, T. Caesar, J.M. Gloger, and E. Mandler. Sophisticated topology of hidden Markov models for cursive script recognition. In *Proc. of the 2nd Int. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan*, pages 139–142, 1993.

[7] 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.

[8] G. Dzuba, A. Filatov, D. Gershuny, and I. Kil. Handwritten word recognition - the approach proved by practice. In S.-W. Lee, editor, *Advances in Handwriting Recognition*, pages 153–162. World Scientific Publ. Co., 1999.

[9] A.C. Downton and R.W.S. Tregidgo. The use of a trie structure dictionary as a contextual aid to recognition of handwritten British postal addresses. In *Proc. of the 1st Int. Conf. on Document Analysis and Recognition, Saint-Malo, France*, volume 2, pages 542–550, 1991.

[10] S.N. Srihari, V. Govindaraju, and A. Shekhawat. Interpretation of handwritten addresses in US mailstream. In *Proc. of the 2nd Int. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan*, pages 291–304, 1993.

[11] A. El Yacoubi, J.-M. Bertille, and M. Gilloux. Conjoined location and recognition of street names within a postal address delivery line. In *Proc. of the 3rd Int. Conf. on Document Analysis and Recognition, Montréal, Canada*, volume 2, pages 1024–1027, 1995.

[12] M. Schüssler and H. Niemann. A HMM-based system for recognition of handwritten adress words. In *[?]*, pages 213–225. World Scientific Publ. Co., 1999.

[13] S. Impedovo, P.S.P. Wang, and H. Bunke, editors. *Automatic Bankcheck Processing.* World Scientific Publ. Co., 1997.

[14] N. Gorski, V. Anisimov, E. Augustin, D. Price, and J.-C. Simon. A2ia check reader: A family of bank check recognition systems. In *5th Int. Conference on Document Analysis and Recognition 99, Bangalore, India*, pages 523–526, 1999.

[15] A. Agarwal, A. Gupta, K. Hussein, and P.S.P. Wang. Bank check analysis and recognition by computers. In H. Bunke and P.S.P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, chapter 24, pages 623–652. World Scientific Publ. Co., 1997.

[16] G. Kaufmann and H. Bunke. A system for the automated reading of check amounts - some key ideas. In *Proc. 3rd IAPR Workshop on Document Analysis Systems, Nagano, Japan*, pages 302–315, 1998.

[17] J.T. Favata, S.N. Srihari, and V. Govindaraju. Off-line handwritten sentence recognition. In *[?]*, pages 393–398. World Scientific Publ. Co., 1997.

[18] G. Kim, V. Govindaraju, and S.N. Srihari. Architecture for handwritten text recognition systems. In *[?]*, pages 163–172. World Scientific Publ. Co., 1999.

[19] A. Kundu. Handwritten word recognition using hidden Markov model. In *[?]*, chapter 6, pages 157–182. World Scientific Publ. Co., 1997.

[20] H.J. Kim, S.K. Kim, K.H. Kim, and J.K. Lee. An HMM-based character recognition network using level building. *Pattern Recognition*, 30(3):491–502, March 1997.

[21] C. Farouz, M. Gilloux, and J.-M. Bertille. Handwritten word recognition with contextual hidden markov models. In S.-W. Lee, editor, *Advances in Handwriting Recognition*, pages 183–192. World Scientific Publ. Co., 1999.

[22] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[23] S. Young and G. Bloothooft, editors. *Corpus-Based Methods in Language and Speech Processing.* Kluwer Academic Publishers, 1997.

[24] U.-V. Marti and H. Bunke. A full English sentence database for off-line handwriting recognition. In *Proc. of the 5th Int. Conf. on Document Analysis and Recognition, Bangalore, India*, pages 705–708, 1999.

[25] T. Caesar, J. M. Gloger, and E. Mandler. Preprocessing and feature extraction for a handwriting recognition system. In *Proc. of the 2nd Int. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan*, pages 408–411, 1993.

[26] M.K. Brown and S. Ganapathy. Preprocessing techniques for cursive script recognition. *Pattern Recognition*, 16(5):447–458, 1983.

[27] S. Johansson, G.N. Leech, and H. Goodluck. *Manual of Information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital Computers.* Department of English, University of Oslo, Oslo, 1978.

[28] W.N. Francis. *Manual of Information to Accompany a Standard Sample of Present-Day Edited American English for Use with Digital Computers.* Providence, Rhode Island: Department of Linguistics, Brown University, 1964.

[29] F. Jelinek. Self-organized language modeling for speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufmann Publishers, Inc., 1990.

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