Offline Handwriting Recognition and Grammar based Syntax Analysis

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Matthias Zimmermann

von Wattenwil, BE

Leiter der Arbeit: Prof. Dr. H. Bunke
Institut für Informatik
und angewandte Mathematik,
Universität Bern

Offline Handwriting Recognition and Grammar based Syntax Analysis

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Matthias Zimmermann

von Wattenwil, BE

Leiter der Arbeit: Prof. Dr. H. Bunke
Institut für Informatik
und angewandte Mathematik,
Universität Bern

Von der Philosophisch-naturwissenschaftlichen Fakultät
angenommen.

Bern, den 13.11.2003 Der Dekan:

Prof. Dr. G. Jäger

Abstract

This thesis investigates the combination of a syntax analysis module with an offline recognition system for cursive handwritten English sentences.

According to the proposed combination scheme the n-best sentence candidates produced by a handwriting recognition system are reordered in the syntax analysis module according to their syntactical soundness. As a measure of the syntactical quality, the probability of the most probable parse is used. This probability is computed by a parser in the syntax analysis module using a stochastic context-free grammar for general English sentences. The validity of this combination scheme has been verified by extensive experiments in both multi-writer and writer independent environments.

Based on the results of these experiments the conclusion can be drawn that stochastic context-free grammars may help to significantly increase the performance of a recognition system for handwritten English sentences.

Acknowledgments

The following words are dedicated to the people which directly and indirectly contributed to my thesis.

First, my thanks are due to Prof. Horst Bunke as my supervisor of this thesis.

For spending his time during many fruitful discussions and providing a very efficient implementation of a probabilistic parser I am very grateful to Dr. Jean-Cédric Chapelier of EPFL.

Furthermore, many colleagues and friends helped with discussions, coding, coffee breaks as well as proof reading. Roman Bertolami, Roger Brooks, Stefan Fischer, Simon Günter, Ergina Kavallieratou, Urs-Viktor Marti, Tobias Roth, Christoph Röthlisberger, Karin Sobottka, Tamas Varga, Alessandro Vinciarelli, Tabea Zimmermann and Marcel Zumbühl, thank you very much.

Finally, I thank my wife Daniela for her love, support and encouragement during my time as a PhD student at the University of Bern.

Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Statement and Aim 4
1.3 Contribution of this Thesis 5
1.4 Outline of the Thesis 7

I Resources **9**

2 Handwritten Image Data **11**
2.1 The IAM Database 12
2.2 Automatic Word Segmentation 15
2.3 Extraction of Sentence Images 22

3 Linguistic Resources **25**
3.1 The LOB Corpus 25
3.2 The Tagged LOB Corpus 26
3.3 The Lancaster Parsed Corpus 27
3.4 Unification of Linguistic Resources 28

II Offline Handwritten Text Recognition **31**

4 Introduction **33**
4.1 Challenges 33
4.2 State of the Art 35
4.3 System Overview 38

5 Methodology **43**
5.1 Preprocessing 43
5.2 Image Normalization 44
5.3 Extraction of Feature Vector Sequences 50
5.4 Hidden Markov Modeling 51
5.5 Character Specific Model Length Optimization . . . 57
5.6 Bigram Language Model 61
5.7 Grammar Scale Factor and Word Insertion Penalty 63
5.8 Recognition Lattices and N-Best Lists 64
5.9 Measuring System Performance 67

6 Experiments and Results **71**
6.1 Experimental Setup 71
6.2 Optimizing the Character Model Lengths 73
6.3 Optimizing the Number of Gaussians 77
6.4 Optimizing the Number of Training Iterations . . . 78
6.5 Optimizing Grammar Scale Factor and Word Insertion Penalty 80
6.6 Validation Set Results 83
6.7 Test Set Results 84

7 Conclusions **87**
7.1 Data Preparation and Preprocessing 87
7.2 Character Modeling 88
7.3 Decoding 88

III N-Best Sentence Candidate Parsing **91**

8 Introduction **93**
8.1 Parsing Natural Languages 94
8.2 Stochastic Context-Free Grammars 96
8.3 State of the Art 100
8.4 System Overview 105

9 Methodology **107**
9.1 Notations and Definitions 107
9.2 Construction of the SCFG 109
9.3 Bottom-up Chart Parsing 110
9.4 Combination Scheme 114

10 Experiments and Results **117**
10.1 Experimental Setup 117
10.2 The Quality of the SCFG 118
10.3 Optimizing the Parse Scale Factor 119
10.4 Validation Set Results 121
10.5 Test Set Results 123

11 Conclusions **129**
11.1 State of the Art 130
11.2 The Stochastic Context-Free Grammar 130
11.3 The Combination Scheme 131

IV Language Model Smoothing **133**

12 Introduction **135**
12.1 State of the Art 135

13 Methodology **139**
13.1 Statistical Language Modeling 139
13.2 Perplexity of Statistical Language Models 140
13.3 N-Gram Language Models 141
13.4 Generating Random Sentences from a SCFG 144

14 Experiments and Results **145**
14.1 Resources 145
14.2 Experimental Setup 146
14.3 Optimizing Perplexity 147
14.4 Optimizing System Performance 148
14.5 Test Set Results 151

15 Conclusions **155**

15.1 Language Model Smoothing with SCFG 155

15.2 Trigram Language Models 156

V Conclusions and Outlook **157**

16 Conclusions **159**

17 Outlook **161**

Bibliography **165**

Introduction 1

1.1 Motivation

1.1.1 History

The recognition of handwriting by machines has been a research topic for over 40 years [121]. While high recognition rates in the case of recognition of isolated numerals [86, 110] are standard today, system performance decreases significantly for more complex cases like numeral string recognition [49, 74] or the recognition of isolated handwritten words [11, 44]. Only in the last decade have researchers begun to investigate the recognition of general handwritten text [10, 71, 120].

1.1.2 Offline versus Online Recognition

The field of handwriting recognition is divided into offline and online recognition. Offline handwriting recognition can be seen as the more general case since there are fewer constraints on the writing instrument, the writing pad (e.g. paper) and the place and time of the capturing of the handwritten material. Online recognition is based on the recoding of the movement of a special pen during the writing process. This technique has two advantages over the

offline handwriting recognition. First, online data contain temporal information representing the writing process. Such information is not present in the case of offline data, where only the result of the writing process (the image) is available to a recognition system. Second, online data directly represents a time dependent physical signal. Therefore, well known techniques from the domain of signal processing and speech recognition are better suited for recognition of online data.

1.1.3 Applications of Offline Handwriting Recognition

The main industrial applications for the recognition of offline handwritten text are currently found in the area of address reading [125] and bank check processing [56], where recognition is based on digitized images of physical envelopes or bank cheques.

Applications for offline recognition of unconstrained handwritten text may be found in the future to the automatic recognition of personal notes and communications. Recognition rates of current systems in the offline, writer independent case are still far from being perfect.

The automatic transcription of large handwritten archives seems a more realistic scenario today. Such applications could support automatic indexing for information retrieval systems used in digital libraries which do not require perfect recognition rates.

1.1.4 Integration of Task Specific Knowledge

The recognition of addresses and bank checks are very demanding tasks since they require the offline recognition of a huge variety of writing styles and writing instruments. The system performance for such applications benefits substantially from the very restricted domains which allow the use of task specific knowledge. In the case of bank check recognition the equivalence of the legal and the courtesy amount is required [68]. Address recognition systems are restricted to existing zip codes, city names, and street names [148]. Furthermore, specific databases can narrow down the search space

to valid relations between zip codes and city names and between street names and cities.

For the recognition of general text, task specific information can only be found in the linguistic domain. So far, the successful application of statistical language models has been reported. Most often, so called n-gram models are used. N-gram models are based on the observed frequency of neighboring words that are trained using large amounts of training text [95, 143]. Although n-grams represent a very crude approximation of natural language they can significantly help to improve the performance of handwritten text recognition systems.

1.1.5 Human Performance in Recognizing Handwriting Texts

Given the handwritten text shown in Fig. 1.1 most people familiar with English will reach character and word recognition rates of 100% or come at least close to it.

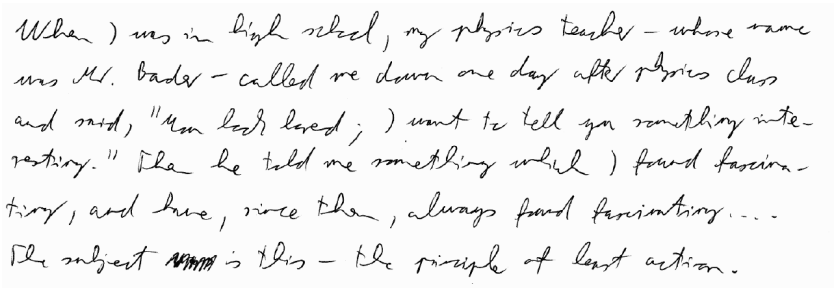

Figure 1.1
A handwritten text in English.

The quality of the transcriptions produced by humans is impressive. If asked to transcribe the handwritten text provided in Fig. 1.2 it is likely (assuming the reader is not familiar with Magyar) that the character recognition rate will drop to 70% and the word recognition rate will end up at about 30%.

This demonstrates that the quality of the transcriptions produced by humans depends heavily on the fact that we are capable of understanding what we read. The conclusion of this experiment is that human performance will be often poor if a letter by letter recognition is required. The knowledge of the possible vocabulary and frequently used word sequences (word n-grams) can already help

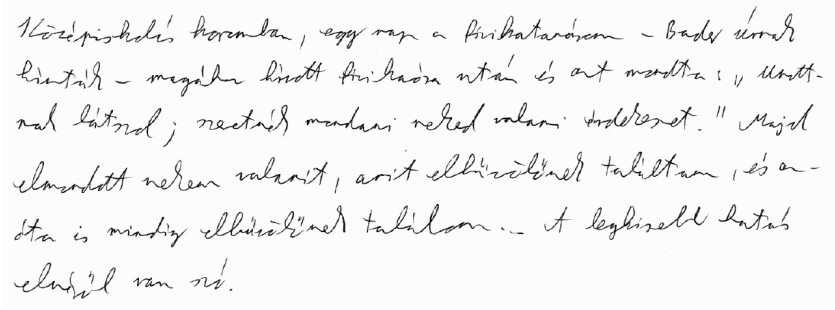

Figure 1.2
The Magyar version of the same text written by the same person.

to improve the word recognition rate. However, optimal transcriptions of handwritten texts can be expected if the reader is familiar with both the language and the topic of the text, i.e. if the text is actually understood.

It can generally be observed that the amount of task specific knowledge has a significant impact on the resulting 'system' performance.

1.1.6 Syntax Analysis

Syntax analysis can be seen as the next logical step towards text comprehension and represents an additional source of task specific knowledge compared to the n-gram modeling of natural language.

The syntax analysis investigated in this thesis is based on stochastic grammars for natural languages. Such grammars are used to find (grammatical) explanations for given input sentences and are further useful to assign probabilities to such explanations. The latter capability of stochastic grammars is of special interest since it allows the determination of the (grammatically) best alternative among a set of different candidate sentences.

1.2 Problem Statement and Aim

In this thesis the problem of writer independent offline recognition of general handwritten English text is addressed. The few works which have been published in this area have used n-gram language models to improve the recognition performance.

The aim of this thesis is to investigate the capability of large stochastic context-free grammars to improve a state of the art system for handwritten text recognition.

So far, only simpler forms of syntax analysis have been considered in the literature [25, 51, 53, 73] while stochastic context-free grammars have not yet been investigated in the field of handwriting recognition.

1.3 Contribution of this Thesis

The main contribution of this thesis to the field of offline handwriting recognition can be seen in its extensive experimental work. A more detailed list of the various contributions is provided in the following subsections.

1.3.1 IAM Database

During this thesis the IAM database has been enlarged and its usability has significantly increased. The database now represents the largest collection of offline handwritten English texts which is publicly available today.

An automatic word segmentation scheme has been developed and applied to the whole database [153]. It now contains the necessary material to support research in the fields of single-writer, multi-writer and writer independent recognition of isolated words, lines of handwritten text and complete sentences.

1.3.2 Baseline Sentence Recognizer

A new character specific model optimization technique has been introduced [154] and for the first time in handwriting recognition research the incorporation of the statistical language model has been systematically optimized.

1.3.3 Use of a Stochastic Context-Free Grammar

For a sequential coupling of a handwriting recognition system and a probabilistic parser a new combination scheme for recognition

scores and parse probabilities has been proposed [155]. Using this combination scheme, the first successful use of a large stochastic context-free grammar for general English texts in the context of a handwriting sentence recognition system has been achieved and confirmed by large scale experiments.

In additional experiments, the stochastic context-free grammar has been used to smooth bigram and trigram language models. This technique has been reported to significantly increase the recognition performance of speech recognition systems. However, no similar positive effect has been observed for the case of a large stochastic context-free grammar for general texts.

1.3.4 Trigram Language Models

So far, only the successful use of bigram language models has been reported for the recognition of offline handwritten texts [95, 143]. In contrast to the findings published in [143], trigram language models have clearly outperformed the bigram language model in the experimental setup used in this thesis.

1.4 Outline of the Thesis

Part I

The first part of this thesis presents the different sources of handwritten image data and linguistic resources which were used for the experiments described in the remaining parts.

Part II

The construction and optimization of an offline recognition system for handwritten sentences is described. The performance of this system serves as baseline for the experiments carried out in the next two parts of the thesis

Part III

A combination scheme for a baseline recognizer and a syntax analysis module is proposed. A probabilistic parser is used to compute the probabilities of the most likely parses for the n-best list of sentences candidates provided by the recognizer.

Part IV

Different strategies are investigated to improve bigram and trigram language models which are used to improve the performance of the baseline recognizer.

Part V

A summary of the main conclusions and an outlook on future research is provided.

I Resources

Handwritten Image Data

2

Large databases play a significant role in the development, evaluation, comparison and improvement of handwriting recognition systems. Until recently the majority of publicly available databases for offline handwriting recognition provided character and word images for specific domains only. Databases have been built for the recognition of handwritten legal amounts [45, 67] of which the CENPARMI database [132] is widely known. For the recognition of handwritten addresses [42, 90] the CEDAR database [52] is frequently used.

Most available databases containing English texts have limitations which restrict their usability for the problem of writer independent general text recognition. The handwritten texts provided in the NIST database [37] cover only a very small vocabulary based on the first few sentences of the American constitution. In the database described in [120] material from only a single writer is represented. A larger database containing essays written by students has been recently presented in [29].

The next section presents the IAM database, which contains the handwritten material used throughout all experiments reported in this thesis. Sec. 2.2 explains the automatic word segmentation scheme and Sec. 2.3 describes the extraction of the handwritten sentences.

2.1 The IAM Database

The IAM Database [96] has been built at our institute over the last few years to build, train and test offline handwriting recognition systems for unconstrained English texts [93, 94, 95]. The database has since been enlarged and its purpose was significantly enhanced to support additional tasks [153]. I.e. the recognition of isolated handwritten words and writer identification.

In its format reported in [96] the database contained over 1,000 pages of handwritten text including the ASCII transcription for each form image. The handwritten material has been provided by approximately 400 different writers. Unfortunately no information has been collected about personal information like sex, age, cultural and educational background of the contributing writers. Nevertheless it can be stated that a large variety of different text categories, writing styles, and writing instruments are covered by the IAM database. Achieving high recognition rates with the IAM database poses a considerable challange.

An example of a handwritten form is given in Fig. 2.1. At the upper part of each form a few sentences based on texts from the Lancaster-Oslo/Bergen corpus[1] have been provided. Each contributing person was then asked to copy the text in her/his everyday style of writing with a free choice of the writing instrument. The optional name field placed at the bottom of each form has only been filled in by some of the writers.

2.1.1 Text Line Extraction

Using the methods described in [96] the handwritten part of the form images can be extracted and then be segmented into its individual text lines.

The extraction of the handwritten part is simplified by the fact that two long horizontal lines separate the handwriting region from the text printed in the upper part of the form. The printed lines can easily be detected by horizontal projections and help to estimate the skew of the scanned form image. Three text lines from the extracted handwritten part are shown in Fig. 2.2

[1] See Sec. 3.1 for a brief summary of the LOB corpus.

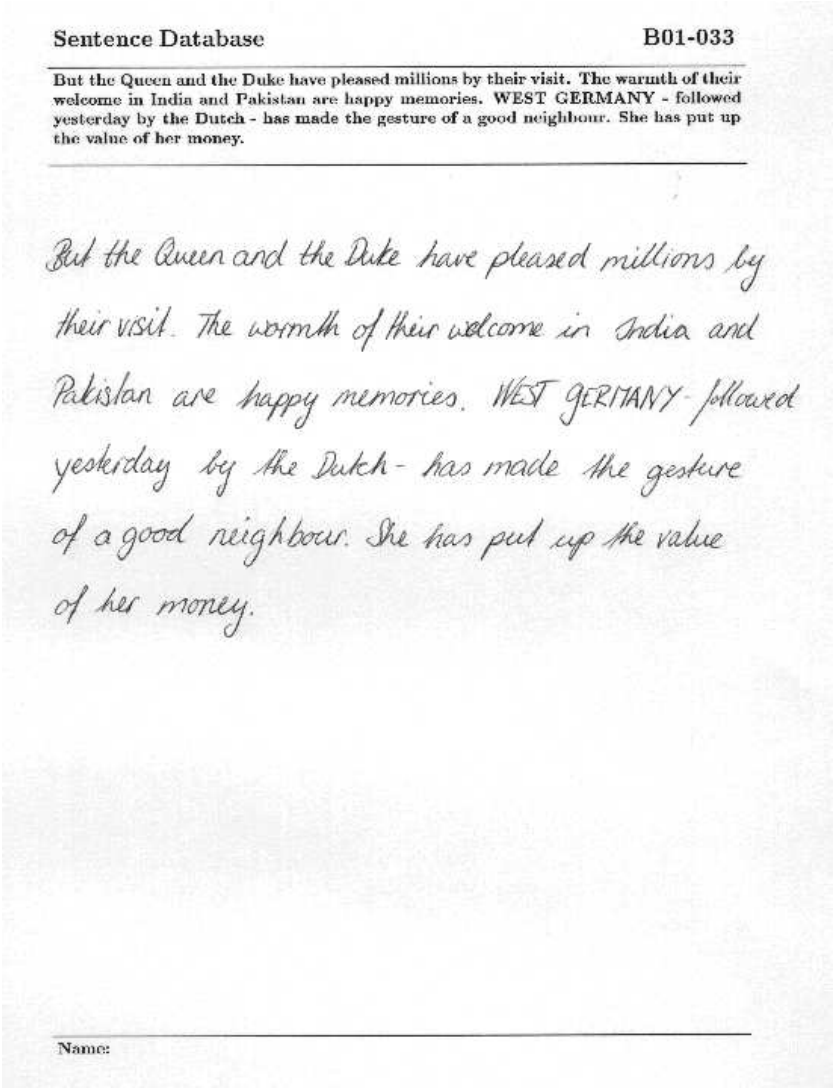

Figure 2.1
Form B01-033 from the IAM database.

The segmentation of the handwritten part into its text lines is based on a histogram of the horizontal black/white transition. Local minima in this histogram indicate candidates for segmentation points.

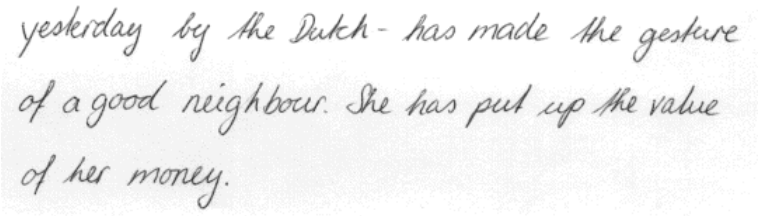

Figure 2.2
Three lines extracted from the handwritten part of the form image.

In the ideal case two consecutive lines of text are separated by one or more horizontal runs of background pixels. The separation of two consecutive handwritten lines is trivial if such lines do not contain any black/white transitions at all. The non-trivial cases may include a vertical overlap of handwritten text lines or even touching characters. See [96] for the description of the method used in such cases. Figure 2.3 illustrates the result of the text line extraction.

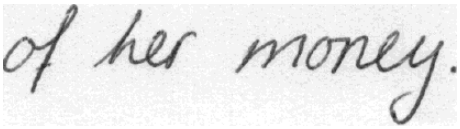

Figure 2.3
A resulting line image from text line extraction.

2.1.2 Current State

In its current state the IAM database contains over 1,500 forms of handwritten text contributed by over 600 different writers. Additional information about the current state is provided in Tab. 2.1. Correctly segmented words can be extracted automatically and are thus available for isolated word recognition tasks. The covered vocabulary counts the number of words for which at least a single handwritten instance can be found in the database.

Content of the IAM database	
Scanned form images	1,539
Contributing writers	657
Handwritten sentences	5,685
Handwritten text lines	13,353
Handwritten words	115,515
Correctly segmented words	96,591
Covered vocabulary	13,547

Table 2.1
Current state of the IAM database.

A large amount of additional information for each form has been produced and added to the form image file in the form of metadata where PNG[2] has been chosen as the image format and XML[3] has been used to store the metadata.

[2]Portable Network Graphics [114]. The format is free from patents and currently supported by most Web browsers. Libraries exist for a large variety of platforms and environments.

[3]Extensible Markup Language [24]. XML is a simple text based format which has become a standard to exchange structured information on the Web and elsewhere.

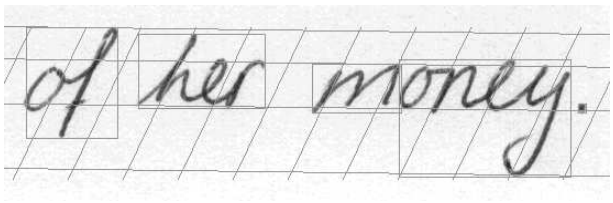

Figure 2.4
Line level metadata including estimated slant and reference lines and connected components.

The metadata includes an identification of the contributing writer and the bounding box of the handwritten text region for each form image in the database. Further information is provided on the text line level. It includes the ASCII transcription, its bounding box, the list of connected components as well as the estimated slant angle and the main writing regions. See Section 5.2 for a description techniques involved to estimate these parameters. Fig. 2.4 provides a graphical representation of the text line level metadata.

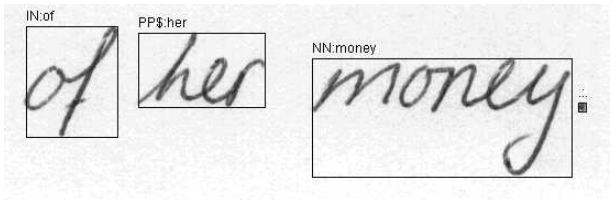

Figure 2.5
Word level metadata including word transcriptions, grammatical tags and bounding boxes.

Finally the word label and its grammatical category as defined in the Tagged LOB corpus[4] is provided for each handwritten word. Word bounding boxes and the corresponding lists of connected components are also available. See Fig. 2.5 for an illustration of the included word level metadata.

2.2 Automatic Word Segmentation

This section presents the automatic word segmentation scheme [153] which has been used to produce accurate word bounding box information for the IAM database. See Fig. 2.5 for the result of the word segmentation scheme presented in this section.

[4]See Sec. 3.1 for a brief summary of the Tagged LOB corpus.

The segmentation scheme consist of two steps. First, a Hidden Markov Model (HMM) based word segmentation step is applied on the normalized text line images. As a result, a list of word boundaries is produced for the normalized text lines. Second, connected components assigned to word boundaries found in the first step are used to determine the word bounding boxes in the original (non-normalized) text lines.

Please note that a number of techniques used for handwriting recognition will not be explained in the following subsections. However, text line normalization is described in Sec. 5.2 of this thesis, a description of the applied extraction of the feature vector sequences is provided in Sec. 5.3, and Sec. 5.4 contains a brief overview of the fundamentals of the HMM framework.

2.2.1 HMM Based Word Segmentation

The first step of the word segmentation scheme relies on the HMM based forced alignment technique. In contrast to recognition of handwritten text no attempt to recognize text lines is undertaken. Instead, the transcription of a complete line is provided to the Viterbi decoder used in forced alignment mode. This produces an optimal segmentation of the text line into its words[5]. To achieve very good segmentation results well trained character models are required. In order to obtain reliable character HMMs all text line images have to be normalized to reduce the variability found in the different writing styles.

Fig. 2.6 provides an example of the text line normalization and the extraction of the corresponding feature vector sequence $X = (X_1, X_2, \ldots X_i \ldots X_n)$. Using a sliding window technique a single feature vector X_i is extracted for each column of the normalized text line image. Consequently horizontal coordinates of image columns are directly represented by the indices of the corresponding feature vectors. This fact will allow the accurate location of the word beginnings.

The HMM state sequence $Q = (q_1, q_2, \ldots q_j \ldots q_m)$ used in the forced alignment mode can be generated by the concatenation of

[5]The segmentation produced by the Viterbi decoder is optimal for a given HMM state sequence and a feature vector sequence extracted from the text line image.

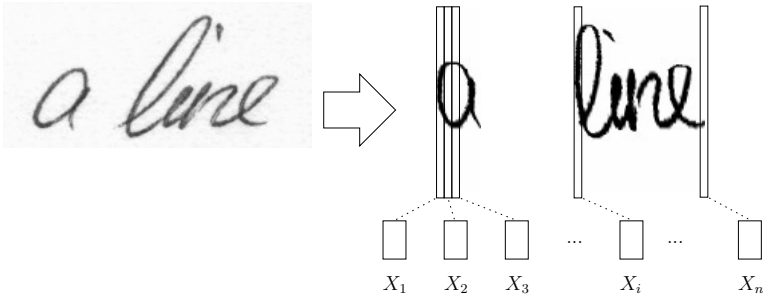

Figure 2.6
Text line normalization and extraction of a feature vector sequence.

the trained character HMMs according to the transcription of the text line. This is shown in Fig. 2.7 where the character HMMs have been simplified by using fewer states and omitting the state transitions inside the models. All character HMMs in our system have a linear left to right topology which only allows transitions from a given state to itself or to its right neighbor.

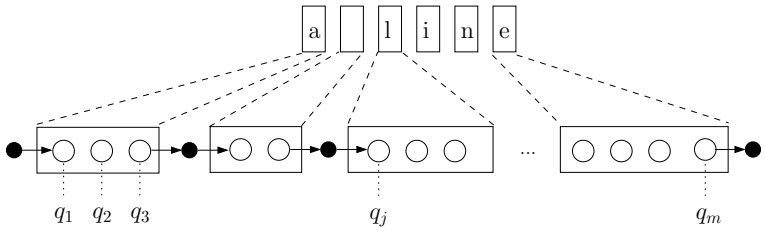

Figure 2.7
Generation of a state sequence by concatenation of character HMMs.

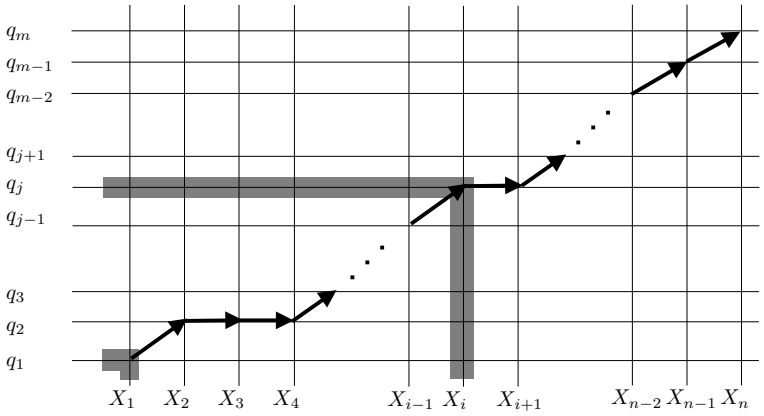

Figure 2.8
Forced alignment of a given feature vector sequence and a sequence of model states.

Providing both the feature vector sequence X and the state sequence Q to the Viterbi decoder the optimal alignment $\hat{A}$ can be found by dynamic programming.

$$\hat{A} = \underset{A}{argmax}\, p(A|X, Q) \tag{2.1}$$

where $p(A|X,Q)$ represents the likelihood of path A through the trellis defined by X and Q. $\hat{A} = ((X_1, q_1), \ldots (X_i, q_j) \ldots (X_n, q_m))$ therefore represents the most likely assignment of each feature vector X_k to a model state q_l. The dynamic programming is illustrated in Fig. 2.8 using the example given in Fig. 2.6 where $\hat{A}$ is represented as a sequence of arrows and the detected beginnings of the words 'a' and 'line' have been marked using gray shadows. Because each feature vector X_k has to be absorbed by exactly one model state and each state q_l consumes at least one feature vector it follows that each A considered in the computation of $\hat{A}$ has to start with the assignment (X_1, q_1) and must end with (X_n, q_m). All elements of the path between these two positions have to respect the restrictions imposed by the topology of the character HMMs involved in the state sequence Q. If the length of the state sequence Q is greater than the length of the feature vector sequence X no path A can be found and hence the forced alignment fails. In all other cases the forced alignment produces a path $\hat{A}$ and the beginning of a word in the normalized text line can be found easily, as the smallest index of all feature vectors assigned to the first state of a given word directly represents the horizontal coordinate for this word.

Since coordinates of word beginnings in normalized text lines do not directly correspond to the coordinates in the original images (as shown in Fig. 2.5) an additional step is required to determine the bounding boxes for the original, non-normalized images. This procedure is described in the following subsection.

2.2.2 Assignment of Bounding Boxes

In the assignment step of the segmentation scheme the connected components of the original text line are assigned to the individual words according to the word boundaries provided by the forced alignment step.

As mentioned above, the resulting coordinates correspond to the beginnings of the words in the segmented text line. It is important to keep in mind that these positions represent the word boundaries for the normalized text line only. The mapping of these boundaries to the word bounding boxes in the original line images has been implemented using an intermediate representation of the text lines.

For the assignment of connected components to words and the computation of the word bounding boxes the following algorithm has been used.

1. Binarize the original, non-normalized image.
2. Extract the connected components and filter out small components.
3. Label every remaining connected component by assigning a unique label value to all its pixels and record the coordinates of its bounding box in the original form image.
4. Apply the normalization scheme used in the HMM based segmentation procedure to the labeled connected components extracted in step 3.
5. Assign the normalized connected components to the word boundaries obtained from the HMMs according to their horizontal positions.
   - If a normalized component falls in between the left and the right boundary of a word, assign it to this word. Please note that the right boundary of a word is identical to the left boundary of the next word (see Fig. 2.9).
   - If a normalized component crosses a word boundary assign it to the word with the larger overlap.
6. Use the bounding box coordinates recorded in step 5 of all connected components assigned to a given word to compute the corresponding word bounding box (defined as the minimal rectangle enclosing all bounding boxes of the connected components).

It could be verified by experiment that connected components belong to a single word or punctuation mark in most cases [6]. Based on this observation word bounding boxes were defined as the minimal rectangle including all connected components of a given word or punctuation character.

[6] Using Otsu's [105] binarization algorithm only 33 exceptions in the 30'785 processed words have been detected. In about 80% of such cases a word in the line above or below has been touched.

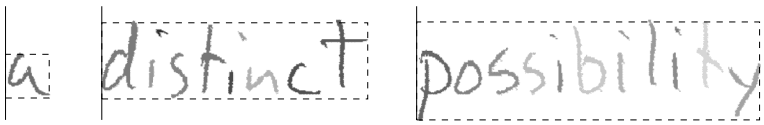

Figure 2.9
Assignment of normalized connected components to word boundaries.

The assignment of the connected components as described in step 5 of the algorithm is visualized in Fig. 2.9. The long vertical lines indicate the word boundaries detected by the HMM based segmentation procedure. The dashed rectangles indicate the assignment of the connected components to the words. The normalized connected components are drawn using unique color values as described in step 3 of the algorithm.

2.2.3 Segmentation Results

A detailed analysis of the word segmentation rate has been made on three subsets A, B, and C of the IAM database. The three subsets include 417 form images, comprising 3,703 lines of handwritten text. Subsets A and B were involved in the development of the segmentation scheme and subset C was used as the test set.

For the manual verification of the segmentation process, an image of each of the processed text lines is generated. The image contains the handwritten text, the corresponding transcription, the word bounding boxes, and the individual word labels as shown in Fig. 2.10.

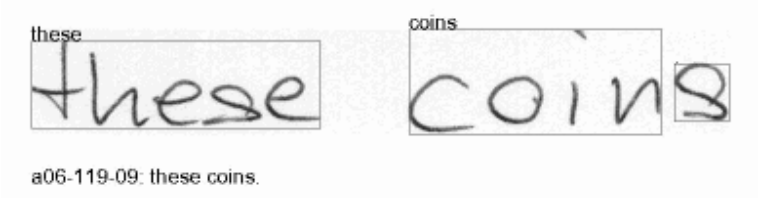

Figure 2.10
Interface for verification and manual correction of transcriptions.

In several refinement steps the HMM based segmentation and the connected component assignment have been improved and transcriptions which did not match the handwritten text have been corrected. Fig. 2.10 shows an example of an incorrect transcription. The writer omitted the final period of the line. Because it is

included in the lines transcription the component assignment step marks the letter 's' as the period.

In all experiments a word was considered to be segmented correctly when its bounding box was minimal, did not truncate any part of the word and the handwritten word image corresponded to the assigned word label. A text line or a complete form was considered to be correctly segmented if all of its words were correctly segmented according to the definition given above.

Set	Forms	Lines	Words
A	162 (62%)	1490 (93%)	11954 (98.3%)
B	116 (53%)	996 (87%)	8357 (98.1%)
C	139 (38%)	1217 (87%)	10474 (97.0%)

Table 2.2 *Correct segmentation rates for forms, lines and words.*

The result of the proposed segmentation procedure is documented in Tab. 2.2. For each of the three considered text categories, the number of forms, text lines and words is given together with the correct segmentation rate. The handwritten texts from category C have been defined as test set. The correct word segmentation rate on this set demonstrates that the proposed segmentation scheme provides robust segmentation results without any manual tuning of parameters. The difference between the correct segmentation rate of set C compared to sets A and B is shown in the error analysis for set C. Almost a third of the errors were caused by wrong transcriptions of text lines. This is due to the fact that no corrections of incorrect transcriptions have been made on set C.

Lines	Errors		Error category
59	122	(39%)	Component assignment
59	93	(30%)	Transcription
25	53	(17%)	Line extraction
31	42	(14%)	Filtering
162	310	(100%)	All

Table 2.3 *Distribution of word segmentation errors on set C.*

Tab. 2.3 provides the error analysis for test set C. It shows the distribution of the detected segmentation errors over four types of errors. Component assignment errors usually result from small normalized connected components ending up at the "wrong" side of a word segmentation boundary.

Line extraction errors typically mix up some connected components or parts of connected components from either the line above or be-

low. Those components are assigned to a word according to the word boundaries obtained in the HMM based segmentation step. This frequently results in a bounding box larger than the minimal bounding box for the word under consideration. Most of the remaining errors are caused by either filtering out too many small components (periods or i-dots, for example) or not filtering some small components which are not part of any word and have been introduced by either the writer or the form scanning process.

The distribution of wrongly segmented text lines for the complete IAM database is given in Tab. 2.4. The number of forms containing no segmentation errors can be found in the first row while the second lists the statistic for the number of forms containing a single segmentation error, etc.

Table 2.4 *Distribution of wrongly segmented text lines per form for the complete IAM database.*

Forms	Err.	Lines	Percentage
583		0	38%
420		1	27%
256		2	17%
159		3	10%
59		4	4%
52	$\geq$	5	4%

2.3 Extraction of Sentence Images

Since most experiments reported in this thesis aim at recognizing complete sentences, the handwritten image data provided in the IAM Database had to be segmented into sentence images. This could be achieved automatically thanks to the following two items stored in the metadata of the form images: Exact word bounding box information is available for most words and special marks for the first word of each sentence have been used in the database.

Fig. 2.11 provides an automatically extracted sentence image including some of the available metadata.

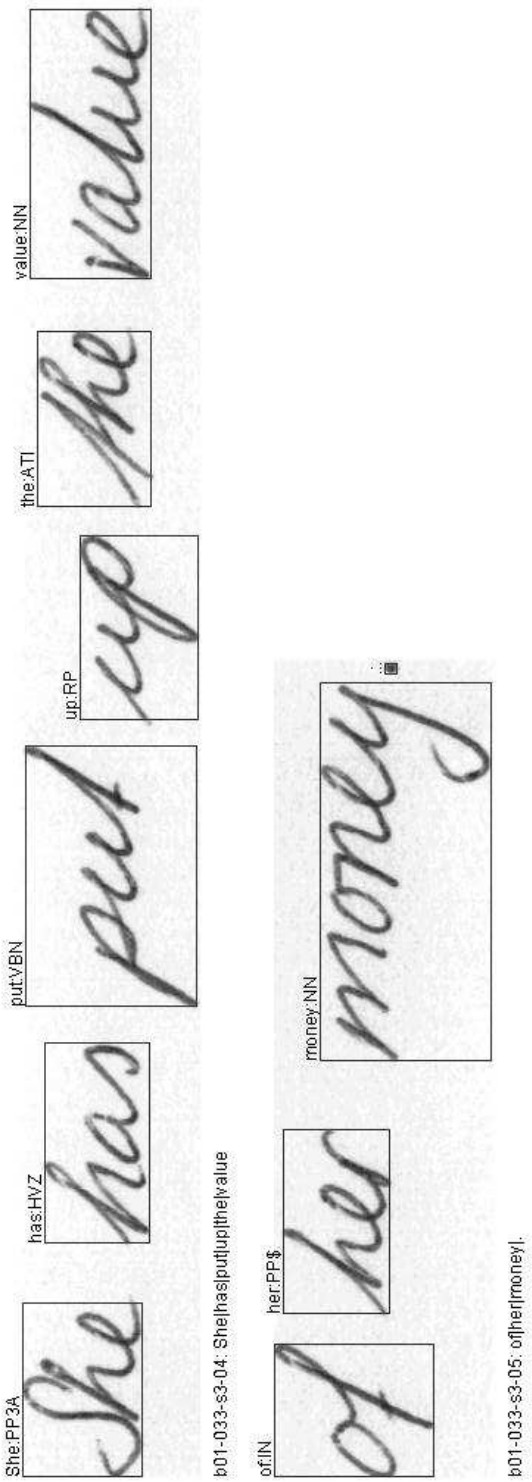

Figure 2.11
An extracted sentence from the IAM database. Additional information like word bounding boxes and the transcriptions have been superimposed.

Linguistic Resources 3

Linguistic resources in the form of text collections (corpora) represent the raw material to produce lexicons and statistical language models. Lexicons are essential since virtually all handwriting text recognition systems found in the literature perform lexicon based recognition. Statistical language models have proved to be effective in improving recognition rates of general text recognition systems.

Additional linguistic information like the grammatical structure of the sentences is available in specialized corpora called treebanks.

In the next two sections the main linguistic resources are presented which were used for the training of the statistical language models of the handwritten sentence recognizer. The treebank from which the stochastic context-free grammars was extracted is introduced in Sec. 3.3. Finally, the unification of the presented resources is addressed in Sec. 3.4.

3.1 The LOB Corpus

The Lancaster-Oslo/Bergen corpus (LOB) [60] contains a wide variety of printed English texts published in 1961. It has been designed by linguists to represent the British English equivalent to the Brown Corpus of American English [35]. The LOB corpus contains 500 printed texts of about 2,000 words each or about a million running words in all. The corpus is available in ASCII format including a part of the formatting found in the original printed text.

Fig. 3.1 provides two sample sentences from the LOB corpus. In most cases characters represent themselves. Some of the more important exceptions include the following. Character ^ indicates the start of a new sentence and character | can represent a new paragraph. The character combination *0 stands for the begin of lower case script, etc. See [60] for the complete description of the coding format.

Figure 3.1
Two sentences from the LOB corpus.

```
B01  56    |^W*2EST GERMANY*- *0followed yesterday by the Dutch*- has made the
B01  57 gesture of a good neighbour. ^She has put up the value of her money.
```

3.2 The Tagged LOB Corpus

The tagged LOB corpus [59] contains the explicit segmentation of the LOB corpus into individual words and provides the grammatical word tag for each word[1]. The tagset used for the LOB corpus is an extended version of the tagset used for the tagged Brown Corpus and covers over 130 different grammatical word classes. See [59] for the complete list and the definition of the word tags.

Examples for tags are NN for nouns like 'money' and VBN for verbs in their base form like 'put'. More specialized tags exist for verbs like 'be' and 'have', e.g. 'has' is tagged HVZ.

Figure 3.2
Two sentences from the Tagged LOB corpus.

```
B01  56    ^ West_NP Germany_NP *-_*- followed_VBN yesterday_NR by_IN the_ATI
B01  56 Dutch_NNPS *-_*- has_HVZ made_VBN the_ATI
B01  57 gesture_NN of_IN a_AT good_JJ neighbour_NN ._. ^ she_PP3A has_HVZ
B01  57 put_VBN up_RP the_ATI value_NN of_IN her_PP$ money_NN ._.
```

Two sentences from the tagged LOB corpus are provided in Fig. 3.2. Please note the differences to the same sentences from the untagged version of the LOB corpus given in Fig. 3.1.

Some examples for the differences between the LOB and the tagged LOB are provided in Tab. 3.1 Major differences between the LOB and the tagged LOB corpus include the normalized capitalization of the words. It ensures that any given word will have the same surface

[1] Punctuation marks like commas, periods, etc. are treated as words in the tagged LOB corpus.

LOB	Tagged LOB
∗2CURIOUS	curious_JJ
^This	^this_DT
\0Mr.	\0Mr_NPT
don't	do_DO n't_XNOT
won't	will_MD n't_XNOT
cannot	can_MD not_XNOT

Table 3.1 *Some difference between the LOB and the tagged LOB corpus.*

form independent of its position in the sentence or the original capitalization in the printed text.

3.3 The Lancaster Parsed Corpus

The Lancaster Parsed Corpus (LPC) [38] is a treebank containing a subset of 11,827 sentences from the LOB corpus for which the result of a syntactical analysis is available. For each analyzed sentence a parse tree[2] in the form of a bracketed sentence is available. Unfortunately the available set of parsed sentences is not quite representative for the LOB corpus since only the first 10 text samples of each text category have been considered for the LPC. Moreover, the sentences contained in the LPC are considerably shorter, on the average, compared to the LOB corpus (11.4 words for the LPC and 19 words for the LOB corpus). The manual of the LPC [38] reports that the probabilistic parser used for the analysis of the sentences was unable to find a parse for most sentences over 20-25 words in length.

```
B01   31
[S[N West_NP Germany_NP *-_*- [Tn[Vn followed_VBN Vn][N yesterday_NR N][P
by_IN [N the_ATI Dutch_NNPS N]P]Tn] *-_*- N][V has_HVZ made_VBN V][N
the_ATI gesture_NN [Po of_INO [N a_AT good_JJ neighbour_NN N]Po]N] ._. S]

B01   32
[S[Na she_PP3A Na][V has_HVZ put_VBN V][R up_RP R][N the_ATI value_NN [Po
of_INO [N her_PP$ money_NN N]Po]N] ._. S]
```

Figure 3.3 *Two sentences from the Lancaster Parsed Corpus.*

Fig. 3.3 provides two bracketed sentences from the LPC. The corresponding parse tree for the top level noun phrase of the second sentence is given in Fig. 3.4.

[2]Parse trees represent the hierarchical structure of its grammatical constituents like noun phrases or verb phrases.

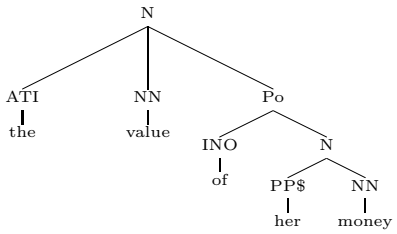

Figure 3.4
Parse tree for a noun phrase 'the value of her money' of the second bracketed sentence of Fig. 3.3.

Incompatibilities between the LPC and the tagged LOB can be found mainly at the tagging level. In some cases additional tags have been introduced (e.g. prepositions `of_INO` and `for_INF` have both been tagged `IN` in the tagged LOB corpus). Idiom tags have been adapted to indicate the position of each word in the word sequence (e.b. `as_CC well_CC" as_CC"` from the tagged LOB corpus is changed into `as_CC31 well_CC32 as_CC33` in the case of the LPC.

3.4 Unification of Linguistic Resources

Although the resources introduced in the last sections are principally based on the same texts, a number of minor inconsistencies exist between the different resources making it impossible to directly work with the provided raw material. In order to extract texts, sets of lexica, n-gram language models and grammars in a consistent way the different corpora need to be unified.

The following two subsections address the unification of the transcriptions of the IAM database and the three corpora presented above and outline necessary adaptations required for the use of the unified resources in the domain of handwriting recognition.

3.4.1 Target Format Requirements

The target format for the linguistic resources has to incorporate the available information from the IAM database, the LOB corpus, the tagged LOB corpus and the LPC. For the explicit segmentation of the texts into words, the tagged LOB corpus has been used, but

the tagset has been taken from the LPC. Since the words should be as close to the transcriptions provided by IAM database as possible[3] the surface forms of the words have been taken from the LOB corpus.

The main differences between the target format and the individual corpora can be summarized as follows. Each sentence is contained in a single line of ASCII text. Inconsistencies between word surface forms or word tags are eliminated and a unique mapping between sentences originating from the IAM database, the LOB corpus, the tagged LOB and the LPC is defined. The last property is important to automatically exclude linguistic resources related to the test set of recognition experiments.

3.4.2 Additional Adaptations

Once the linguistic resources are processed in the way described above. an additional processing of the unified corpora becomes necessary. The goal of this step is to facilitate the use of the resources in the context of a handwriting recognition system. The main adaptations include the treatment of words which have been hyphenated at the end of handwritten text lines and the merging of words like 'cannot' which have been split in the tagged LOB corpus.

Hyphenated words have been marked with variants of the attached tags. A hyphenated version of `Government_:NN` has been tagged as `Govern-_:NN-` and ment_:-NN. The two words `do_:DO n't_:XNOT` have been merged back to the single word `don't_:DOX` which required the introduction of additional word tags for the corresponding negations. As can be seen in the above examples all word tags have been assigned the prefix : which simplifies the separation of word tags and constituent tags.

```
b01-033-s3 She|has|put|up|the|value|of|her|money|.

B01 61 She_:PP3A|has_:HVZ|put_:VBN|up_:RP|the_:ATI|value_:NN| \
  of_:INO|her_:PP$|money_:NN|._:.

B01 32 [ S0 [ S [ Na [ :PP3A She ] ] [ V [ :HVZ has ] [ :VBN put ] ] \
  [ R [ :RP up ] ] [ N [ :ATI the ] [ :NN value ] [ Po [ :INO of ] \
  [ N [ :PP$ her ] [ :NN money ] ] ] ] [ :. . ] ] ]
```

Figure 3.5
A sentence in the different unified format.

[3]The transcriptions represent the format found in the LOB corpus in most cases.

Fig. 3.5 provides the same sentence in the three different formats after the unification process. The first version represents the transcription from the IAM database, the second version corresponds to the tagged sentence found in the tagged LOB corpus and the bracketed version is derived from the LPC. Please note that an additional constituent tag `S0` has been introduced to ensure that all parsed sentences have the same bracketing format. Many sentences in the original LPC start or end with quotation marks. These quotation marks are often placed outside of the independent sentence constituent `S` which leads to incomplete parse trees. The artificial constituent tag `S0` unifies the format of all parse trees and can therefore be associated with the start symbol of context-free grammars.

II Offline Handwritten Text Recognition

Introduction 4

An offline recognition system for unconstrained handwritten sentences is presented. The recognizer will serve as the baseline system for the next two parts of this thesis. In Part III it will be combined with a syntax analysis module and in Part IV the recognizer will be used to incorporate alternative statistical language models.

The sentence recognition system is based on the text line recognizer introduced in [94]. Optimizations and enhancements of the original text line recognizer will be pointed out at the appropriate places in the following text.

Sec. 4.1 addresses the main challenges posed by the task of offline recognition of text. The state of the art in the domain of offline handwriting recognition is summarized in Sec. 4.2. An overview of the sentence recognition system is provided in Section 4.3.

The remaining chapters of this part are organized as follows. First, the methodologies used by the sentence recognizer are introduced in Chapter 5. Experiments and Results are presented in Chapter 6 and conclusions are drawn in Chapter 7.

4.1 Challenges

Offline handwritten text recognition can be seen as the most general case of handwriting recognition. Less assumptions are made than in the other fields of handwriting recognition.

The following subsections will introduce the main challenges posed by this task and briefly describe how these are addressed by the proposed sentence recognition system.

4.1.1 Offline Handwriting Recognition

Offline handwriting recognition systems try to find a correct transcription for a handwritten document which may contain an isolated character, a single word, some general text or represents an address or a bank check. The document is normally available in the form of a gray level image scanned at a relatively low resolution of 200-300dpi.

In contrast to online handwriting recognition the user is free to select any writing instrument and writing pad (e.g. pencil and paper) as long as the resulting document shows some contrast and can be digitized with a scanner. This is the main advantage over online handwriting recognition where the user is typically required to use a special pen on a special device to capture the movement of the pen during the writing process.

4.1.2 Writer Independent Recognition

A writer independent recognition system is optimized to the highest possible generalization regarding both the number of writing styles and writing instruments. In contrast to multi-writer recognition or single writer recognition no handwritten samples of the writers in the test set are available for the training or optimization of the recognition system.

For commercial systems like bank check reading, or handwritten address recognition writer independent recognition is a requirement. It would be infeasible to ask all customers to provide their handwritten samples to train the recognition systems. Instead, real life data is typically sampled over a period of time and used to train and refine the handwriting recognition modules.

Writer independent recognition would also be desirable for single writer environments. It would allow for handwriting recognition software which does not need any further training by the user and could therefore be used out of the box.

4.1.3 Recognition of General Text

The goal of general text recognition consists in finding the correct transcription (sequence of words) for a given image of some handwritten text. Since the correct number of words is not known in advance additional types of errors are often introduced. The source of these types of errors is the unknown segmentation of the text image into isolated word images. A similar segmentation problem - the segmentation of a word image into individual character images - is known as Sayre's paradox [116].

> *To recognize a letter, one must know where it starts and where it ends, to isolate a letter, one must recognize it first.*

Too many or too few words may be present in the result of the recognizer depending on the chosen segmentation of the image into its individual words. When the result contains too many words the recognition process results in an over-segmentation of the input image. In the case where too few words are included in the recognizers solution the term under-segmentation is used.

In addition to the segmentation problem the task of general text recognition includes an additional handicap. No assumption except for the language of the text to recognize is made. The search space for general text recognition is therefore significantly larger than for bank check or address recognition tasks. Instead of a database of valid addresses, lexica and statistical language models are playing the role of contextual knowledge in handwriting recognition systems for general text.

4.2 State of the Art

This section summarizes the state of the art for offline handwriting recognition systems. The focus will lie on the level of complete systems. References for the use of different methods of the individual components of a recognition system like image normalization, feature extraction and classification will be provided at the appropriate places in the following chapters.

As handwriting recognition has been a research topic for over four decades [121] a number of surveys can be found in the literature. Wherever possible references to summaries and overviews will be preferred over the citation of individual works. A general overview for the offline recognition of cursive Roman handwriting can be found in [12]. A survey including both online and offline handwriting recognition can be found in [109].

The following sections concentrate on different fields of the offline recognition of Roman script. First, the domain of isolated character and numeral recognition is considered in Section 4.2.1. Then word recognition and general text recognition are summarized in Section 4.2.2 and 4.2.3. Finally, applications of offline handwriting recognition techniques are covered in Section 4.2.4.

4.2.1 Isolated Character and Numeral Recognition

The recognition of isolated characters represents the typical pattern classification problem. For an unknown input sample the most likely class out of a given set of pattern classes has to be computed. Recognition methods for isolated characters have been surveyed a while ago in [55] and a brief current survey is included in [12].

In the last decade databases like the NIST database [37], the CEDAR database [54], and the ETL-6 database [79] (upper case characters only) became publicly available and are widely used for training and testing of isolated character recognition systems. Among others [71, 107, 108, 136, 137] addressed the problem of offline character recognition in the last year.

Recognition of isolated numerals is a special case of isolated character recognition. Two properties of isolated numeral recognition make this field to the perhaps most frequently addressed problem of handwriting recognition research. First, the small number of classes (ten) makes the problem accessible for most known classifiers, and second, large amounts of isolated handwritten digits have been available for several years. Most notably the NIST, the MNIST [31], the CEDAR, and the CENPARMI databases [132]. Recent publications which made use of these data-bases include [7, 86, 108, 110].

4.2.2 Word Recognition

Handwritten word recognition is recognized as a more difficult task than the recognition of isolated characters since the problem of the segmentation of an input word image and the recognition of the resulting word characters must be solved. For the recognition of handwritten words specific surveys are available in [126] and more recently in [12, 139]. Compared to the case of isolated character recognition fewer publicly available databases exist for training and testing of offline handwritten word recognition systems. In the domain of postal applications the CEDAR database is widely used. Recently a segmented version of the IAM database [96] became also available [153].

The commonly used criteria to classify the different word recognition methods is the technique applied to segment the word image into individual characters. Segmentation-based methods attempt to first segment a handwritten word image into individual characters and then use a classifier for handwritten characters to recognize the resulting images. A recent implementation of this strategy without lexical postprocessing can be found in [71]. Lexical support has been added in [72].

The majority of current handwritten word recognition systems are based on Hidden Markov Models (HMM) which implement a segmentation-free recognition. This means that the segmentation of the word image and the recognition of the characters are performed in a single step. Most current publications concentrate on the optimization of already existing systems [11, 44, 100, 117, 118, 154]. Newer attempts to combine HMM with neural networks can be found in [22, 78].

4.2.3 General Text Recognition

Offline recognition of handwritten text can still be considered a relatively new research topic. Only few works published in the literature address the problem of general text recognition. In most cases either a sequence of isolated word images is assumed or text lines are first split into a sequence of isolated words which can then be fed to a handwritten word recognition system [10, 28, 71, 120, 149].

Systems which avoid an explicit segmentation of text lines into individual words and make use of a statistical language models to support the handwriting recognition system can be found in [95, 143, 155].

4.2.4 Applications

Most applications of offline handwriting recognition technology reported in the literature are found in the domain of address recognition used for mail sorting and in the domain of bank check reading for the automatic processing of financial transactions. Automatic address reading deals with the recognition of an address found on a parcel or a piece of flat mail like letters or cards. An overview of the problem is provided in [125] and a comparison of systems can be found in [26]. Furthermore related problems like the detection of the address block [97, 43], the orientation of a mail piece [103], or the location of a numerical sequence (e.g. the zip code) [77] have been investigated.

In the case of bank check recognition the necessary elements for the automatic processing of the financial transaction are tried to be recognized. A family of commercially available multilingual check reading systems is presented in [41, 40]. In this domain the automatic segmentation and recognition of a multi-lingual date field proved to be particularly difficult and has been addressed in [147].

The recognition of historical documents can be seen as a third potential application of offline handwriting recognition technology. So far, no successful attempt has been made to automatically transcribe such documents but an increasing amount of publications investigate different aspects related to this task. Methods to separate text pixels from background pixels can be found in [82, 145]. Further works include the mapping of transcriptions to the corresponding image regions [133], a word spotting technique [112], and the segmentation of handwritten dates in historic documents [32].

4.3 System Overview

The core of the presented recognition system for offline handwritten sentences is based on the Hidden Markov Modeling technique.

On the top level, the system can be divided into preprocessing, recognition and postprocessing modules as illustrated in Fig. 4.1.

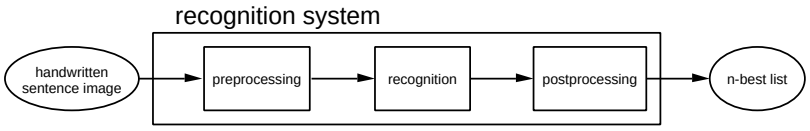

Figure 4.1 *Handwriting recognition system overview.*

The following three subsections detail the preprocessing, the recognition, and the postprocessing modules of the recognition system. Brief description of the HMM training and language model extraction are provided in the last two subsections.

4.3.1 Preprocessing

In the preprocessing module the handwritten text image is first segmented into text lines. From the text lines sentence fragments are then extracted and normalized.

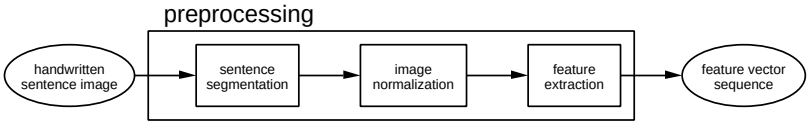

Figure 4.2 *Preprocessing of the handwriting recognition system.*

The image normalization is performed to remove a part of the variance introduced by the writing styles of different writers. The normalized images are finally transformed into feature vector sequences by the feature extraction module as shown in Fig. 4.2.

4.3.2 HMM Based Recognition

The Hidden Markov Model based recognition module produces a recognition lattice[1] for each feature vector sequence which represents a sentence fragment.

The recognition is performed by the Viterbi decoding step as can be seen in Fig. 4.3. For the decoding step a number of resources

[1]Recognition lattices can contain a large number of recognition alternative solutions for the input image and may be interpreted as finite state automata.

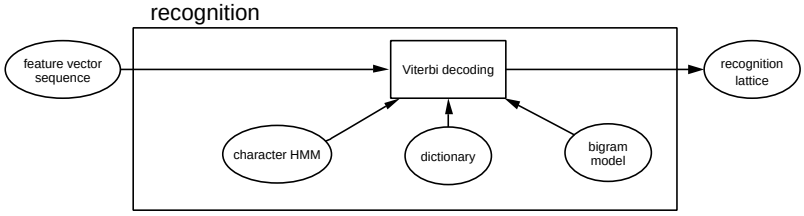

Figure 4.3
Recognition module of the handwriting recognition system.

are required. Most important, the trained Hidden Markov Models (HMM) which represent the different character classes. Additionally, a dictionary is used to map the words to corresponding sequences of character HMMs, and finally the bigram model, a statistical language model. This model is used by the Viterbi decoding step to prefer frequently observed word sequences over word sequences which are not typical for the language under consideration.

4.3.3 Postprocessing

In the postprocessing module the set of recognition lattices generated for the sentence fragments are merged to a single sentence lattice. From this lattice the n most likely candidate sentences with their corresponding recognition scores are then extracted and compiled into the n-best list as shown in Fig. 4.4.

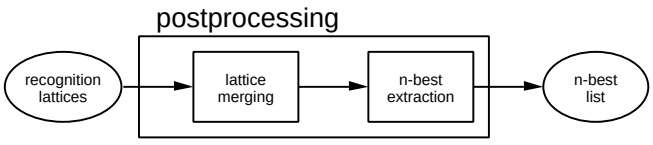

Figure 4.4
Postprocessing of the handwriting recognition system.

4.3.4 Language Model Extraction

For the extraction of the language model the following resources must be available. First, the transcriptions of the handwritten text images are used to ensure that all words of the handwritten samples will appear in the dictionary. Then the tagged LOB corpus is used to increase the size of the dictionary up to a predefined number of

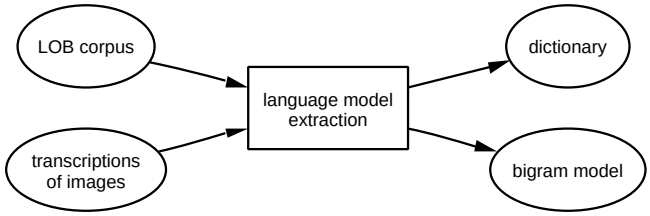

Figure 4.5
Extraction of the bigram language model and the dictionary.

words. Finally, the tagged LOB corpus is also used to build the bigram language model.

Fig. 4.5 illustrates the process of dictionary generation and language model extraction.

4.3.5 HMM Training

The procedure for the character HMM training is shown in Fig. 4.6. For the generation and training of the character HMM both the feature vector sequences (as provided by the preprocessing step) and the corresponding transcriptions are used. The dictionary obtained by the language model extraction is required to convert the transcriptions into the corresponding sequence of character HMMs.

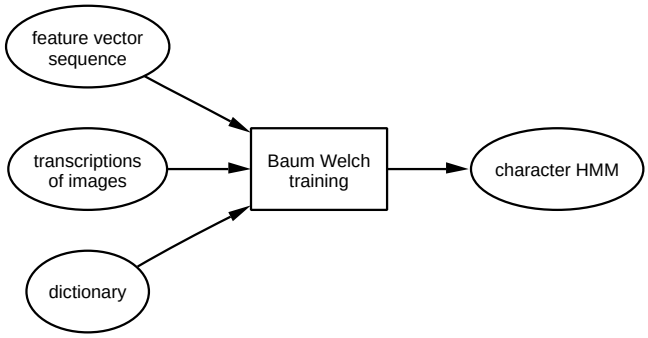

Figure 4.6
Construction and training of the character HMM.

The initial character HMMs are then iteratively trained using embedded Baum-Welch training.

5 Methodology

The methods used in the main modules of the handwriting recognition system are described in this chapter. The sections are organized as follows. After the description of the preprocessing step the image normalization is detailed in Sec. 5.2. The extraction of the feature vector sequences is briefly explained in Sec. 5.3. Hidden Markov modeling is covered in Sec. 5.4 and more specifically, the optimization of the length of the characters HMM is addressed in Sec. 5.5. The last four sections cover the bigram language model in Sec. 5.6, HMM decoding parameters in Sec. 5.7, recognition lattices and n-best lists in Sec. 5.8, and the measurement of the system performance in Sec. 5.9.

5.1 Preprocessing

The task of the preprocessing module consists in the extraction of sentences from the handwritten texts provided by the IAM database. The handwritten sentences have been automatically extracted from the IAM database using the procedure described in Section 2.3. It is important to notice that in contrast to the extraction of handwritten lines of text it is generally not possible to automatically extract sentences from a handwritten text without recognizing the text itself.

As a result of this preprocessing step a sequence of consecutive handwritten text line images will result. Fig. 5.1 illustrates the

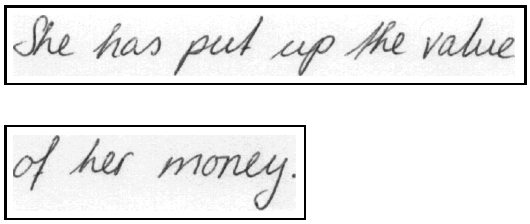

Figure 5.1
Sentence b01-033-s3 from the IAM database after preprocessing.

result of the preprocessing step. Considering the good quality of the scanned images no additional preprocessing steps were necessary.

5.2 Image Normalization

In the image normalization step the variability between different writing styles is reduced without destroying information relevant to the recognition of a handwritten text line. The image normalization consists of a set of independent normalization procedures which include slant removal, slope removal the normalization of the main writing zones, and contrast normalization.

5.2.1 Slant Removal

The goal of the slant removal operation is to normalize the direction of the long vertical strokes in characters like 'b', 'f', 'l' etc. Depending on the individual writing style the vertical strokes of these characters will be more or less slanted to the right or the left. Slant removal is always achieved by two steps. First, the slant angle is estimated form the input image and then the image is deslanted using a shear transform with the estimated angle obtained in the first step.

A large number of slant removal methods have been proposed in the literature. See [10] for a technique based on the selection of near-vertical strokes. Many methods define a measure for "deslantedness" which is computed for all possible slant values in a predefined interval. The angle which achieves a maximum for the measure is then selected as the estimate for the slant [14, 69, 95, 102, 141]. All methods mentioned above provide a single estimate for the slant angle and make therefore the assumption that the slant does not vary within a word or line of text. Only few methods have been suggested which are based on local slant estimation and removal [138].

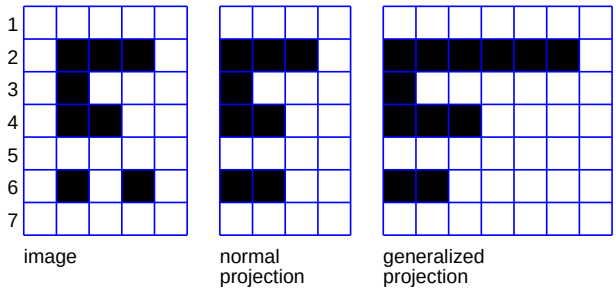

Figure 5.2
Comparison of the normal and the generalized projection for $\alpha = 0$ of a sample image.

The selection of the slant normalization method published in [102] was motivated by the claim of the authors that generalized projections were applicable not only for the estimation of the slant angle but also the slope angle and estimation of the writing zones for cursive handwritten words. Unfortunately this could not be verified for the estimation of the slope angle and the writing zones. However generalized projections proved to be accurate to estimate the slant angle for a large amount of different writing styles.

After binarization of the input image a histogram using generalized projections is computed for each slant candidate angle α. Generalized projection emphasize long runs of foreground pixels and differ from normal projections in the assignment of weights to the foreground pixels. While an equal weight of 1 is assigned to all foreground pixels in the case of a normal projections, the weight of a foreground pixel in general projections depend on the weight of the predecessor pixel. As long as there are subsequent foreground pixels the weight is increased by 1 for each new pixel. When a background pixel is encountered the weight is reset to 1. See Fig. 5.2 for the illustration of the difference between a normal projection and a generalized projection for angle $\alpha = 0$. On the right hand side the generalized projection count $gp_0(i)$ is indicated, where $gp_0(2) = 1 + 2 + 3 = 6$.

For each histogram the sum h_α of all resulting pixel counts are recorded in the following way.

$$h_\alpha = \sum_i gp_\alpha(i) \tag{5.1}$$

where $gp_\alpha(i)$ represents the generalized projection count for direction α and histogram column i. For the example shown in Fig. 5.2 the sum of the pixel counts is $h_0 = 12$.

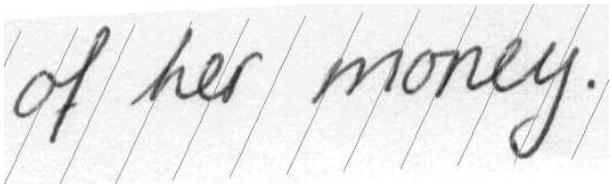

Figure 5.3
Estimation of the slant superimposed over the text line image.

Finally the slant is estimated by the angle α_{slant} which maximizes h_α over the predefined range of candidate angles as shown below.

$$\alpha_{slant} = \underset{\alpha}{argmax}\, h_\alpha \tag{5.2}$$

The implemented slant estimation is sampling the candidate angles from 40 to 120 degrees measured counter-clockwise from the horizontal with a step size of 2 degrees.

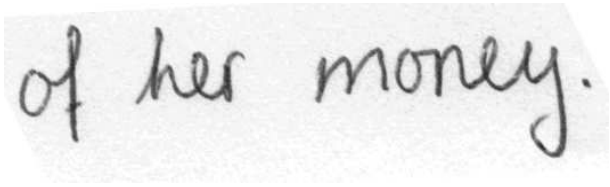

Figure 5.4
The deslanted version of the text line.

Fig. 5.3 shows the estimated slant angle which has been superimposed over the text line image. The resulting text line after slant removal is given in Fig. 5.4

5.2.2 Slope Removal

The slope removal, also called skew correction, tries to eliminate the difference between the horizontal and the imaginary line on which the words of the input text line have been written to. As in the case of slant removal, the slope removal consists of two steps. First, the slope angle is estimated and second, the input image is normalized to a horizontal slope using either a rotation or a shear transform.

Most methods to estimate the baseline are based on linear regression for a set of points which are supposed to represent the position of the baseline in average. The selection of the points can be based on a previous estimation of the core region [10, 140], on the preprocessed image using "smearing" [66, 122], or directly on the image [95]. In many cases linear regression is not robust enough when

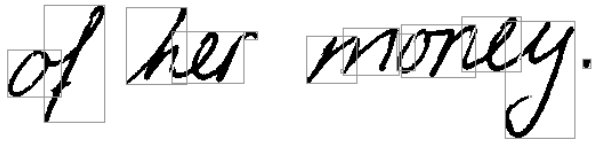

Figure 5.5
Bounding boxes for the splitted connected components.

outliers are present in the point set and an iterative removal of the outliers takes place until the remaining mean square error is below a predefined threshold. As in the case of slant estimation a single linear baseline is provided by most baseline estimation techniques. While in certain cases this approximation is reasonable, especially in cases where the writer has to write on given printed line, there are situations where a single linear baseline does not seem to be adequate. Some solutions to local baseline estimation have been proposed in [88, 99, 148]. A different approach is suggested in [70] which is based on the Wigner-Ville distribution and works on a small set of text lines or an entire document.

For the recognizer described in this thesis the initial point set is defined by the lower left corners of the bounding boxes around the modified connected components of a line of text. Since connected components can become very large for cursive handwriting[1] wide components are split into several pieces before the bounding boxes are computed as shown in Fig. 5.5.

The baseline is then estimated using linear regression for the set of the lower left corner of the resulting bounding boxes. Iterative removal of outlier points is then applied until the mean square error is below a predefined threshold.

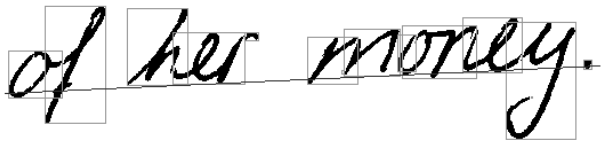

Figure 5.6
The resulting estimate for the base line obtained through linear regression for the lower left corners of the bounding boxes.

Fig. 5.6 shows the result of the baseline estimation process as described above. The resulting text line image after deslanting and

[1] A word may be represented by a single connected component.

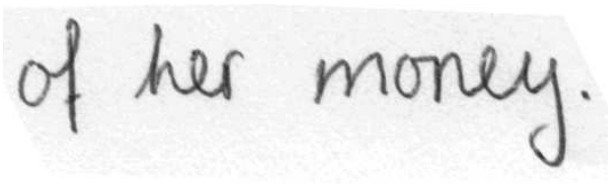

Figure 5.7
The deslanted and deskewed text line.

deskewing is given in Fig. 5.7

The motivation behind this slope normalization technique lies in the simple extension for the estimation of the main writing regions as will be described in the following subsection. Please note that the proposed method would be badly affected by a heavily fragmented input image. However, this problem has not been observed for the handwritten text lines provided by the IAM database.

5.2.3 Normalization of the Writing Regions

Once the slant and the slope of a text line image have been normalized the reference lines are detected which vertically separate a handwritten word or text line into the main writing regions, i.e. the ascender, the core, and the descender region. The estimated reference lines can then be used to normalize the main writing zones. This makes the feature extraction more robust since the location of handwritten strokes is closely correlated with the corresponding characters[2].

For the detection of the core region most published works propose a technique based on the horizontal projection of the binarized version of the input image [10].

The method implemented in this thesis to estimate the upper baseline is a modified version of the slope estimation method introduced in the last subsection. Instead of taking the lower left corners of the bounding boxes (from the splitted connected components) the linear regression is based on the upper left corners. First, the median of height of all bounding boxes is computed as an estimate for the height of the core region. Then the upper left corner of

[2]Characters like 'a', 'c' and 'm' are written completely in the core region while digits, upper case letters and characters like 'l' and 't' also use the ascender region. The descender region is used by letters like 'f', 'g' or 'y'.

a bounding box or the lower left corner with the subtracted core height is added to the point set for each bounding box. The decision to select between the two alternatives is based on the vertical position of the two points. Always the point with the lower vertical position is used.

The ascender and the descender lines are then be determined as follows. For the angle a weighted sum of the baseline and the upper baseline angle is used giving more weight to the baseline which can usually be estimated more accurate than the upper baseline. The position can then be fixed to the extreme points of all bounding boxes.

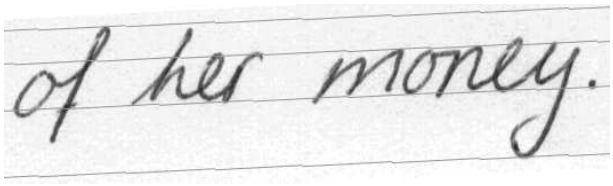

Figure 5.8
Estimation of the main writing zones superimposed over the original text line.

Fig. 5.8 provides an example for the result of the reference line estimation process. The resulting text line after the normalization of the main writing region (all regions are normalized to have the same height) is provided in Fig. 5.9. In addition to the normalization of the writing regions also the contrast of the image is normalized using simple thresholding. Pixels with a gray level below a fixed lower threshold will be set to black, pixels with a gray level higher than a fixed upper threshold are set to white. Gray levels between these two thresholds are then linearly interpolated between black and white.

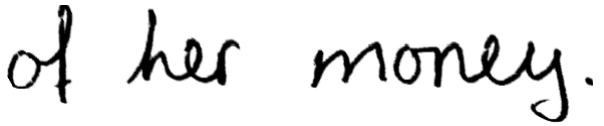

Figure 5.9
The normalized text line image including contrast normalization.

The normalization of the contrast and the main writing regions are the last normalization steps involved in the implemented recognizer. Further normalization steps like the normalization of the stroke width or the character width could also be applied but have not been investigated in the context of this thesis.

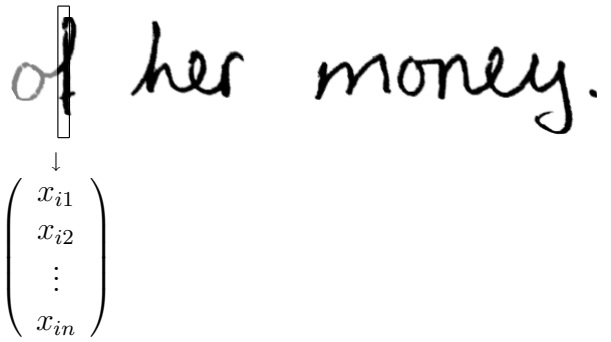

Figure 5.10
The extraction of a feature vector X_i at position i using the sliding window approach.

5.3 Extraction of Feature Vector Sequences

The feature extraction step has to transform the normalized text line images into a format suitable for the selected recognition scheme. The use of Hidden Markov Models in the recognition step requires an input in the form of a sequence of feature vectors sometimes also called observation vectors. The standard method to obtain a feature vector sequence from an image is called the "sliding window" technique.

Fig. 5.10 illustrates the sliding window approach. A narrow window is moved from the left to the right of the image. At each position the content of the window is used to extract a feature vector. The systems proposed in the literature based on this technique differ both in the selected window parameters and the extracted features. Window parameters are the width of the window, and the step size which is specified by the horizontal distance between two consecutive positions of the window.

Many different feature extraction methods have been proposed in the literature. The different approaches can mainly be characterized by the type of the extracted features. [66, 143] are using foreground pixel densities measured in individual cells of the partitioned input window, where the cells are typically arranged in grid of n columns and m rows. Typically the estimated reference lines determine the vertical partition of the window. Other feature extraction approaches use geometric quantities like moments, coefficients from series expansion or first and second order moments and the position of the upper and the lower contour as described in [95]. Finally, structural features like stroke directions [120] endpoints, forks, and

loops [13] have also been suggested.

In [92] three different types of features have been described and the performance of the corresponding systems was compared. First, the raw pixels of the window were taken as input to the Karhunen-Loève transform[3] to reduce the dimensionality of the resulting feature vectors. Second, the limits of the linear transform were avoided using a simple feed forward multi layer perceptron (MLP) as described in [119] and finally, geometrical features as described in [95] were used. On a small text line recognition task with a vocabulary of 412 words a word recognition rate[4] of 52% was reached for the KL-transformed window, 68% for the system using the features obtained by the MLP transformed window, and 80% for the geometrical features respectively.

For the recognizer implemented in this thesis the geometrical features as described in [95] have been used. For each image column i a feature vector $X_i = (x_{i1}, x_{i2}, \ldots x_{i9})$ containing geometrical features is extracted, which signifies that the sliding window has width 1 and is moved 1 pixel per step. The first three features contain the number of foreground pixel in the window, as well as the first, and the second order moment of the foreground pixels. Features four to seven contain the position of the upper and the lower contour, and the first order derivative from the upper and the lower contour respectively. The last two features contain the number of vertical black white transitions and the pixel density between the upper and the lower contour.

5.4 Hidden Markov Modeling

Several important aspects of the Hidden Markov Model (HMM) technique are introduced in the following subsections. First, some fundamental issues are presented. Then, model topologies, emission probabilities and path probabilities are covered. Finally, the applied training and decoding algorithms are mentioned.

[3]The Karhunen-Loève transform, also called KL-transform is a version of the principal component analysis (PCA) [61], a popular technique for dimensionality reduction

[4]See Section 5.9.2 for the definition of the word recognition rate.

5.4.1 Fundamentals

The HMM technique[5] is based on a mathematically solid foundation and especially suited for the recognition of a sequence of observations[6]. One of the advantages the HMM framework lies in its capability to perform segmentation and recognition at the same time. This is a significant advantage over systems relying on a segmentation-followed-by-recognition scheme. A further advantage of HMMs lies in the fact that individual models[7] can be trained using global data only. A prior segmentation of the data into its model-specific parts, like characters or words, is not required. It is sufficient to provide the image data and the corresponding transcription (word sequence).

When the HMMs are used for the recognition of a handwritten text the goal is to find the most likely sentence $\hat{s} = (w_1, w_2 \dots w_n)$ for a given observation sequence $X = (X_1, X_2, \dots X_m)$ provided by the feature extraction step.

$$\hat{s} = \underset{s}{argmax}\, p(s|X) \tag{5.3}$$

Unfortunately a HMM based classifiers can only be used to directly compute an estimate of the likelihood $p(X|s)$ for a given hypothesis s. To compute the desired quantity the Bayes' rule can be applied as provided in Eq. 5.4.

$$p(s|X) = \frac{p(X|s)p(s)}{p(X)} \tag{5.4}$$

Since the probability of the observation sequence $p(X)$ remains constant for all possible interpretations the most likely word sequence $\hat{s}$ can be found by rewriting Eq. 5.3 as follows.

$$\hat{s} = \underset{s}{argmax}\, p(X|s)p(s) \tag{5.5}$$

[5] See [111] chapter 6 for an overview and theoretical introduction to the field. A more practical view can be found in the first chapter of [150], a manual for a widely used HMM toolkit.

[6] The observations can be represented by both symbols or feature vectors.

[7] Individual models may represent characters in the case of offline handwriting recognition, strokes for online handwriting recognition or phones in the case of speech recognition.

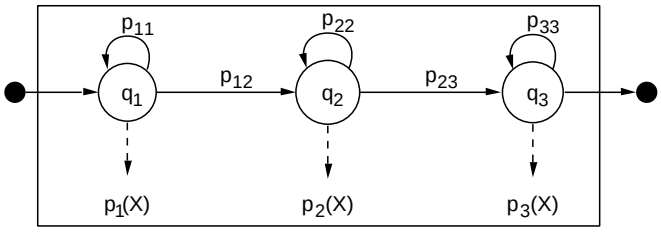

Figure 5.11
A linear left-to-right HMM.

Therefore, the result of the HMM classification needs to be combined with the sentence probability $p(s)$ which can be estimated using a statistical language model[8].

The following subsections will address the main aspects of the HMM framework which is needed to compute $p(X|s)$. Model topologies are covered in the next subsection, the computation of emission probabilities is explained in Sec. 5.4.3 and Sec. 5.4.4 includes the calculation of path probabilities. The language model probabilities will then be addressed in Section 5.6 and more specifically Part IV of this thesis.

5.4.2 Model Topologies

The topology of a HMM is described by the set of model states $Q = \{q_1, q_2, \dots q_m\}$, the transition probabilities p_{ij} which define the probability to move from state p_i to p_j and the initial state probability distribution where π_i specifies the probability to start the model in state q_i.

Fig. 5.11 provides an example of a linear topology [3, 57] of a HMM for modeling a character where $\pi_1 = 1$ and $p_{ij} = 0$ for $i > j$ or $j > i+1$. The topology of the above example consists of an initial state on the left, three emitting states in the middle and an exit state on the right[9]. The allowed transitions between the states are indicated with the arrows where only self-transitions and transitions to the next state are allowed. The estimation of the emission probabilities $p_i(X)$ for a single feature vector X is indicated with the dashed arrows.

[8]Statistical language models are covered in more detail in Part IV. The special case of bigram language models is addressed in Sec. 5.6 of this chapter.

[9]Only emitting states are theoretically necessary. The non-emitting initial and exit states are convenient to concatenate individual character models for the construction of word models.

The selection of the topology of HMM can still be considered to be an open issue and many HMM based systems found in the literature still use a fixed topology for all modeled characters. Only few attempts were made to derive the HMM topologies from the data [81, 131]. A model merging strategy is described in [131] which has been applied to the recognition of spoken words. In the case of online handwriting recognition different strategies are proposed. The construction of a multi-branch HMM in combination with a state tying scheme is documented in [81]. References for the use of linear topologies can be found for the recognition of online handwritten Hangul characters [123], the recognition of faxed machine printed words [30], offline handwritten numeral strings [62] and offline cursive handwriting [13, 95, 143]. Sometimes additional transitions are added to allow the model to skip one or more states [46, 84]. In a few cases more complex topologies are used, called multiple parallel-path HMMs [81] or multi-branch HMMs [146], which both use HMMs with left-to-right topologies as their building blocks. Recently some new HMM model selection techniques have been proposed in [8, 117, 118]. A discriminative information criterion is proposed in [8] to select a character model among competing HMMs. Model-length adaptation is described in [118] and the determination of the number of writing variants of single characters is addressed in [117]

Although the linear topology can be seen as the most simple topology it has a number of advantages. Most important, there is only a single free parameter describing the model topology, the number of emitting states, which allows for simple topology optimization schemes[10]. From the practical point of view linear modeling seems to be the dominating topology used in speech recognition systems and has also proved to work well for the modeling of handwritten characters.

5.4.3 Emission Probabilities

In an emitting state q_i of an HMM the probability of a given observation $p_i(X)$ is computed. Depending on the nature of the observation, emitting states are modeled using either discrete probability distribution or continuous probability density functions (PDF).

[10] See Section 5.5 for different optimization schemes using the linear topology.

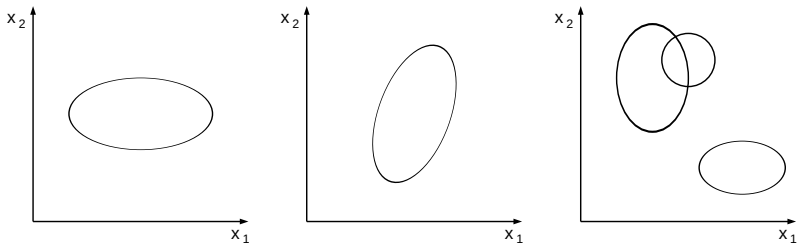

Figure 5.12
Single Gaussian with a diagonal (left), a non diagonal covariance matrix (middle) and a mixture of three Gaussians (right).

If the HMM models a sequence of symbols the HMM are called discrete, and in the case of feature vector sequences the term continuous density HMM is used respectively. As a third variant, so called semi-continuous HMM exist. In contrast to continuous HMM where each emitting state has its own set of PDF, a pool of PDF is shared among all states of a semi-continuous HMM. A specific set of mixture weights is then assigned to each such state to define its own multivariate PDF. All types of HMM have been used for offline handwriting recognition. See [46, 100] for the application of discrete HMM, [11, 13, 68] for semi-continuous HMM and [95, 143] for continuous HMM.

For the case of continuous HMM for d-dimensional feature vector sequences any d-dimensional multivariate PDF to compute the vector probabilities may be used. The PDF can be modeled with a simple multivariate Gaussian density [95] or a - often better performing - mixture of Gaussians[11] [143].

In the case of a d-dimensional single Gaussian density function $p(x|\mu, \Sigma)$ is used to estimate the emission probability of a d-dimensional feature vector X, where μ specifies the mean vector and Σ the covariance matrix. For the multi-Gaussian case with m mixture components the following probabilistic model is assumed:

$$p(x|\Theta) = \sum_{i=1}^{m} \alpha_i p_i(x|\theta_i) \tag{5.6}$$

where $\Theta = (\alpha_1, \ldots \alpha_m, \theta_1, \ldots \theta_m)$ represents the complete set of parameters involved. The individual components are represented by $\theta_i = (\mu_i, \Sigma_i)$ and the mixture weights must satisfy $\sum_{i=1}^{m} \alpha_i = 1$.

[11] See Section 6.3 for a comparison of the performance of using single Gaussians or multi-Gaussians to estimate the output probabilities.

Fig. 5.12 illustrates three different types of probability density functions. On the left, a single, two-dimensional Gaussian density with a diagonal covariance matrix is shown. In the middle, a single Gaussian with a non-diagonal covariance matrix is provided and on the right, a mixture of three Gaussians with diagonal covariance matrices are plotted.

In the general case a d-dimensional mixture Gaussian model with m mixture components can be specified by $m(d+d(d+1))/2+m-1$ parameters. For the complete HMM classifier including n emitting states the number of parameters becomes then $n(m(d+d(d+1))/2+m-1)$. In order to reduce the number of parameters and to improve the estimates for the majority of the parameters diagonal covariance matrices are typically used[12]. The use of diagonal covariance matrices implies that the individual features of a feature vector X are assumed to be uncorrelated. Since for most feature vector types this is not the case the feature vectors should therefore be decorrelated before they are fed into an HMM system using diagonal covariance matrices. See [142] for a comparison of different methods to decorrelate feature vector sequences.

5.4.4 Path Probabilities

The computation of the path probability $p(A)$ for a path $A = (A_1, A_2 \dots A_n)$ includes both the state transition probabilities and the emission probabilities where a path element $A_t = (X_t, q_i)$ maps the feature vector X_t to the model state q_i. Eq. 5.7 provides the calculation of a particular path A

$$p(A) = \prod_{t=1}^{n} p_{t,t-1}\ p_t(X_t) \tag{5.7}$$

where q_t denotes the current state at time t. $p_t(X_t)$ defines the emission probability for the feature vector X_t at this state and $p_{t,t-1}$ specifies the corresponding probability to move from state q_{t-1} to state q_t.

[12]For the recognition system implemented in this thesis the number of parameters can be reduced from over 430,000 to 150,000 values if the covariance matrices are restricted to the diagonal.

The most likely path $\hat{A}$ computed in the forced alignment mode of a Viterbi decoder can then be found by dynamic programming as previously described in Sec. 2.2.1 of Part I.

5.4.5 Training

For the training of the HMM the Baum-Welch training algorithm[6, 5] is used is most cases which will iteratively optimize the parameters of the provided models. This training scheme can be seen as a variant of the Expectation-Maximization (EM) algorithm [27].

5.4.6 Decoding

In the decoding step of the recognition phase the Viterbi algorithm [34, 144] is generally applied[13]. An alternative formulation of the Viterbi algorithm is the token passing mechanism [151] which allows to produce not only the most likely interpretation of the input observation sequence, but also a lattice containing the n-best hypothesis.

5.5 Character Specific Model Length Optimization

In contrast to the optimization of the parameters of an HMM, its topology (the number of model states and the possible transitions between the states) needs to be specified in advance and remains fixed during the training. The HMM framework does not provide any optimization of the topology.

In the case of a recognition system for isolated (spoken) words using left-to-right HMMs two schools of thought regarding the selection of the number of states are mentioned in [111]. The first is based on the idea to let the number of states roughly correspond to the number of sounds (phonemes) within the word. The other idea is named after R. Bakis and consists in selecting the number of model states proportionally to the average number of observations of the

[13]Therefore, the term Viterbi decoding is often used for the recognition step.

corresponding training samples [3]. In speech and online handwriting recognition systems where a sequential, time dependent signal has to be recognized, the first idea is adopted in most cases. In such applications the model states are normally related to the stationary parts of a time depending signal. In the case of speech recognition systems phones are defined by the stationary parts of the signal. For online handwriting recognition systems strokes are normally associated with the stationary parts of the signal. In theory one state per phone or stroke model would suffice since a single output density function can model any stationary signal. While such models are simple they produce an exponential state duration distribution which is normally not adequate to model phone durations or stroke lengths. In order to cope with this deficiency explicit state duration modeling has been suggested in [33] and [83].

In the case of offline handwriting recognition no time depending signal is available but the sliding window mechanism can be used. Following this approach time dependent observations are replaced by observations depending on the horizontal position of the window. As a consequence the mapping between the model states and the stationary parts of the signal is not obvious anymore and no commonly accepted idea exists what a HMM state should correspond to in the case of offline handwriting recognition. In a black box approach characters are most frequently modeled by a single HMM with a linear topology. Such a topology provides a sequence of emission probability functions which can cope with complex character shapes and a flexible character duration modeling.

For the commonly used sliding window technique, character HMMs used in [94, 98, 143] have a fixed (globally optimized) number of states. The use of model specific numbers of states using the Bakis-model is mentioned in [8, 30, 46, 81]. Although [81] investigates online recognition of Hangul characters it was the only publication found which provided a direct comparison of the recognition performance for both a fixed and a character specific number of states.

In the following subsections three different methods to optimize the number of states for the linear left-to-right topology are compared. The first method optimizes the HMMs using a fixed number of states for all character models. The second method represents the Bakis-model where a given fraction of the average character length determines the number of states for the corresponding HMM. The third method is called quantile modeling and is motivated by the

Figure 5.13 *Character segmentation using forced alignment.*

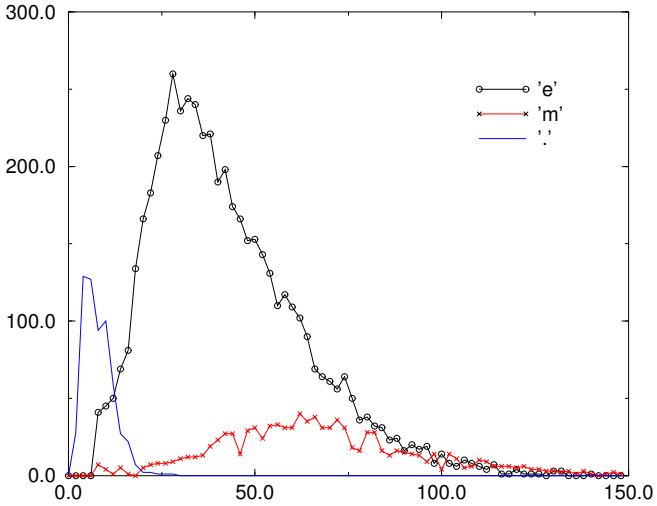

Figure 5.14 *Length histograms for some characters.*

concept of minimum duration modeling where the number of states for each character HMM is defined by a specified quantile of the character length histogram. The two latter methods require statistics about the character specific number of feature vectors. These statistics can be obtained by segmenting handwritten words into its characters using a Viterbi decoder in forced alignment mode. Fig. 5.13 illustrates the result of of this

Using the estimated character lengths of 9'929 word images from the IAM database the obtained length histograms for the three characters 'e' 'm' and '.' are plotted in Fig. 5.14 where the x-axis corresponds to the character width in pixels and the y-axis to the number of samples. The different heights of the three curves reflect the fact that roughly 5000 samples of 'e', 1000 samples of 'm' and 600 samples of '.' were present in the set of the 9'929 words. To ensure that the character lengths were estimated in a reliable way, the number of states of the character HMM had to be reduced significantly for the forced alignment experiments. This has been done because of the following reason. A character model

with a linear topology and n states will never measure a length of less than n feature vectors for any instance of the corresponding character Therefore, ten or even less states have been used for most character HMM of the forced alignment system.

5.5.1 Fixed Length Modeling

For the fixed length modeling no assumptions are made at all. Instead of trying to predict a good length for the individual character HMM all models are assigned the same length (number of states).

The optimal number of states can then be found by measuring the recognition rate of the word recognizer for each possible number of states. To speed up the method, the search range can be constrained by the use of empirical knowledge.

5.5.2 Bakis Length Modeling

For the Bakis modeling method it is assumed that the length of the HMM should depend on the available training data for the individual models in the following way. For each HMM the length (number of states) is set to a fraction of the average number of observations (feature vectors) of the corresponding samples in the training data.

The optimal number of states per model can then be found by measuring the recognition rate of the word recognizer for different fractions of the average character lengths. To speed up the search for the optimal fraction it can be considered that for the left-to-right topology a model will only accept samples containing at least as many feature vectors as its number of states. Consequently a fraction of 1.0 would reject already half of the samples. Therefore, the optimal fraction can be found between 0.0 and a number significantly smaller that 1.0.

5.5.3 Quantile Length Modeling

The quantile length modeling method can be seen as a statistical variant of the minimum duration modeling where each HMM only accepts samples which are at least as long as the shortest samples

observed in the training data. In this method the length (number of states) of each HMM is set to a specified quantile of the corresponding character length histogram[14].

5.6 Bigram Language Model

Bigram language models represent a simple version of the more general statistical n-gram language models. Such models can help the Viterbi decoding step to favor more commonly found word sequences over less frequently observed ones. The text provided in this section will focus on the bigram case while statistical language modeling and n-gram language models are covered in Part IV.

5.6.1 Fundamentals

Bigram language models are based on the observation that we are often able to guess the next word when we are reading a given text and stop in the middle of a phrase. In cases were the choice is less clear we can at least narrow down the huge amount of known words to a relatively small number of candidates. This property of natural language suggests that the probability of a word is highly depending on the previous text.

In the case of the bigram language model the previous text is approximated by the single previous word[15]. This dependency is modeled by the conditional probability $p(w_i|w_{i-n})$ where w_i represents the word to guess and w_{i-1} stands for the previous word.

The probability $p(s)$ of a sentence $s = (s_1, \dots s_n)$ can then be decomposed as follows.

$$p(s) = \prod_{i=1}^{n} p(w_i|w_{i-1}) \tag{5.8}$$

[14]The character length histogram is computed by the estimation of the number of observations (feature vectors) for each corresponding sample in the training data.

[15]When the previous text is approximated by the last two previous words the term trigram language model ($n = 3$) is used. 4-gram language models are based on the last three words etc.

Table 5.1
Sample bigram probabilities for the word 'to'.

w_{i-1}	w_i	$p(w_i\|w_{i-1})$
to	the	0.009333
to	be	0.002239
to	a	0.000138
to	have	0.000105
to	make	0.000076
to	do	0.000068

For $i = 1$ either the unigram $p(w_1)$ or a special bigram $p(w_1| < s >)$ can be used where $< s >$ marks the beginning of a new sentence.

A sample set of bigrams with the corresponding conditional probabilities is given in Tab. 5.1 where the bigrams have been estimated from a part of the LOB corpus and are sorted with decreasing probabilities.

The conditional probabilities used for a bigram model are based on the relative frequencies $f(w_i|w_{i-1})$ of a word pair (w_{i-1}, w_i) observed in a large training corpus for a specified lexicon.

$$f(w_i|w_{i-1}) = \frac{N(w_{i-1}, w_i)}{N(w_{i-1})} \tag{5.9}$$

where $N(w_{i-1}, w_i)$ represents the number of times the word pair has been observed in the training corpus and $N(w_{i-1})$ corresponds to the count for the word w_{i-1}.

For a lexicon of 10,000 words the bigram language model will have 10^8 conditional probabilities to estimate. Therefore, not a single instance will be observed in the 1,000,000 word LOB corpus for most of the possible bigrams. In all sentences s where a zero count bigram would be included the application of Eq. 5.8 would lead to $p(s) = 0$ which is clearly not desirable. To solve the zero frequency problem a number of smoothing techniques[16] have been proposed. These techniques redistribute some of the probability mass from the frequently observed bigrams to the bigrams which have never been observed in the training corpus.

5.6.2 Bigrams and HMM Decoding

For the recognizer presented in this thesis a bigram language model has been chosen for the following practical reasons. First, the size

[16] A brief description a specific smoothing technique is provided in Sec. 13.3.

of the LOB corpus available for the training of the language model is not very large. Second, experimental results show a significant performance gain when using bigram language models [95, 143] but indicated that no major improvements of the recognition rates can be achieved using trigrams [143].

Thanks to the factorization of the sentence probability given in Eq. 5.10 the word transition probabilities $p(w_i|w_{i-1})$ provided by the bigram language model can efficiently be included in the Viterbi decoding process as follows

$$\phi(s_i) = \phi(s_{i-1}) + \log p(X_i|w_i) + \log p(w_i|w_{i-1}) \tag{5.10}$$

where $\phi(s_i)$ denotes the recognition score for the word sequence $s_i = (w_1, w_2, ...w_i)$ and $p(w_i|X_i)$ the likelihood for the feature vector sequence X_i to represent the word w_i. Please note that X_i in Eq. 5.10 represents not a single feature vector but a feature vector sequence representing word w_i.

5.7 Grammar Scale Factor and Word Insertion Penalty

Grammar Scale Factor and Word Insertion Penalty are two parameters used in the Viterbi decoding step. These parameters are needed to compensate for the fact that both the HMM and the bigram language model do not compute true probabilities. First, continuous density HMMs are producing likelihoods[17] as described in Section 5.4. Second, n-gram language models are only approximating sentence probabilities. As a consequence Eq. 5.10 can be rewritten in the following way.

$$\phi(s_i) = \phi(s_{i-1}) + \log p(X_i|w_i) + \alpha \ \log p(w_i|w_{i-1}) + \beta \tag{5.11}$$

where the parameter α will be called grammar scale factor throughout this thesis[18]. Parameter β is called word insertion penalty or

[17] A likelihood is a real value and does not necessarily lie in the range $[0, 1]$.

[18] In other works the terms linguistic weight, language weight or, more specifically, language model weight are used.

simply insertion penalty. Since the probabilistic meaning of these two parameters is unclear, the optimal values for α and β are normally determined by experiment on the validation set as suggested in [111].

Only few works investigate the role of these two parameters [9, 104, 127]. In [9] the estimated probabilities from the acoustic model and the language model are treated as outputs of two experts. The scores are then weighted with a reliability factor and a linear combination is used to merge the two values. A different approach is proposed in [104] where both parameters are shown to be dependent on the sentence length and the n-gram Bernoulli language modeling is investigated which allows for a better sentence length modeling.

5.7.1 Grammar Scale Factor

The grammar scale factor is used to weight the influence of the language model probabilities against the optical model probabilities provided by the HMM character models. In addition to the role of weighting the language model an additional influence should be considered. Since n-gram models assign lower probabilities to longer sentences[19] α should also depend on the length of a candidate sentence [104]. Dynamic adjustment of α (on the sentence level) has been investigated in [127] where the optimal grammar scale factor has been determined for each test sentence separately. However, no clear improvement over a fixed globally optimal grammar scale factor has been reported.

5.7.2 Word Insertion Penalty

An optimized word insertion penalty helps the Viterbi decoding to balance the word insertion rate and the word deletion rate. In [104] the use of negative word insertion penalties has been reported to compensate for word deletions induced by large grammar scale factors.

[19] The probability of a sentence of a given length will always be higher than the probability of the same sentence with an additional word appended.

5.8 Recognition Lattices and N-Best Lists

Recognition lattices and n-best lists correspond both to data structures which represent different alternative recognition results produced by the Viterbi decoding step. Recognition lattices contain the most promising subspace of results which has been investigated by the decoding step. N-best lists can be extracted from recognition lattices and contain the n best candidate sentences.

5.8.1 Recognition Lattices

The recognition lattices are produced by an alternative formulation of the Viterbi algorithm called the Token Passing Model [151]. Recognition lattices can be represented as weighted finite state machines where a node represents a segmentation boundary and a link corresponds to a word hypothesis between two segmentation boundaries. The word's score can then be assigned to the link in the form of a link weight.

Fig. 5.15 provides a small part of a recognition lattice produced by the sentence recognition system. The edges correspond to the word hypothesis with the corresponding recognition scores.

5.8.2 N-Best Lists

Given a recognition lattice in the form of a weighted finite state machine the n paths with the highest total score can then be found. These path correspond to the n-best recognition results and are called the n-best list.

An example for a n-best list is provided in Fig. 5.16. The first column contains the rank information where rank 1 corresponds to the best solution found by the recognizer. In the column 'Score' the recognition score of the corresponding candidate sentence is provided and the candidate sentence is given in the third column.

Please note that other n-best decoding algorithms exist [58, 80] which produce n-best lists more efficiently than explicitly generating a recognition lattice first.

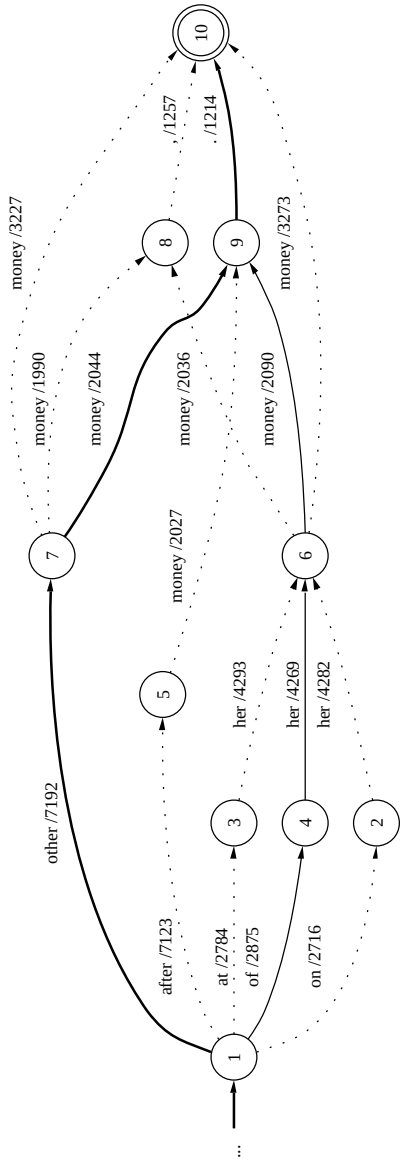

Figure 5.15
A simplified part of a recognition lattice. The bold path corresponds to the sentence with the highest score, the solid path represent the second best solution.

Figure 5.16
A n-best list showing the first five candidate sentences with the corresponding recognition scores.

Rank	Score	Candidate sentence
1	23,924	She has put up the value other money .
2	23,922	She has put up the value of her money .
3	23,890	She had put up the value other money .
4	23,888	She had put up the value of her money .
5	23,854	She has put up the value at her money .

5.9 Measuring System Performance

Measuring the performance of text recognition systems is more complex than in the case of the recognition of isolated words. Unfortunately performance measures provided in the are not always comparable and sometimes not defined at all.

In order to provide transparent results, all performance measures used in this thesis are motivated and properly introduced in the following subsections.

5.9.1 Sentence Recognition Rate

The sentence recognition rate is defined as the percentage of the correctly recognized sentences. A handwritten sentence image is considered to be correctly recognized when an exact matching between the recognized words and the transcription of the corresponding sentence image can be found. A 100% rate of this measure could only be reached if the recognition system would find the correct transcription for all handwritten sentence images.

5.9.2 Word Recognition Rate

The word recognition rate reflects the percentage of correctly recognized words given the transcription of the sentence image. A word recognition rate of 100% will only be reached if all words present in the transcription are correctly recognized.

For the domain of information retrieval systems the recognition rate is a suitable performance measure to estimate the system performance which can be reached for the automatic indexing of handwritten documents. The recognition rate may not be an appropriate measure in the context of a transcription system for handwritten documents. Since additional words are not taken into account, a 100% recognition rate does not necessarily mean that the recognition result is completely correct[20].

[20]The recognition result may contain additional words which are not present in the transcription of the handwritten text image.

Table 5.2 *Measurement of deletions (D) insertions (I) and substitutions (S) and corresponding word recognition rates and word level accuracies.*

Alignment	H	D	I	S	N	Rec.	Acc.
a b c a a b c a	4	0	0	0	4	100%	100%
a b c a a a c a	3	0	0	0	4	75%	75%
a b c a a c a	3	1	0	0	4	75%	75%
a b c a a b a c a	4	0	1	0	4	100%	75%
a b c a a b b a a	3	0	1	1	4	75%	50%

For the comparison of a sentence hypothesis found by the recognition system with the transcription of the sentence image a Dynamic Programming-based string alignment procedure is used. This procedure determines an optimal alignment between the word sequence provided by the recognition module and the word sequence provided by the transcription of the sentence. For the optimal alignment the number of correctly recognized words H, the number of deletions D, insertions I, and substitutions S are counted. Assuming that the transcription contains N words the word recognition rate is then defined as follows.

$$Word\ Recognition\ Rate = \frac{H}{N} \times 100\% \tag{5.12}$$

Examples of the string alignment procedure are shown in Tab. 5.2 where the alignment of transcription with the recognition result is shown in the first column. All examples assume the transcription 'a b c a' ($N = 4$) which is then compared to different hypothetical recognition results. The resulting word recognition rates and word level accuracies can be found in columns 'Rec.' and 'Acc.' respectively.

5.9.3 Word Level Accuracy

The word level accuracy is the complementary measure for the recognition rate. A 100% word level accuracy will only be reached if the recognition result matches the transcription word by word. Since insertions are taken into account, the word level accuracy

may be less than 100% even if all words of the transcription occur in the recognition result.

$$Word\ Level\ Accuracy = \frac{H - I}{N} \times 100\% \qquad (5.13)$$

Comparing Formula 5.12 and 5.13 it can be seen that the word recognition rate is an upper limit for the word level accuracy.

Experiments and Results

6

This chapter summarizes the various optimization steps which were involved to build the baseline system for the offline recognition of handwritten sentences.

6.1 Experimental Setup

This section introduces the main assumptions which have been made and describes the experimental setup used for the training, optimization and testing of the baseline recognition system.

6.1.1 Segmented Sentences Assumption

The first assumption which is made for the offline recognition of English text in is the *Segmented Sentences Assumption*. This signifies that it is taken for granted that the handwritten input is properly segmented into complete sentences. The sentences are defined by the sentence initial marks provided by the Tagged LOB corpus.

The assumption has been made when it became clear that a parsing of complete texts was computationally not feasible. Therefore, the input images are split into sentences which represent the next smaller building blocks of natural language. As a result the tasks introduced in Section 6.1.3 will be sentence recognition tasks.

6.1.2 Closed Vocabulary Assumption

The second assumption made is called the *Closed Vocabulary Assumption* which means that all words contained in the test sentences are known in advance. It is clear that for real world applications a more general approach will be required for the case of general text recognition but the closed vocabulary assumption provides a more controlled environment to measure the system performance. No empirical trade-off between the size of the vocabulary and the expected percentage of out of vocabulary cases has to be made in order to optimize the system performance.

6.1.3 Recognition Tasks

For the optimization of the number of training iterations, the grammar scale factor and the word insertion penalty two tasks have been defined. In the multi-writer task (MW) an environment is simulated where handwriting samples are available for most of the writers. For the second task a writer independent (WI) situation is assumed. This means that the recognition system will not have seen any of the the writing styles of the test set during the training or the parameter optimization (validation) steps.

For both tasks training, validation, and test sets have been defined. The handwritten text images for the training set have been selected to support both the MW and the WI task at the same time. Included are 5,799 images of handwritten text lines containing a total of 39,993 words where the text lines have been written by 448 different writers.

Table 6.1 *The definition of the different validation sets.*

	MW		WI	
Quantity	Small	Large	Small	Large
Sentences		200		200
Text lines	100	534	100	582
Words	746	3,814	826	4,094
Lexicon size	8,820	8,824	8,817	8,827
Writers	100	157	100	100

The specification of the validation sets are provided in Tab. 6.1. The small validation sets are used for the optimization of the number of iterations needed for the training of the HMM. For the optimization

of the grammar scale factor and the word insertion penalty the large validation sets are used. The lexicon size corresponds to the number of word surface forms in the lexicon, where the lexicon has been compiled using the following steps. First, using the closed vocabulary assumption, all (tagged) words from the test set are first added to the initially empty lexicon. Second, based on the tagged LOB corpus the most frequent words are added until the lexicon contains 10,000 tagged words. Finally, the tags are stripped from the words leading to a smaller number of lexicon entries since a given word surface form may occur with more than one tag[1].

Quantity	MW	WI
Sentences	200	200
Text lines	573	575
Words	3,933	3,956
Lexicon size	8,822	8,821
Writers	156	100

Table 6.2 *The definition of the test sets.*

For the test sets the definition is given in Tab. 6.2. In the case of the MW task 142 of the writers are common with the validation set and 147 are also present in the training set. The 100 writers of the WI task are different from both the 100 writers of the validation set and the 488 writers represented in the training set.

Please note that the optimization of the character model lengths and the optimization of the number of Gaussians have been carried out on separate tasks which are briefly described in the corresponding sections.

6.2 Optimizing the Character Model Lengths

For the optimization of the character specific model lengths a word recognizer was used very similar to the sentence recognizer described in this part of the thesis. The only difference was the modeling of the states where single Gaussian instead of multi-Gaussians were used.

[1]The word 'order' can be tagged as a noun and as verb. Since different verb forms are identified separately by the tagset verbs like 'put' will occur in the base form, in the present and in the past tense.

For the experiments an isolated word recognition task based on word images from the segmented version of the IAM database has been defined. Training and testing has been carried out on 10,929 word images. For the test set 1,000 words have been randomly chosen, which left 9,929 words for the training set. The lexicon contained the 2,296 words covering the transcriptions of the 10,929 word images.

Each of the presented length modeling scheme described in Sec. 5.5 can be optimized using different settings of a single parameter. The resulting number of model states and the corresponding recognition rates for each method are summarized in Tab. 6.3, 6.4, and 6.5.

Table 6.3 *Word recognition rates using a fixed number of states.*

States	Size	Recognition rate
8	584	57.2%
10	730	58.9%
12	876	59.7%
14	1022	60.7%
16	1168	61.0%
18	1314	58.6%
20	1460	56.7%
22	1606	53.7%
24	1752	50.9%

Tab. 6.3 documents the resulting recognition rates for different (fixed) number of states. The Bakis length modeling method using different fractions of average character lengths (column 'Fraction') is provided in Tab. 6.4, and the recognition rates for the quantile length modeling method are shown in Tab. 6.5 using the indicated quantiles (column 'Quant.'). The total number of states of each recognition system is reported in the column 'Size'.

To explain the significant improvement of both the Bakis and the quantile length modeling over the fixed length modeling approach, some additional experiments have been carried out.

For the comparison of the different length modeling approaches some properties of the fixed length modeling (using 16 states) and the quantile modeling (using the 2% quantile) have been selected. After training of the HMMs, the models for the characters 'i' and 'm' have been used to generate length histograms for the corresponding characters[2] where the x-axis correspond to the character width in pixels and the y-axis to the number of samples.

Fraction	Size	Recognition rate
0.20	702	60.5%
0.30	1049	67.3%
0.36	1258	68.3%
0.38	1331	69.0%
0.40	1398	69.2%
0.42	1465	68.8%
0.44	1538	68.6%
0.50	1752	66.0%
0.60	2099	59.8%

Table 6.4
Word recognition rates for the Bakis method.

Quantile	Size	Recognition rate
0.00	1207	56.6%
0.01	1323	65.3%
0.02	1462	69.0%
0.03	1568	69.1%
0.04	1656	69.6%
0.05	1721	68.8%
0.10	2014	65.5%
0.20	2366	57.8%
0.50	3262	29.6%

Table 6.5
Word recognition rates for the quantile method.

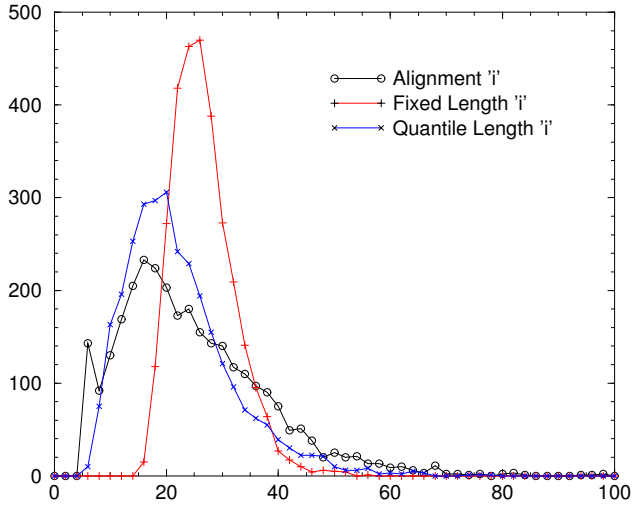

Figure 6.1
Estimated and generated character lengths for the character 'i'.

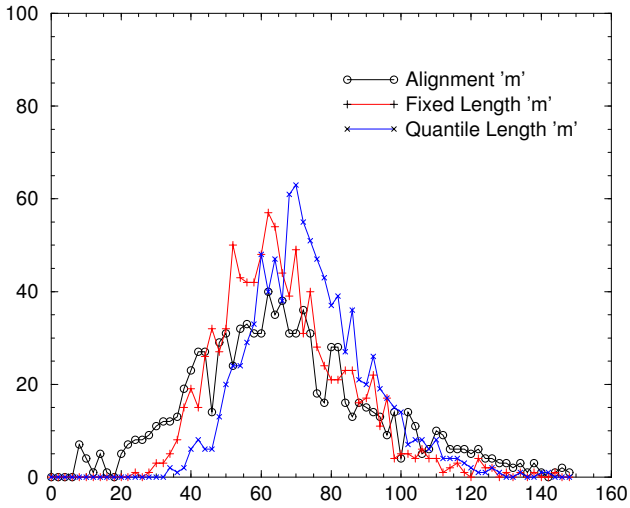

Figure 6.2
Estimated and generated character lengths for the character 'm'.

In Fig. 6.1 and Fig. 6.2 the resulting length distributions for the fixed length modeling and the quantile modeling method are compared with the 'true' length histograms obtained by forced alignment. It can be observed that for short characters (represented by 'i') the fixed length modeling approach produces length distributions which do not adequately model the estimated length histograms. In case of the character 'm' both the fixed length and the quantile length modeling produce similar length distributions with a peak which is roughly at the position of the peak of the estimated length histograms.

Based on this observation two additional experiments were carried out to verify the importance of the correct modeling of short characters. The Bakis and the quantile length modeling scheme were combined with the fixed length modeling in the following way. After the length calculation of each character using either the Bakis or the quantile method, characters with more than 16 states (the optimal length obtained using the fixed length modeling scheme) have been truncated to a maximal length of 16 states.

Both the truncated Bakis and the truncated quantile methods did not only produce higher word recognition rates than the system us-

[2]This has been done using the stochastic Monte Carlo method to simulate runs through the finite state automata defined by the HMM transition probabilities.

Fixed	Bakis	Quantile	T. Bakis	T. Quantile
16	0.4	0.04	16/0.4	16/0.04
1168	1398	1656	1088	1092
61.0%	69.2%	69.6%	68.1%	67.9%

Table 6.6 *Comparison of the different length modeling schemes.*

ing fixed length modeling but also resulted in significantly smaller HMM systems. This is shown in Tab. 6.6 which summarizes the achieved word recognition rates (third row) of the different length modeling schemes. The results for the truncated Bakis method are reported in column 'T. Bakis' and the results for the truncated quantile method in column 'T. Quantile'. In the first row the corresponding parameter settings are provided, the second row lists the number of resulting model states and the last row the corresponding recognition rates.

Based on the results in Tab. 6.6 the truncated Bakis method has been chosen for the handwriting sentence recognition system implemented in this thesis.

6.3 Optimizing the Number of Gaussians

The number of Gaussians have been optimized on a text line recognition task using 4,313 lines of handwritten text for the training set, and 100 lines for the validation set. The lexicon contained 10,000 entries.

For the training of the multi-Gaussian system the following training scheme has been used[3].

1. A single Gaussian system is trained using truncated Bakis length modeling and four training iterations
2. The number of Gaussians are increased by one and the new system has been retrained using further training iterations
3. The System performance is evaluated every two training iterations and the best system is kept for further increasing the number of Gaussians

[3]See [44] for a comparison of different training schemes.

4. Until a predefined number of Gaussians has been reached continue with Step 2

5. Return the best performing HMM as the result

Table 6.7 *System performance for different number of Gaussians.*

Gaussians	Line Rate	Word Rate	Word Accuracy	Iter.
1	0.0%	20.6%	-41.0%	4
2	1.0%	30.9%	1.6%	14
3	7.0%	40.6%	24.6%	16
4	12.0%	53.9%	46.7%	12
5	13.0%	55.1%	46.6%	16
6	10.0%	55.8%	47.5%	18
7	11.0%	58.8%	51.0%	14
8	19.0%	60.9%	53.1%	18

Tab. 6.7 provides the the performance for the best system for the number of Gaussians tested. The text line recognition rate is given in column 'Line Rate', the word recognition rates and word level accuracies can be found in columns 'Word Rate' and 'Word Accuracy' respectively. Column 'Iter.' contains the number of additional training iterations. A huge performance gain over the single Gaussian can be observed for all performance measures. The best performing system used a mixture of eight Gaussians trained with a cumulative number of 112 iterations. It can be further observed that all performance measures improve monotonically with increasing number of Gaussians. Unfortunately time constraints did not allow for a further increase of the number of Gaussians.

As a result, the handwriting recognition system implemented in this thesis also uses a mixture of eight Gaussians to model the emission probabilities.

6.4 Optimizing the Number of Training Iterations

For the optimization of the number of training iterations for both the multi-writer and the writer independent task, the small validation sets containing 100 lines of handwritten text has been used.

The resulting system performance has then been evaluated every 10 iterations. In contrast to the experiment described in the previous section the single Gaussian system has been directly transformed to a system with eight Gaussians. As reported in [44] no large difference could be observed between the two alternative approaches of the training of multi Gaussian HMM. Therefore, the conceptually simpler training scheme has been adopted for this optimization step.

Figure 6.3 presents the measured text line recognition rates[4] for both the small multi-writer and writer independent validation sets. For the writer independent task the best generalization rate is reached after 40 iterations while the performance for the multi-writer task is further increasing up to 140 iterations. Due to the limited time and resources available the optimization of the number of iteration steps has been halted at 150 iterations. Figure 6.4 shows the plots for the measured word level accuracy of the two tasks.

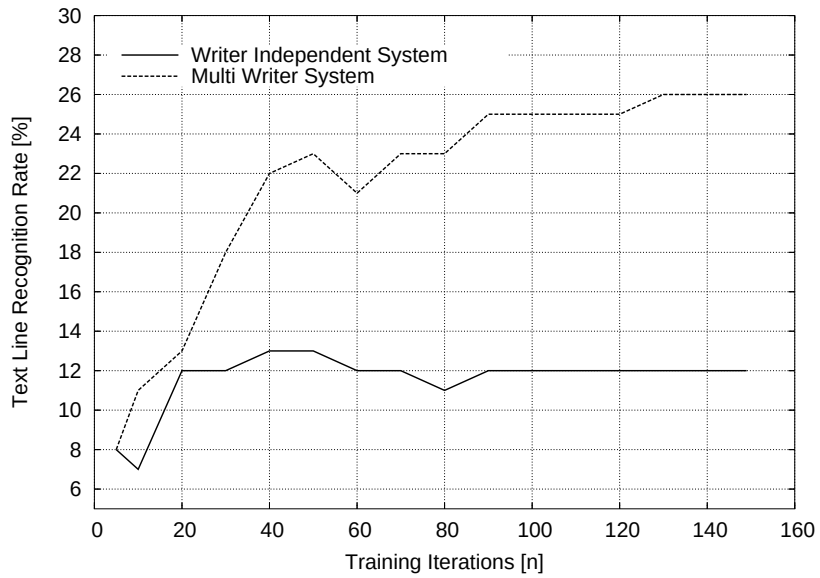

Figure 6.3
Text line recognition rate for different number of training iterations, small validation set.

Based on these results, it has been decided to use 140 training iterations for the multi-writer system and 90 iterations for the writer independent system. The 90 iterations for the writer independent system are motivated in the following way. From the plot it is obvious that there should be 40 or more iterations. To prevent the

[4]The text line recognition rate is defined analogously to the sentence recognition rate introduce in Section 5.9.

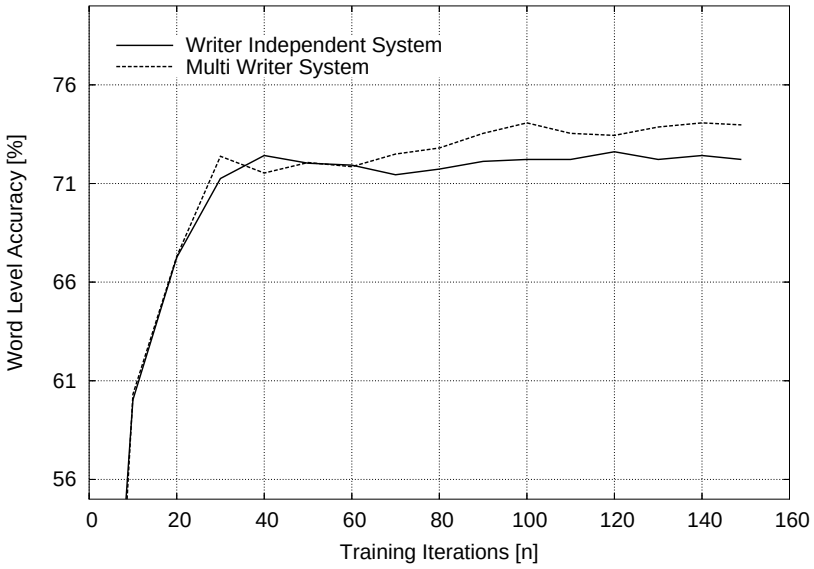

Figure 6.4 *Word level accuracy for different number of training iterations, small validation set.*

system from over-fitting the training data, 140 iterations seemed not be necessary.

6.5 Optimizing Grammar Scale Factor and Word Insertion Penalty

For the optimization of the grammar scale factor α and the word insertion penalty β, a global optimization scheme has been applied. For each combination of grammar scale factor and word insertion penalty indicated by Fig 6.5 the system performance has been evaluated.

The system performance for the different grammar scale factors and word insertion penalties have been measured for the case of the writer independent task. Fig. 6.5 provides the results for the sentence recognition rate, and Fig. 6.6 shows the corresponding word level accuracies. As stated in Section 5.7, it can be seen that the two parameters are not independent from each other. The higher the grammar scale factor becomes, the higher the word insertion penalty has to be chosen for optimal results. It can further be observed that the optimal values for the two parameters depend on

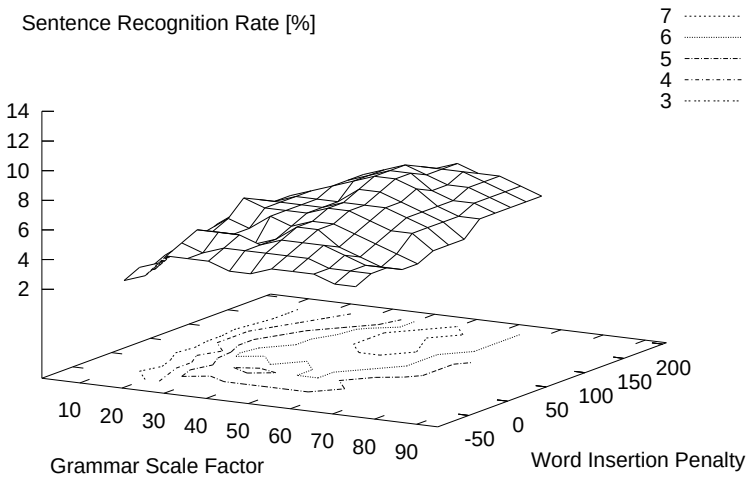

Figure 6.5
Sentence recognition rate for different values of α and β, large validation set.

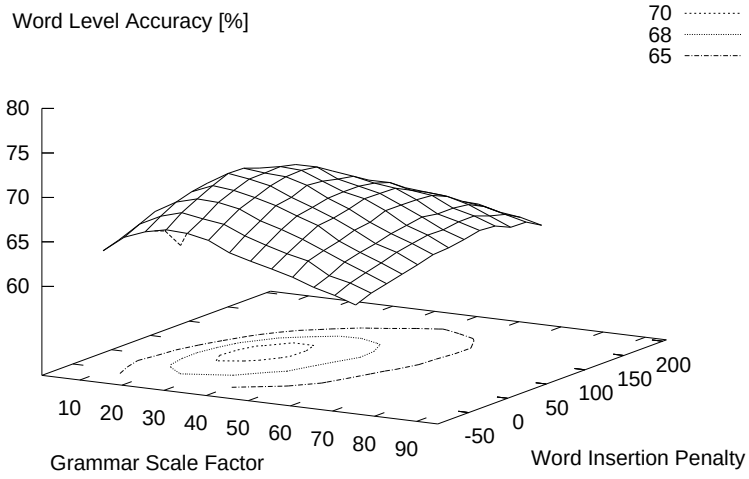

Figure 6.6
Word level accuracy for different values of α and β, large validation set.

the performance measure to be optimized. Parameter values for optimizing the sentence recognition rate ($\alpha = 45,\ \beta = 75$) differ significantly from the parameters which maximize the word level accuracy ($\alpha = 30,\ \beta = 50$).

In order to verify the optimization of the number of training iterations presented in the previous section, six additional experiments have been carried out using the large validation sets. For both the multi-writer and the writer independent systems the grammar scale

factor and the word insertion penalty were optimized after 40, 90 and 140 training iterations. The measured performances are summarized in Tab. 6.8 for the multi-writer system and in Tab. 6.9 for the writer independent system.

Table 6.8
Performance of the multi-writer system after different numbers of training iterations for the large validation set.

Training Iterations	40	90	140
Sentence Recognition Rate	7.0%	9.0%	8.5%
Word Recognition Rate	73.7%	74.9%	74.4%
Word Level Accuracy	71.6%	71.5%	72.2%

Table 6.9
Performance of the writer independent system after different numbers of training iterations for the large validation set.

Training Iterations	40	90	140
Sentence Recognition Rate	6.5%	6.5%	6.5%
Word Recognition Rate	74.2%	74.1%	74.3%
Word Level Accuracy	71.4%	71.2%	71.4%

The difference between the results measured for the small and for the large validation set could be explained by the fact that no optimization of the grammar scale factor and the word insertion penalty has been made.

Table 6.10
Comparison of the baseline and the optimized systems.

	α, β	$1, 0$	$30, 50$	Improvement
MW	Sen.	1.0%	8.5%	7.5%
	Wrd.	61.2%	74.4%	13.2%
	Acc.	53.7%	72.2%	18.5%
WI	Sen.	0.0%	6.5%	6.5%
	Wrd.	58.9%	74.1%	15.2%
	Acc.	49.4%	71.2%	21.8%

Compared to a baseline system where no optimization of the grammar scale factor and the word insertion penalty have been made ($\alpha = 1.0$ and $\beta = 0.0$) a significant improvement was achieved by the joint optimization of these two parameters. The summary of the comparison is provided in Tab. 6.10 where the improvements are shown for both the multi-writer task (MW) and the writer independent task (WI). For each task the sentence recognition rate

(Sen.), the word recognition rate (Wrd.) and the word level accuracy (Acc.) are provided in the corresponding lines.

6.6 Validation Set Results

This section evaluates the performance of the fully optimized sentence recognition system. Tab. 6.11 summarizes the system configuration after all optimization experiments for the two different tasks. The n-best analysis for the sentence recognition rates, the word recognitions rate and the word level accuracy are provided in Fig. 6.7, Fig. 6.8, and Fig. 6.9 respectively.

Parameter	MW	WI
Number of states	0.4/16	0.4/16
Gaussian mixtures	8	8
Training iterations	140	90
Grammar scale factor	30	30
Word insertion penalty	50	50

Table 6.11 *Task specific configuration of the sentence recognition system.*

Considering the differences between the parameter settings for the two tasks provided in Tab. 6.11 it can be observed that the chosen settings differ in the number of training iterations only.

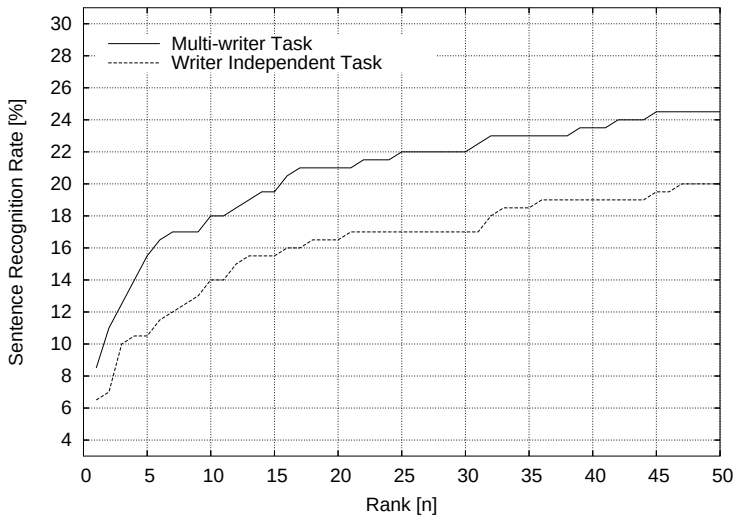

Figure 6.7 *Sentence recognition rate for the large validation set.*

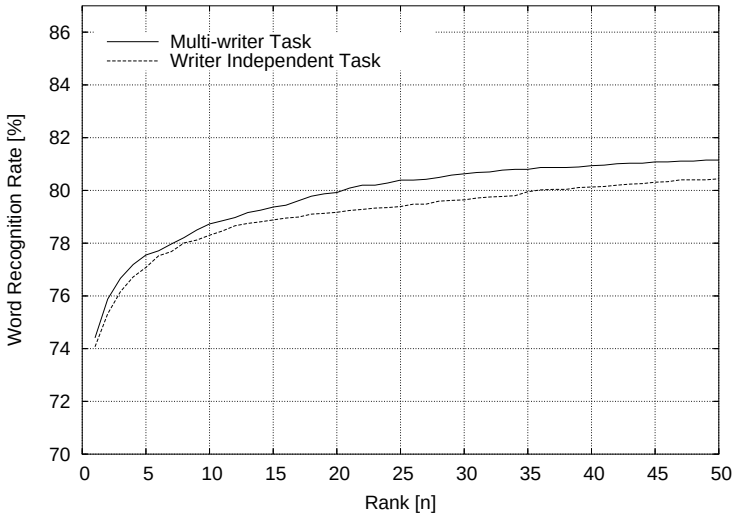

Figure 6.8
Word recognition rate for the large validation set.

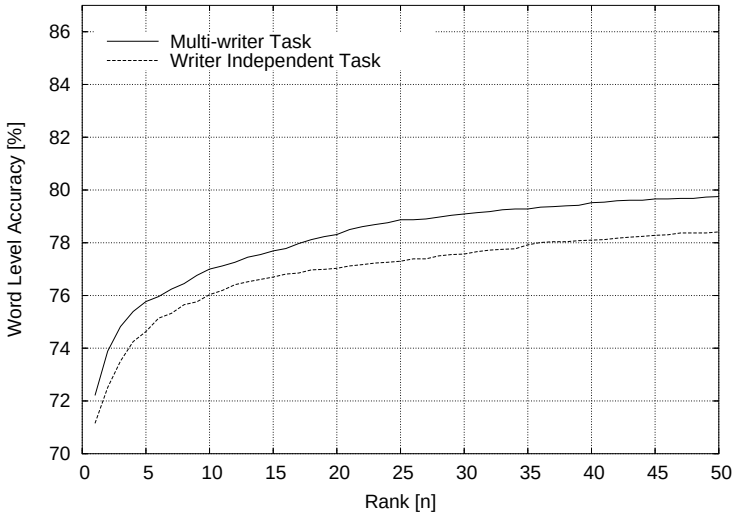

Figure 6.9
Word level accuracy for the large validation set.

6.7 Test Set Results

In this section the performance of the carefully optimized handwritten sentence recognizer[5] is evaluated on the test sets for both the multi-writer and the writer independent task. The n-best analysis for the sentence recognition rate is provided in Fig. 6.10.

For the the word recognition rate the n-best analysis is given in Fig. 6.11, and the corresponding plot for the word level accuracy is

[5]Identical configuration and parameter settings have been used as for the performance evaluation on the validation set.

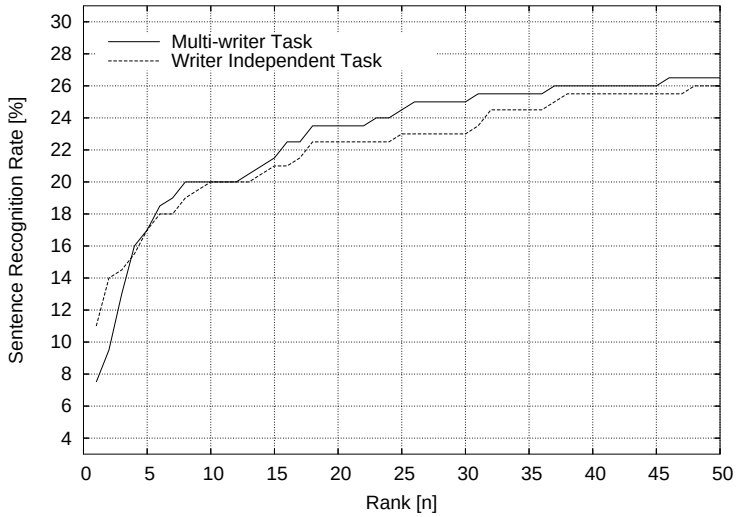

Figure 6.10
Test set sentence recognition rate.

shown in Fig. 6.12.

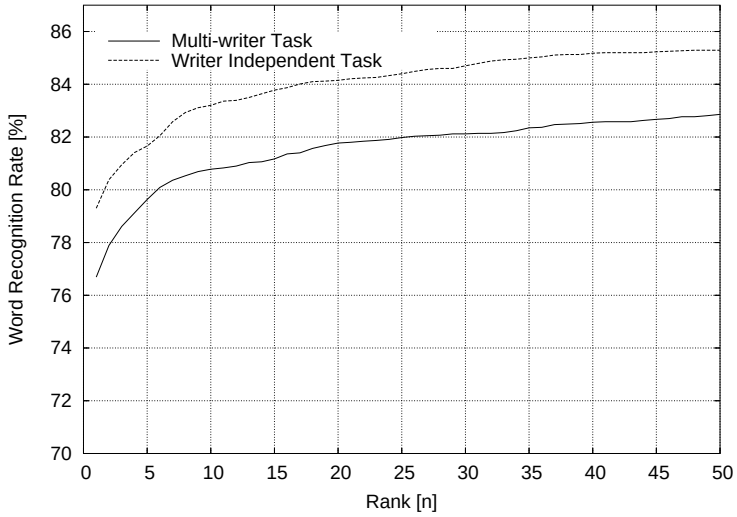

Figure 6.11
Test set word recognition rate.

As discussed in Section 5.9, the word recognition rate is always higher than the word level accuracy for a given system. The intuitive expectations that a writer independent task should always be harder than a multi-writer task is not met by the results. Possible explanations lie in the random selection of the test sets and in the decision to include a large number of writers in the multi-writer task. If the latter hypothesis is assumed, it can be concluded that a multi-writer task for a large number of writers is as hard as a writer independent task. This hypothesis is supported by the ob-

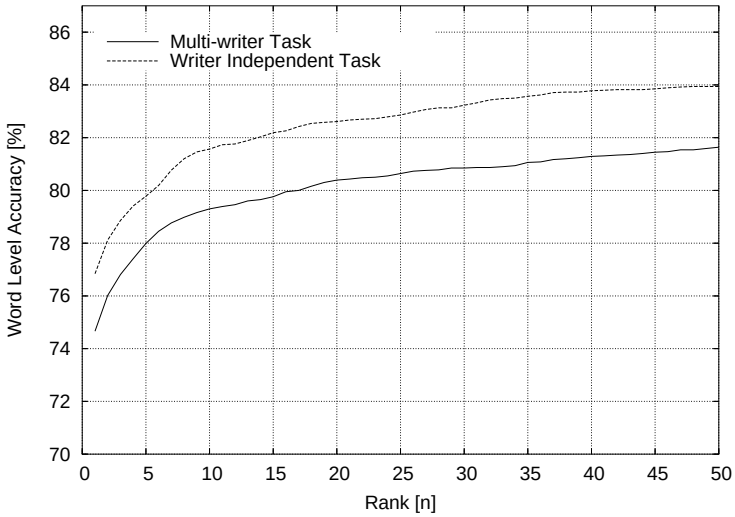

Figure 6.12
Test set word level accuracy.

servation that the optimization of the parameters does not seem to depend on the task. For both the multi-writer task and the sentence recognition task the same parameter settings have found to be optimal. With the exception of the number of training iterations which can be judged to be arbitrary considering the results reported in Tables 6.8 and 6.9.

An additional counter-intuitive observation can be made by comparing the results for the large validation sets and the test sets. The evaluated performance of all measures is always better on the test set compared to the validation set. This result shows that the performance of two recognition systems should only be compared if the results have been produced using exactly the same experimental setup. It must be concluded that it is not enough to train and test systems using material from the same database. It must further be ensured that the evaluation of the performance has been made using identical test sets.

7 Conclusions

An offline recognition system for cursively handwritten sentences has been implemented and carefully optimized as a part of this thesis. The system based on [95] uses continuous density HMM and a bigram language model to perform segmentation free recognition of English sentences.

The following sections address the different areas where the original system [95] has been enhanced or improved. Section 7.1 addresses the data preparation and the changes of the preprocessing. Improvements of the modeling of the characters are discussed in Section 7.2 and the optimization of the decoding step is reviewed in Section 7.3.

7.1 Data Preparation and Preprocessing

The segmentation ground truth provided by the IAM database has been used to automatically extract complete handwritten sentences. The extracted sentences were then used to define two different sentence recognition tasks. The first task simulates a multi-writer environment. The second task is writer independent which means that disjunct sets of writers were used to define the training, the validation and the test set.

In the preprocessing steps different slant and skew normalization techniques have been implemented. The new normalization proved to be robust for the large variety of writing styles and writing instruments represented in the IAM database. For the slant normalization technique the method proposed by [102] has been used and for skew normalization and reference line detection a new method based on connected components has been applied.

7.2 Character Modeling

In the domain of Hidden Markov Modeling two problems were specifically investigated. First, the optimization of the number of states for the linear model topology has been addressed by a comparison of different strategies to determine the optimal number of states per character model [154]. Second, the number of Gaussian mixtures has been increased to improve the overall recognition rate.

For the character HMM three different length modeling schemes to optimize the number of states in left-to-right HMMs have been implemented. The investigated methods included the frequently used fixed length modeling and the Bakis length modeling schemes. Also a new method, based on the quantiles of character length histograms, has been proposed. Applying the original recognition system described in [95] to a isolated word recognition task a performance improvement of more than 8% has been observed for both the Bakis modeling and the proposed quantile based modeling scheme over the previously used fixed length modeling approach.

The performance increase gained by the introduction of multi Gaussian models to estimate emission probabilities has been measured on a text line based recognition task. The text line recognition rate could be improved from 0% to 19%, the word recognition rate from 21% to 61%, and the word accuracy level from -41% to 53% respectively.

7.3 Decoding

A full optimization of the grammar scale factor and the word insertion penalty has been carried out for the first time in the domain

of handwriting recognition systems. Average (absolute) improvements of 7% for the sentence recognition rate, 14% for the word recognition rate, and 20% for the word level accuracy have been achieved.

Instead of providing only the most likely hypothesis, large recognition lattices were produced. From the lattices the n-best sentence candidates have then been extracted.

III N-Best Sentence Candidate Parsing

Introduction

8

The sequential coupling of a handwriting recognition system and a probabilistic parser for the evaluation of the n-best sentences candidate is investigated. A bottom up chart parser as described in [15] is used to analyze the n-best candidate sentences produced by the handwritten sentence recognition system. For each sentence candidate the parser will compute the most probable parse and its corresponding probability. Using the combination scheme proposed in this thesis, the parse probability is merged with the score from the HMM based recognizer into a sentence score. Based on this new sentence score, the n-best list can then be reordered to take the grammatical quality of the hypotheses into account.

In the next section of this introduction parsing of natural languages will briefly be discussed and stochastic context-free grammars will be introduced in Sec. 8.2. The state of the art in combining syntax analysis with both optical character recognition and speech recognition systems is reviewed in Sec. 8.3. Finally, a system overview is given in Sec. 8.4 describing the different components of the combination scheme.

The next chapters of this part are organized as follows. The bottom-up chart parser and the combination scheme are described in Chapter 9. Experiments and results are provided in Chapter 10 and conclusions are drawn in Chapter 11.

8.1 Parsing Natural Languages

The parsing of sentences allows to discover the grammatical structures of the input sequence of words [1]. Possible outcomes of parsing are a single parse, a (possibly large) set of grammatically correct but ambiguous parses or no solution at all. In the latter case, the input sentence is considered to be grammatically incorrect. In the case where a single solution is found, it is unambiguous and the grammatical structure explaining the sentence is considered to be grammatically correct[2]. Typically, a large number of grammatically correct structures are found in the case of natural languages.

8.1.1 Why Parse?

Parsing of natural languages can be useful for several reasons. It allows to detect grammatical errors in sentences for which no parse can be found. If a parse can be found, the resulting structure of the parse tree carries useful grammatical information which can support a semantic analysis of the sentence. The result of the semantic analysis can then be used in applications e.g. machine translation, or natural language based interfaces for information retrieval systems.

The sequential coupling of a handwritten sentence recognizer and a parsing step used in this thesis is motivated by the fact that the recognizer can provide a large number of possible solutions. Often, the recognizer's top choice is grammatically unlikely or even incorrect. The parsing step can help to identify such cases and assign higher priorities to candidate sentences which are grammatically sound.

8.1.2 Grammatical Structures and Parse Trees

The term *Grammatical Structure* will be used for the hierarchical organization of grammatical units of a sentence which are called

[1] For the purpose of this thesis the sequence of words will always represent a sentence. However, the sequence of words may also represent a phrase or a complete text depending on the scope of the parsing process.

[2] This is typically true for formal languages which are often used to specify the syntax of programming languages or communication protocols.

constituents. Constituents consist of a sequence of words which may stand for a noun phrase, a verb phrase, etc.

Parse trees are a commonly used representation of grammatical structure which are closely related to parsing. In fact, parse trees are the solutions found by the parsing process. They can be represented as shown in Fig. 8.1 where word tags and constituent tags are based on the LPC [38].

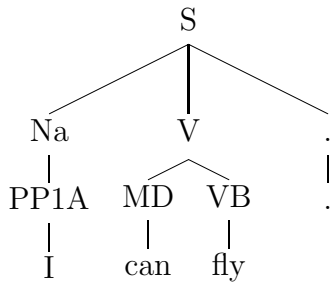

Figure 8.1
Parse tree for the sentence I can fly.

For a more compact representation of parse trees the bracketed notation is typically used. The corresponding bracketed version of the parse tree shown in Fig. 8.1 is given below.

$[_S\ [_{Na}\ [_{PP1A}\ \text{I}\ ]][_V\ [_{MD}\ \text{can}\ ][_{VB}\ \text{fly}\ ]][.\ .\ ]]$

Bracketed sentences are used by major treebanks like the LPC as introduced in Sec. 3.3 or the Penn treebank [91].

8.1.3 Structural Ambiguities

A sentence is structurally ambiguous if more than one parse tree can be found by the parsing process. Since many natural languages sentences are structurally ambiguous, *disambiguation* is a key issue. The goal of disambiguation is to select the correct parse out of all possible parses which can be found for a given sentence. One strategy for disambiguation is based on probabilistic parsing using stochastic grammars. But in many cases additional knowledge is required for disambiguation, as can be seen in the examples below.

In [63] the attachment ambiguity, the coordination ambiguity and the noun phrase bracketing are mentioned as particularly common kinds of structural ambiguities. As an illustration examples found in [63] for the attachment ambiguity and the coordination ambiguity are provided below.

One morning I shot an elephant in my pyjamas.
How he got into my pyjamas I don't know.
Grucho Marx, *Animal Crackers*, 1930

The first example represents an attachment ambiguity where it is syntactically not clear if the phrase *in my pyjamas* should be attached to the elephant or the person. The ambiguity could be solved on the semantic level for this sentence. However, the second sentence shows the wrong assumption made for the disambiguation of the first sentence.

Coordination ambiguity often occurs when different phrases are cojoined using a word like *and*.

old men and women

In the above phrase it is syntactically and semantically ambiguous if the meaning is [[Old men] and [women]] or if it should be read as [Old [men and women]].

A different type of ambiguities is the lexical ambiguity which can be resolved in most cases by parsing of the input sentence.

I can fly.

Although *can* and *fly* in the above sentence can both be tagged as either a verb or a noun, only a single grammatically correct structure will be found, tagging both words as verbs.

8.2 Stochastic Context-Free Grammars

Stochastic context-free grammars (SCFG) represent the probabilistic version of the well known context-free grammars. By assigning probabilities to productions, parse tree probabilities can be computed. This supports the idea of disambiguation in the following way. If several parse trees can be found for a single input sentence the most probable parse is returned by the parser.

SCFG can also be used as a statistical language model. Based on parse tree probabilities it is possible to assign a probability value

to a given input sentence which can be used as a measure for the syntactical quality of the sentence. The probability of a sentence is normally defined as the sum of the probabilities of all possible parse trees. Possible approximations include the probability of the most probable parse or the sum of the probabilities of the n most probable parses.

In the following subsections context-free grammars will be defined informally and the assignment of probabilities to productions required for SCFG is addressed. In both subsections the focus will lie on the application of natural language parsing.

8.2.1 Context-Free Grammars

Context-free grammars (CFG) represent a commonly used formalism by linguists to describe the hierarchical aspects of the grammatical structure of natural languages[3].

In order to specify a CFG the set of words[4] occurring in the language under consideration has to be defined. The words are called terminal symbols. In the case of natural language parsing two distinct sets of nonterminal symbols are used in most cases. First, the tagset containing the grammatical word tags[5] like noun, adjective, etc. has to be specified. The elements representing the word tags are also called preterminal symbols. The second set of nonterminal symbols represent the constituent tags which define grammatical units like noun phrases, verb phrases, etc. Unfortunately many different tagsets for word tags and constituent tags exist today. Typically, each linguistic resource is defining its own set of word tags and constituent tags[6].

The rules which describe the possible expansions of nonterminal symbol are called productions. Finally, a single nonterminal symbol needs to be marked as the start symbol. This symbol represents the top-level structure which can be identified by the grammar. In

[3]Context-free grammars are also used to specify the syntax of formal languages as in the case of programming languages.

[4]For the application of handwriting sentence recognition, capitalization and spelling of words is relevant, *The* and *the* are considered to be two distinct words.

[5]For word tags the terms Part-Of-Speech or POS tags are also used in literature.

[6]In the case of the LPC 167 word tags and 250 constituent tags are defined.

the case of natural language grammars the start symbol represents an independent sentence.

S	→	NP VP :.
NP	→	:Det :NN
VP	→	:VBD NP
:Det	→	the
:NN	→	mouse
:NN	→	cat
:VBD	→	ate
:.	→	.

Figure 8.2 *Productions of a context-free grammar for a subset of English sentences.*

Fig. 8.2 provides the productions of a toy example for a context-free grammar. Each production contains exactly one nonterminal symbol on the left hand side of the arrow and a sequence of nonterminal or terminal symbols on the right side of the arrow. The ':' prefix is used to mark the word tags (preterminal symbols). The remaining nonterminal symbols represent constituent tags. The arrow indicates that the symbol on the left hand side can be rewritten as the sequence on the right hand side (thus the alternative term *rewrite rules* for productions). The set of terminal symbols of the toy grammar is given by the words 'the', 'mouse', 'cat', 'ate', and '.' leaving the remaining symbols as the set of nonterminal symbols. In order to complete the definition of this context-free grammar the letter 'S' is used as the start symbol.

When a CFG is used in generating mode, the start symbol can be rewritten by the application of any production with a corresponding left hand side. On the resulting sequence of symbols productions can be iteratively applied until no more nonterminal symbols can be found in the result. Using the toy grammar of Fig. 8.2 the start symbol 'S' can be rewritten in the following way to generate a sentence.

S ⇒ NP VP :. ⇒ :Det :NN VP :. ⇒ the :NN VP :. ⇒ the cat VP :.
⇒ the cat :VBD NP :. ⇒ the cat ate NP :. ⇒ the cat ate :Det :NN :.
⇒ the cat ate the :NN :. ⇒ the cat ate the mouse :.
⇒ the cat ate the mouse .

Please note that the order of the application of the productions is irrelevant. For clarity always the left most nonterminal has been chosen to be rewritten in the above example. The term 'left derivation' is sometimes used to describe this particular order of application of the productions.

8.2.2 Specification of the Productions

For both CFG and SCFG the number and the structure of the productions need to be specified. A large amount of so called *Grammar Inference* techniques has been proposed in the literature to learn the structure of a grammar from examples. In this text only two cases for the learning of the structure of a CFG will be mentioned.

If a CFG is to be learned in the Chomsky Normal Form (CNF)[7], all possible productions are usually enumerated using all terminal and nonterminal symbols available in the grammar [85]. Although this enumeration of productions is a very simple way to specify the structure of a CFG, it does not take into account the structural information provided by treebanks.

If a treebank is available, the structure of the grammar can directly be extracted from the parsed sentences. Such treebank grammars [17] exploit the structural information present in the treebank.

8.2.3 Probabilization of Productions

Probabilization of productions is the technique of assigning probabilities to productions which augment CFG into SCFG. There are two main techniques of assigning probabilities to productions depending on the availability of a treebank.

When a treebank is available the probabilities of the treebank grammar can be estimated from the relative frequencies of the productions contained in the bracketed sentences [17, 85]. This is done by counting the number of times a production has been used in a derivation of a sentence and then normalizing by the number of times the production's left hand side has been observed in the treebank. The treebank SCFG can also be used as an initial grammar. More compact grammars can then be derived from such treebank grammars by discarding productions which have been observed only once [17, 85]. Furthermore, a reestimation scheme for the production probabilities of treebank grammars is presented in [85] which is based on the Viterbi score [101].

[7] A CFG grammar is said to be in CNF if all its productions are either of the form A $\rightarrow$BC or the form A $\rightarrow\alpha$ where A, B anc C are nonterminal symbols an α is a terminal symbol of the grammar.

If no treebank is available, the following technique is often applied. As a large number of alternative parses are found for many natural language sentences separate counts are used for the different alternative parses. These counts can then be weighted with the probability of the corresponding parse tree. The Inside-Outside algorithm (IO) [2] represents the standard algorithm to estimate production probabilities in this case. Unfortunately, the IO algorithm can be computationally inhibitive for large CFGs for natural languages because of both the high time complexity per iteration and the large number of iterations necessary to converge[8].

SCFG with probabilities estimated by one of the techniques mentioned above have been shown to be automatically consistent [21]. This means that such a SCFG induces a proper probability distribution over all sentences which can be derived from it[9].

8.3 State of the Art

In the field of offline handwriting recognition the tendency can be observed to address problems of increasing complexity. High recognition rates have been published for the recognition of isolated digits [48] or characters [135]. The achieved performance for numeral string recognition [87] or isolated word recognition [78] is already significantly lower. If the task complexity increases further, as in the case of the recognition of handwritten addresses [89] or bank checks [41], task specific knowledge like the relation between zip code and city name, or between courtesy amount and legal amount becomes essential.

For general text recognition task specific information can be found in the linguistic domain. So far, the successful application of n-gram language models supporting the recognition of handwritten text lines has been reported [95, 143]. Unfortunately the power of n-gram language models is limited by the fact that they are unable to capture long distance relationships between words. For $n > 3$ it is difficult to accumulate sufficient amounts of text for an

[8] See [1] for a comparison of the IO algorithm with other techniques to learn the probabilities of a SCFG.

[9] See Sec. 9.1.4 for the defininition of the sentence probability based on a SCFG.

accurate estimation of the probability for less frequently observed word sequences.

A number of different approaches to further increase the recognition rate of existing systems in the domains of both speech and optical character recognition were proposed in the last decade. Especially in the domain of speech recognition, different approaches of incorporating syntactical knowledge in the recognition process have been investigated. A summary of the more promising techniques as well as the description of a lattice chart parser for speech recognition can be found in [16].

The following two subsections provide an overview of some systems presented in the literature which incorporate different syntax analysis schemes to improve recognition results.

8.3.1 Syntax Analysis and Optical Character Recognition

In the domain of Optical Character Recognition (OCR) only few publications are available which investigate the use of syntax analysis in any form. The use of linear grammars is described in [53, 124]. Sets of valid syntactic patterns were used in [25] and a word lattice rescoring mechanism has been proposed in [73]. A CFG to improve word recognition rates has been used in [51].

In [53] a stochastic linear grammar in the form of an HMM is used. A subset of the word tags found in the Brown corpus are represented by the model states and words occur in the form of emitting symbols. Using a simulated output of a machine print OCR for different levels of noise, a set of hypothesis for each word (called word neighborhood) in the test sentence was produced. Based on the n-best paths through the states of the HMM the size of the word neighborhoods was then reduced for a given input sentence. Depending on the noise level, word error rate reductions between 27% and 43% have been reported. In the field of online handwritten sentence recognition the use of a stochastic linear grammar has been reported in initial experiments [124].

Valid syntactic patterns and a hierarchical pattern matching parser are used in [25] to reduce word neighborhoods. The syntactic patterns have been automatically extracted from the Brown corpus.

Experiments based on the texts provided by the Brown corpus led to a word neighborhood reduction of 24%.

In [73] a statistical syntactic analyzer based on a concept similar to class n-grams is described. Bigram and trigram of grammatical word classes are combined with a so called grammatical frequency factor and a lexical probability factor. The combined scores are then used to rescore a synthesized word lattice. Results indicate an improvement in the rank of the correct words over the other words.

The use of an enhanced CFG has been investigated in [51] where a probabilistic lattice chart parser based on the Cocke-Kasami-Younger algorithm[10] was used to compute the most probable parse tree for a word lattice. The CFG containing 706 productions was written manually and included confidence scores to indicate the priority of a production. As in the case of [25, 53] the output of a simulated machine print OCR was used to produce the test sentences describing word neighborhoods. For experiments on over 6,000 words of text an improvement of the word recognition rate from 13% to nearly 80% has been reported.

8.3.2 Syntax Analysis and Speech Recognition

The use of syntactical knowledge in the form of grammars has mostly been applied in the context of speech recognition systems. In earlier works as [75, 76] context-free grammars (CFG) were used. The use of a unification based CFG[11] was reported in [50]. The use of a different structural language model is proposed in [18]. More recently, the tendency can be observed to use SCFG [36, 64, 113] or a unification based SCFG [16].

In [75] a generalized LR parser [134] is used to predict the next phone according to two different grammars for Japanese phrase structure. The general CFG contains 1,461 productions and 1,035 words while the task specific CFG has been reduced to 582 productions and a lexicon of 275 words. The achieved phrase recognition rate was 72% for the general CFG, and 80% for the task specific CFG respectively.

[10]The CYK parsing algorithm has been developed independently by Cocke, Younger [152] and Kasami.

[11]See [63] for an introduction to unification parsing.

Essentially the same parsing technique has been incorporated into the SPHINX speech recognition system [76]. A LR parser is used to truncate word transitions during the decoding step which would lead to ungrammatical word sequences. The CFG used for the SPHINX system has been derived from 900 templates provided with the Resource Management Task. A comparison of the combined system and the baseline system has been made using 150 sentences from the Resource Management Task. It was found that the word accuracy was improved from 94.1% to 95.8% while the sentence accuracy could be increased from 69.3% to 70%.

Three different interaction strategies between the acoustic decoder and a parsing module were investigated in the context of the VERBMOBIL project [50]. Bottom-up parsing of the word-lattice provided by the decoder and top-down prediction of the next word by a LR parser. Finally, the parser for the verification of the hypothesized words has been used by the decoder during its search step. The unification grammar contained 47 productions and 363 word form entries. Using ten test utterances the different strategies were compared against the acoustic decoder alone. The performance of the bottom-up and the prediction strategies were measured at 70% utterance recognition rate compared to the verification strategy and the decoder alone for which a 50% recognition rate for utterances were measured.

In [64] a SCFG was used as a language model in the Berkeley Restaurant Project speech recognition system. The SCFG based on a manually written CFG containing 1,389 productions and included many task-specific nonterminal symbols. Different approaches to exploit the information contained in the SCFG were investigated. The SCFG was first used to smooth a bigram language model, then it was integrated in the decoding step using a probabilistic version of an Early parser [130] to prune the decoders search space. Finally, two combinations of the smoothed bigram and the integrated parsing were proposed. Using a subset of 364 sentences from the training corpus the reduction in the word error rate was measured. The original word error rate of 34.6% could be reduced to 29.6% by any of the investigated strategies. A slightly lower word error rate of 28.8% could be observed for the combination of a smoothed bigram model and the probabilistic parsing.

Applications of hybrid language models defined by a linear combination of a word based trigram model and a broad coverage SCFG

are described in [36, 113]. In [36] the SCFG in Chomsky Normal Form contained all possible productions with 35 nonterminal symbols and 45 word tag (terminal) symbols provided by the Penn treebank [91]. Production probabilities were then estimated on the word tag sequences from the Penn treebank as described in [1]. On the DARPA '93 HUB1 test setup the word error rate could be reduced from 16.6% to 16.0% if the performance of the combined language model is compared to the trigram model alone.

A SCFG combined with a conditional probability model which conditions production probabilities based on the context within which they appear is proposed in [113]. Structure and parameters of the grammar were estimated using the Penn treebank. On the DARPA '93 HUB1 test setup a reduction of the word error rate from 16.5% to 15.1% has been reported. Both [36] and [113] used the Penn treebank (containing 1M words) to train the trigram model for their experiments.

For the structural language model proposed in [18]. the word error rate could be reduced from 13.7% to 13.0% when using the DARPA '93 HUB1 test setup. In this case, the trigram model provided with the corpus has been used. This language model has been trained on 40M words.

Lattice parsing for speech recognition is proposed in [16] where a sequential coupling of a speech recognizer and a probabilistic bottom-up chart parser is described. The SCFG containing 28,253 productions and 6,648 nonterminal symbols has automatically been compiled from a unification grammar containing 92 productions and 25 nonterminal symbols relevant for a phone-book inquiry system. Using ten test sentences, the performance of the combined system and the speech recognizer alone have been compared. Coupling with the SCFG strictly improved the results in 50% of the sentences. In 80% of the sentences the result of the combined system has been considered to be at least as good as the speech recognizer alone.

8.4 System Overview

The proposed combination scheme of the sentence recognizer and the syntax analysis module is illustrated in Fig. 8.3. As demonstrated by this figure, a sequential coupling has been chosen to combine the two techniques.

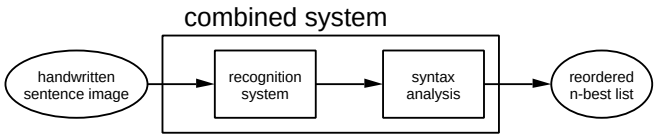

Figure 8.3 *The coupling of handwriting recognition and syntax analysis.*

Advantages of a sequential coupling lie in a clean design and a better runtime performance since only a fraction of the search space investigated by the recognizer needs to be checked by the parser. A potential disadvantage of the sequential coupling lies in premature decisions made by the handwritten sentence recognizer. If the correct solution is not contained in the n-best list provided by the recognizer, the syntax analysis module has no possibility to recover.

Since the handwriting recognition module has been described in detail in Part II only the syntax analysis module is presented in the next subsection.

8.4.1 Syntax Analysis

The syntax analysis module consists of two steps. First, the n-best list is parsed using the probabilistic parser. Then the n-best list is reordered in the combiner according to the calculated parse probabilities and the sentences scores provided in the n-best list.

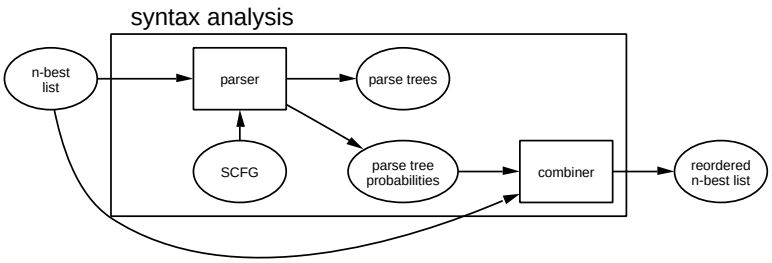

Figure 8.4 *The two step syntax analysis module.*

Fig. 8.4 provides a graphical representation of the syntax analysis module. Since the goal of the syntax analysis is to provide an improved n-best list only, the parse trees generated by the probabilistic parser are kept as an internal result. The extraction of the SCFG used by the parser is explained in the next subsection.

8.4.2 Grammar Construction

For the construction of the SCFG a number of resources are involved as shown in Fig. 8.5.

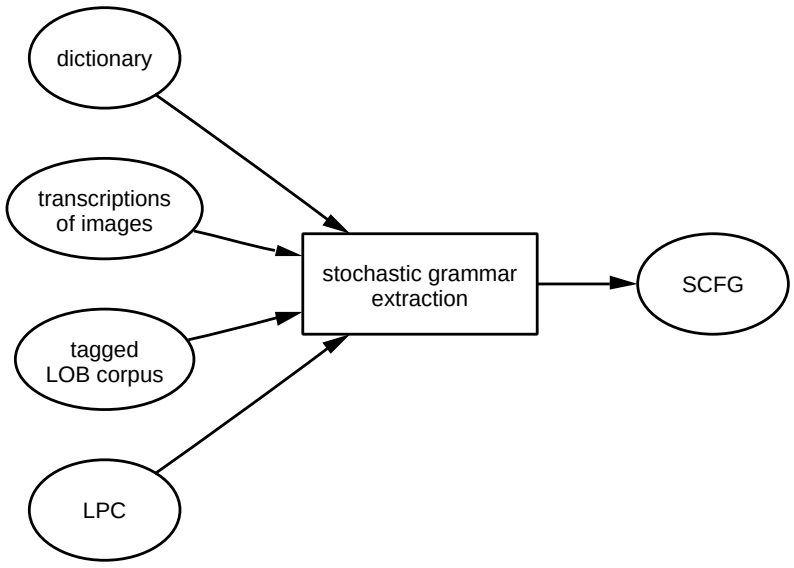

Figure 8.5 *The extraction of the stochastic context-free grammar.*

Using the transcriptions of all (test) sentence images the corresponding sentences from the tagged LOB corpus and the LPC are removed. Based on the reduced versions of the tagged LOB corpus and the LPC a SCFG is constructed which does not include any sentences from the test set. The set of terminal symbols (words) is defined by the dictionary. The set of preterminal symbols (word tags) and nonterminal symbols (constituent tags) are extracted from the reduced tagged LOB corpus and the reduced LPC. Productions are extracted from the reduced LPC. Finally, the production probabilities are estimated using the reduced versions of both the tagged LOB corpus and the LPC.

Methodology

9

This chapter covers the techniques used for the syntax analysis module as well as the combination scheme for the handwritten sentence recognizer and the syntax analysis module.

First, context-free grammars and stochastic context-free grammars are formally defined in the next section. Then the extraction of the SCFG used for the experiments is explained in Sec. 9.2. The bottom-up chart parser used for the parsing of candidate sentences is introduced in Sec. 9.3. The proposed combination scheme used to merge the parse probabilities with the recognition scores provided by the baseline recognizer is presented in Sec. 9.4.

9.1 Notations and Definitions

9.1.1 Context-Free Grammars

A Context-Free Grammar (CFG) G is a four-tuple (N, T, P, S) where N represent the set of nonterminal symbols and T the set of terminal symbols where $N \cap T = \emptyset$. The set of productions is denoted by P and $S \in N$ is used as the start symbol. All productions in P can be written as $A \rightarrow \alpha$ where $A \in N$ and $\alpha \in (N \cup T)^+$. Productions of the form $A \rightarrow a$ with $a \in T$ will be called lexicalized productions. Non-lexicalized productions must not have terminal symbols on their right hand side and can be formalized with $A \rightarrow \alpha$ where $\alpha \in N^+$.

9.1.2 Parse Trees and Left-Derivations

Parse trees are a graphical representation of the derivation of the start symbol S into the sentence $s = (w_1, w_2, \dots w_n)$ which can be written as $S \stackrel{*}{\Rightarrow} s$. A derivation $d(s) = (P_1, P_2, \dots P_m)$ of a string s is specified by the sequence of productions $P_1, P_2, \dots P_m$ applied such that $S \stackrel{P_1}{\Rightarrow} \alpha_1 \stackrel{P_2}{\Rightarrow} \alpha_2 \dots \stackrel{P_m}{\Rightarrow} s$.

The left-derivation $d_l(s) = (P_1, P_2, \dots P_m)$ is then defined as a derivation where P_i is always rewriting the left-most nonterminal symbol of α_i. Please note that exactly one left-derivation exists for every parse tree.

In the bracketed notation a parse tree will look like $[_S \ \dots \ ]$and the application of a production $A \rightarrow B_1 B_2 \dots B_n$ will produce the following bracketed structure

$[_A \ [_{B_1} \ \dots \ ] \ [_{B_2} \ \dots \ ] \dots \ [_{B_n} \ \dots \ ] \ ]$

The one to one correspondence of parse trees and bracketed sentences facilitates the production of left-derivations out of bracketed sentences.

9.1.3 Stochastic Context-Free Grammars

Stochastic Context-Free Grammars (SCFG) can be defined as the pair (G, p) where G is a CFG and p a probability function over the productions $A \rightarrow \alpha$ where $p(A \rightarrow \alpha) \rightarrow (0, 1]$.

The probabilities of all productions $A \rightarrow \alpha_i \in P$ have to be normalized in the following way

$$\sum_{A \rightarrow \alpha_i} p(A \rightarrow \alpha_i) = 1 \tag{9.1}$$

A further requirement for the production probabilities has to be satisfied if the SCFG would be used as a language model. In that case it is required that the probabilities of all possible sentences which can be derived from the start symbol S sum up to one, i.e. that the SCFG is consistent. Please note that in the context of this thesis, the production probabilities of the SCFG are computed using a relative frequency estimation which has been shown to produce consistent SCFG automatically [21].

9.1.4 Probabilities of Parse Trees and Sentences

The probability of a parse tree $p(d_l(s))$ for input sentence s is based on the corresponding left-derivation $d_l(s) = (P_1, P_2, \ldots P_m)$ as follows

$$p(d_l(s)) = \prod_{i=1}^{m} p(P_i) \tag{9.2}$$

This probability corresponds also to the sentence probability $p(s)$ if only a single parse tree can be found for sentence s. If the sentence s is structurally ambiguous several parse trees exist which can be represented by the set of k left-derivations $d_{l_1}, d_{l_2} \ldots d_{l_k}$. In this case the sentences probability is defined as the sum of the different parse tree probabilities.

$$p(s) = \sum_{i=1}^{k} p(d_{l_i}(s)) \tag{9.3}$$

The probability of the most probable parse $\hat{p}(s)$ is then given by $\hat{p}(s) = max_i p(d_{l_i}(s))$ and the most probable parse can be represented by $\hat{d}_l(s) = argmax_s p(d_{l_i}(s))$.

9.2 Construction of the SCFG

The availability of the LPC treebank allowed a straightforward extraction of the productions and production probabilities using the provided bracketed sentences. The construction of a SCFG has been achieved through the following steps.

First, the set of tagged words is defined using the same method as for the extraction of the bigram language model as described in Sec. 4.3.4. From the tagged words (A, α) the lexicalized productions of the from $A \rightarrow \alpha$ can directly be derived where A represents the word tag and α the word itself. The probabilities $p(A \rightarrow \alpha)$ of the lexicalized productions are estimated using a reduced version of the tagged LOB corpus[1] in the following way.

[1] As in the case of the extraction of the bigram language model the resources linked with the test set must be excluded to estimate the model parameters

$$p(A \rightarrow \alpha) = \frac{N(A, \alpha)}{N(A)} \tag{9.4}$$

where the number of occurrences of the tagged word (A, α) is measured by $N(A, \alpha)$ and $N(A)$ represents the number of times the word tag A has been observed in the reduced version of the tagged LOB corpus.

For the extraction of the non-lexicalized productions the bracketed sentences of the reduced LPC corpus are transformed into left-derivations and finally into the corresponding sequence of non-lexicalized productions by skipping the transformation of word tags into the words. The production probabilities can then be estimated using the following equation

$$p(A \rightarrow \alpha) = \frac{N(A \rightarrow \alpha)}{\sum_{\beta} N(A \rightarrow \beta)} \tag{9.5}$$

where $N(A \rightarrow \alpha)$ represents the number of times production $A \rightarrow \alpha$ has been observed in the sequence of non-lexicalized productions mentioned above.

Please note that no further attempts as proposed in [17, 85] were made to modify this treebank grammar.

9.3 Bottom-up Chart Parsing

For the parsing of the sentence candidates provided by the handwritten sentence recognition system a bottom up chart parser for SCFG as described in [15] has been used. The parser can be seen as a generalized version of the CYK parser [152] and will be called CYK+ in this text. Principally, any probabilistic parsing algorithm could be used to calculate the probability of a sentence or the most probable parse[2].

The following subsection defines the class of accepted grammars. Then, an overview for the CYK+ algorithm is given, followed by a more detailed explanation in the remaining subsections.

[2]E.g. a probabilistic version of the top-down Early parser proposed in [128] could also be used.

9.3.1 Accepted Grammars

The grammar class accepted by CYK+ is the subset of the non-partially lexicalized productions. This signifies that no production must mix terminal and nonterminal symbols on its right hand side. In contrast to the standard CYK algorithm the productions need not to be in the Chomsky Normal Form[3] since the algorithm is performing binarization of the rules dynamically.

For the purpose of natural language parsing the above requirements are directly met by the fact that word tags and constituent tags are kept separate in most treebanks. Therefore, productions extracted from a treebank will already have the required format.

For the description of the algorithm below it is assumed that the grammar does not contain epsilon productions or cycles[4]. While cycles will not prevent the algorithm to find the most probable parse, they have to be taken into account when the n most probable parses have to be determined.

9.3.2 Algorithm Overview

The basic structure used by the CYK+ parsing algorithm is a triangular chart which consists of $n(n+1)/2$ cells for an input sentence $s = (w_1, w_2, \ldots w_n)$ of length n. Each cell consists of two lists, the open list and the closed list which maintain sets of partial parses. In the open list elements representing partially parsed right hand sides of productions are stored. A closed list contains completely parsed sub-strings represented by a corresponding nonterminal symbol. In a first step, called initialization step, the first row of the chart is filled as described in the following subsection. When the initialization step has been completed, the parsing step is filling the remaining cells of the chart. During the parsing step the most probable parse can be computed as described in the last subsection[5].

[3] A CFG in Chomsky Normal Form only contains productions of the form $A \rightarrow B\ C$ or $A \rightarrow a$, where A, B, C represent nonterminal symbols and a stands for a terminal symbol.

[4] As an example consider a CFG containing productions $A \rightarrow B$ and $B \rightarrow A$. Epsilon productions rewrite a nonterminal as the empty word.

[5] The CYK+ algorithm is capable to efficiently compute the n most probable parses. See [15] for the detailed description.

9.3.3 Initialization Step and Self-filling

In the initialization step the cells $T_{1,i}$ in the first row of the chart are filled. For each word w_i of the sentence to parse all lexicalized productions of the form $X \rightarrow w_i$ are searched in the grammar. For each such production the nonterminal symbol X is then added to the closed list of the corresponding cell.

After the filling of the closed lists the self-filling procedure is applied to each cell $T_{1,i}$. This procedure updates the open list and takes care of productions of the form $A \rightarrow B$ where A and B represent nonterminal symbols. For each element B in the closed list the self-filling step is searching all productions of the form $A \rightarrow B\ \gamma$. When γ is the empty string, A is added to the closed list otherwise the element $B\bullet$ is added to the open list of cell T_1, i. Every time an element is added to the closed list, the self-filling procedure has to be applied recursively on the newly added element.

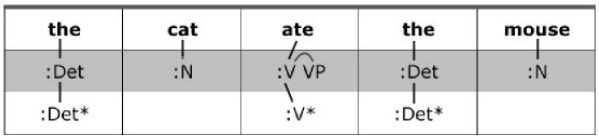

Figure 9.1
The initialization step of the CYK+ algorithm.

Fig. 9.1 provides an example of the result of the initialization step for the sentence *the cat ate the mouse .* where the available productions have been taken from Fig. 8.2. For each cell of the parse chart the closed list has a gray background and the elements of the open list are written on white background.

9.3.4 Parsing Step

After the completed initialization of the chart the parse step is filling the remaining cells word by word from the left to the right. The parsing step for word w_j is filling the cells $T_{2,j}$, $T_{3,j-1}$ $\ldots T_{j,1}$ along the diagonal. For the cell $T_{i,j}$ and all $k \in (1, \ldots i-1)$ possible combinations of elements $\alpha\bullet$ of the open list of cell $T_{k,j}$ and elements B of the closed list of cell $T_{i-k,j+k}$ are explored. For each production $A \rightarrow \alpha B \gamma$ found in the grammar, the open or the closed list of cell $T_{i,j}$ is updated. If γ is the empty string, the symbol A is added to the cell's closed list and the self-filling step described above is applied. In the case where γ is not empty, the element $\alpha B\bullet$ can be added to the open list of cell $T_{i,j}$.

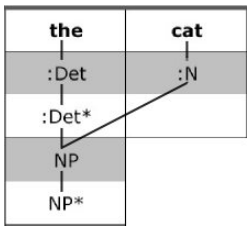

Figure 9.2
The CYK+ chart after parsing the first two words.

Fig. 9.2 provides the resulting chart after parsing cell $T_{2,1}$ where the element : $Det\bullet$ of the open list of cell $T_{1,1}$ could be combined with the word tag : N from the closed list of cell $T_{1,2}$ using production $NP \rightarrow : Det: N$.

The parsing step for the sentence $s = (w_1, w_2, \ldots w_n)$ is completed when cell $T_{n,1}$ has been filled. If and only if the start symbol S can be found in the closed list of cell $T_{n,1}$ the sentence s has been successfully parsed.

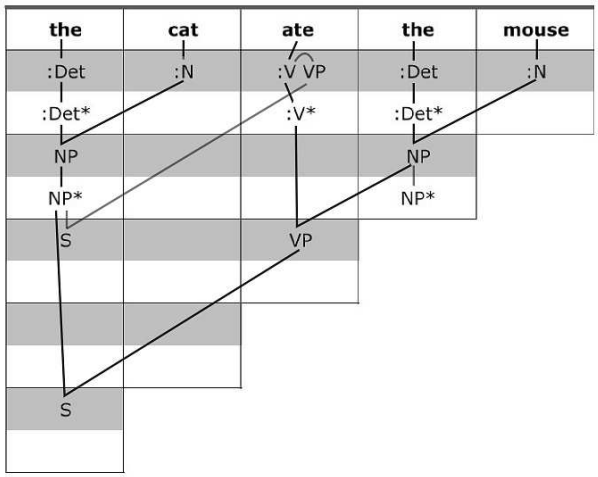

Figure 9.3
The CYK+ chart after completion of the parsing step.

Fig. 9.3 shows the result after the successful completion of the parsing step.

9.3.5 The Most Probable Parse

The computation of the most probable parse is straightforward since the probability of a parse tree has been defined as the product of the probabilities of all involved productions. Every time an element $\alpha B\bullet$ is added to the open list of chart cell $T_{i,j}$ the corresponding probability can be computed as the product of the

probability assigned to the element $\alpha\bullet$ of the open list of chart cell $T_{k,j}$ and the probability of the symbol B found in the closed list of cell $T_{i-k,j+k}$. If element $\alpha B\bullet$ already exists in the open list of cell $T_{i,j}$ the maximum of the existing probability and the newly calculated probability is assigned to the element. When the symbol A is added to the closed list of cell $T_{i,j}$ the product of the probability of production $A \rightarrow \alpha B$, the probabilities of the open list element $\alpha\bullet$ as well as the closed list symbol B are assigned to the symbol A. As in the case of the open list, it has to be checked if the symbol A already exists in the closed list of cell $T_{i,j}$. If A has already been added to the closed list the maximum of the two probabilities is assigned to the symbol.

9.4 Combination Scheme

The proposed combination of the recognition score with the parse probability into a sentence score is handled analogously to the incorporation of the transition probabilities from the bigram language model during Viterbi decoding. In contrast to the transition probabilities (which are merged with the recognition scores on the word level) the parse probabilities are merged on the sentence level.

9.4.1 Recognition Scores and Parse Probabilities

The recognition score $\phi(s)$ is calculated by the Viterbi decoding step including the bigram probabilities as described in Sec. 5.6.

The grammatical quality of a sentence candidate s is ideally represented by the sentence probability $p_G(s)$ calculated by a probabilistic parser using a given SCFG G. Since the calculation of all possible parse trees for natural language sentences can become prohibitive the probability of the most probable parse $\hat{p}_G(s)$ as defined in Sec. 9.1.4 has been used to approximate the sentence probability.

Tab. 9.1 provides examples for parse probabilities calculated by the CYK+ parser. The list of sentences candidates has been computed by the recognizer as shown in the previous part of the thesis.

Rank	Parse Prob.	Candidate sentence
1	7.69052e-23	She has put up the value other money .
2	4.62861e-20	She has put up the value of her money .
3	2.63105e-22	She had put up the value other money .
4	1.58352e-19	She had put up the value of her money .
5	1.12458e-21	She has put up the value at her money .

Table 9.1 *Parse probabilities for a n-best list showing the top five candidate sentences.*

9.4.2 Combining the Two

The sentence score $\psi(s)$ for a sentence $s = (w_1, w_2, ...w_n)$ is defined in the following way

$$\psi(s) = \phi(s) + \gamma \ \log p_G(s) \tag{9.6}$$

where $\phi(s)$ represents the recognition score from the handwriting recognition module and $p_G(s)$ the probability of the sentence using the SCFG G produced by the probabilistic parser. Parameter γ will be called the Parse Scale Factor. It weights the influence of the parse probability on the final sentence score. In the case where no parse could be found a fixed minimum parse probability is used to penalize grammatically incorrect sentence candidates. For $\gamma = 0$, the sentence probability provided by the parser will not affect the sentence score at all. If $\gamma > 0$, the sentence scores are influenced by the parse probabilities and a reordering of the n-best sentences may take place.

Rank	Sent. Score	Candidate sentence
1	23,703	She has put up the value other money .
2	23,729	She has put up the value of her money .
3	23,671	She had put up the value other money .
4	23,700	She had put up the value of her money .
5	23,650	She has put up the value at her money .

Table 9.2 *Sentence scores for a n-best list showing the top five candidate sentences.*

Tab. 9.1 provides the result of the combination of the recognition scores and the parse probabilities. A parse scale factor $\gamma = 10$ was used. This value has been optimized on the validation set as described in the next chapter. After the reordering of the n-best list, candidate sentence number two will be the new top candidate. This choice matches the transcription of the handwritten image exactly.

Experiments and Results

10

The experiments to optimize the combination scheme and the obtained results for both the validation and the test sets are documented in this chapter.

The first section of this chapter explains the experimental setup. In Sec. 10.2 the ability of the SCFG to cope with new sentences and to detect grammatically incorrect sentences is briefly investigated then the optimization of the parse scale factor is covered in Sec. 10.3. The results for the validation sets and the test sets are provided in the Sec. 10.4 and Sec. 10.5, respectively.

10.1 Experimental Setup

All experiments have been carried out using the multi-writer and writer independent tasks introduced in Sec. 6.1.3. The n-best lists have been obtained using the baseline recognizer described in Part II of this thesis. The SCFG has been extracted from the Lancaster Parsed Corpus as specified in Sec. 9.2.

Tab. 10.1 summarizes the configuration of the SCFG produced for the multi-writer (MW) and the writer independent (WI) recognition tasks.

Table 10.1 *Configuration of the SCFG used for the multi-writer and the writer independent experiments.*

Measure	MW	WI
Non-lexical Productions	11,716	11,694
Lexical Productions	10,000	10,000
Constituent Tags	249	248
Word Tags	163	161
Words	8,820	8,819

10.2 The Quality of the SCFG

The main goal of the syntax analysis module was to penalize grammatically incorrect solutions proposed by the sentence recognition system based on a stochastic context-free grammar. In order to estimate the quality of the SCFG extracted from the LPC two sets of experiments have been carried out[1].

10.2.1 Generalization Rate

In the first set of experiments the generalization capabilities of SCFGs extracted from the LPC were investigated. First, a SCFG was extracted from 90% of the available bracketed sentences. Using the remaining 10% of the sentences the coverage rate[2] was measured.

Applying ten-fold cross-validation an average coverage rate of 98.1% was observed for the test sets. Based on this result, a high enough generalization rate for the use of the SCFG in the context of the handwriting recognition experiments could be expected.

10.2.2 Detection of Incorrect Sentences

To measure the capability of the SCFG to recognize ungrammatical input the words in the test sentences were put into a random order to simulate grammatically incorrect sentences in a second set

[1] To approximate the experimental conditions of the handwriting recognition experiments a closed vocabulary was assumed by including all words found in the test set.

[2] The coverage rate measures the fraction of sentences for which at least one parse can be found using the specified grammar.

of experiments. Using the same cross-validation scheme as mentioned above a coverage rate of 74% has been measured for the 'grammatically incorrect' sentences.

Assuming that a sentence consisting of words put in random order should be considered to be grammatically incorrect in most cases, it must be concluded that the SCFG has also a high overgeneralization rate which will result in a limited capability to filter out ungrammatical sentence candidates.

A further disadvantage of the SCFG lies in the nature of its context-free productions which do not enforce linguistic agreements across constituent boundaries.

$[_S\ [_{Na}\ [_{PP3A}$ She $]][_V\ [_{HVZ}$ has $][_{VBN}$ put $]][_R\ [_{RP}$ up $]]\dots]$
$[_S\ [_{Na}\ [_{PP3A}$ She $]][_V\ [_{HV}$ have $][_{VBN}$ put $]][_R\ [_{RP}$ up $]]\dots]$

Figure 10.1 *Grammatically correct and incorrect sentences.*

As an example Fig. 10.1 provides parse trees in the form of bracketed sentences for grammatically correct and a grammatically incorrect sentences. Although the number of the subject is encoded in the word tag *PP3A* the information is not passed to the governing constituent tag *Na*. The same principle applies for the verbs 'has' and 'have' where the number is not passed to the V constituent. Therefore, the linguistically required agreement for the number of the subject and the predicate cannot be enforced by the SCFG extracted from the LPC. Even worse, the parse tree probability for the grammatically incorrect sentence 'She have put up ...' will be higher since production $V \rightarrow HV\ VBN$ has been observed more frequently than $V \rightarrow HVZ\ VBN$ in the LPC.

10.3 Optimizing the Parse Scale Factor

The proposed combination scheme of recognition scores provided by the n-best lists and the parse probabilities requires the optimization of the parse scale factor γ introduced in Sec. 9.4. The goal of the optimization was to achieve a maximum sentence recognition rate.

10.3.1 Tuning the Sentence Recognizer

As can be seen from the optimization experiments for the grammar scale factor and the word insertion penalty provided by Fig. 6.5 and 6.6 two different sets of parameter values are required to either optimize the sentence recognition rate, or the word level accuracy. Since it was a priori unclear if the handwriting recognition system should be first tuned to optimize the sentence recognition rate or to maximize the word level accuracy, two experiments were carried out to compare the results obtained with the two differently optimized recognition systems.

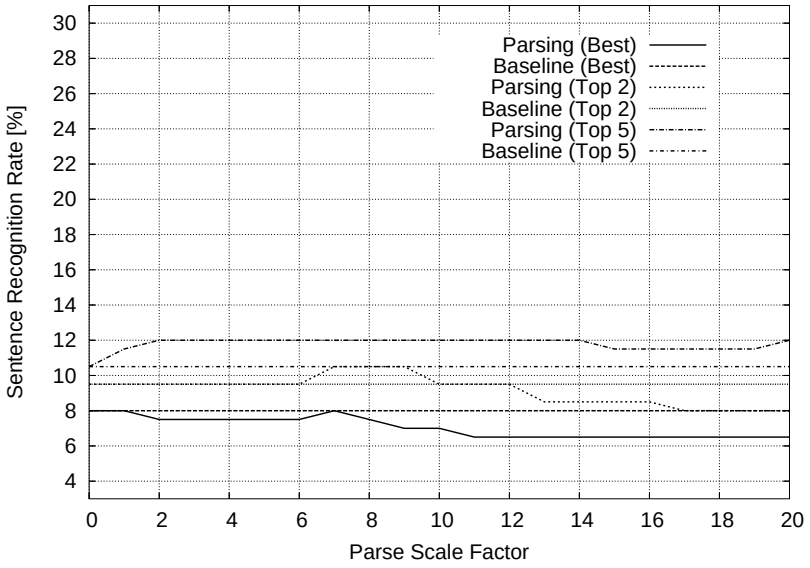

Figure 10.2 *Performance evaluation for different parse scale values. The system was first optimized for optimal sentence recognition rates.*

Fig. 10.2 provides the result for the recognizer for the writer independent environment which has been optimized for a high sentence recognition rate. The corresponding optimization for the parse scale factor for the recognizer tuned for a high word level accuracy is given in Fig. 10.3[3].

Based on these results it was decided to first optimize the handwritten sentence recognition system for a maximum word level accuracy. Then, the parse scale factor is optimized for a maximum sentence recognition rate. The optimization of the parse scale factor resulted in $\gamma = 13$ for the multi-writer system and $\gamma = 10$ for the writer independent system.

[3]Similar results were obtained for the multi-writer environment.

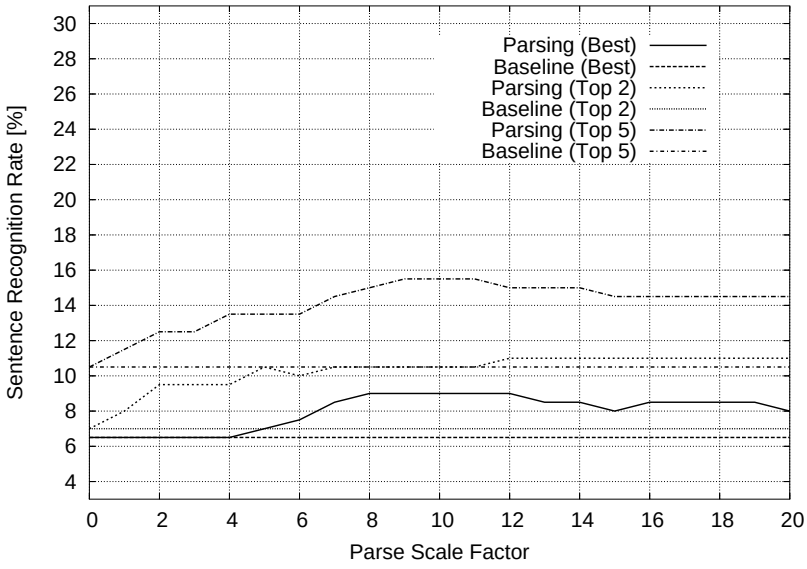

Figure 10.3
Performance evaluation for different parse scale values. The system was first optimized to maximize the word level accuracy.

10.3.2 The Minimum Parse Probability

Both a global minimum parse probability[4] independent of the parse probabilities of any n-best list and a local minimum parse probability for each n-best list were investigated.

The results proved not to be very sensitive to the minimum parse probability. For a global minimum parse probability less than any observed parse probabilities the best results were obtained. As a result, the minimum parse probability was therefore fixed at $1e-300$. A closer examination of the resulting reordered n-best lists showed that this minimum parse probability is effectively working as a filter where grammatically correct solutions will always be favored over grammatically incorrect solutions.

10.4 Validation Set Results

Using the optimized values for the parse scale factor an n-best analysis was performed on the validation set to analyze the achieved performance improvement.

Fig. 10.4 provides the results for the multi-writer system and Fig. 10.5 the results for the writer independent system. It can be seen that

[4]The minimum parse probability is used for grammatically incorrect sentence candidates as introduced in Sec. 9.4.2

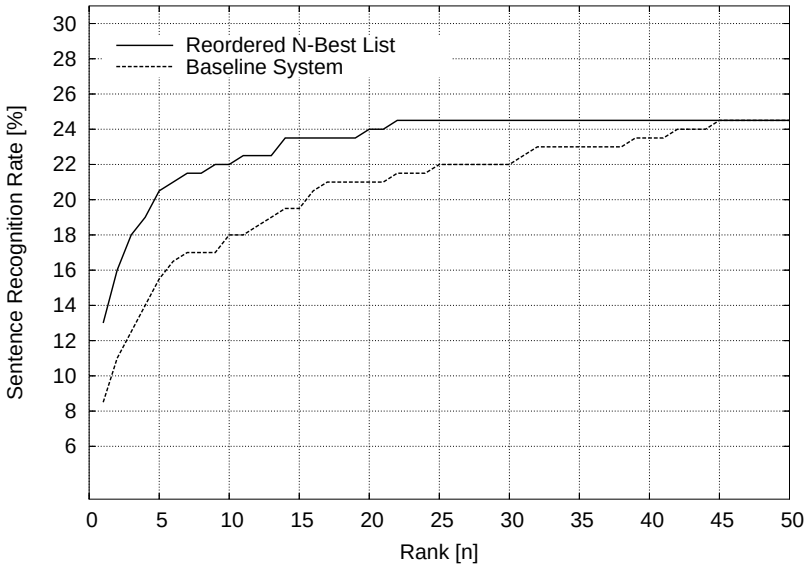

Figure 10.4
Sentence recognition rate on validation set, multi-writer system.

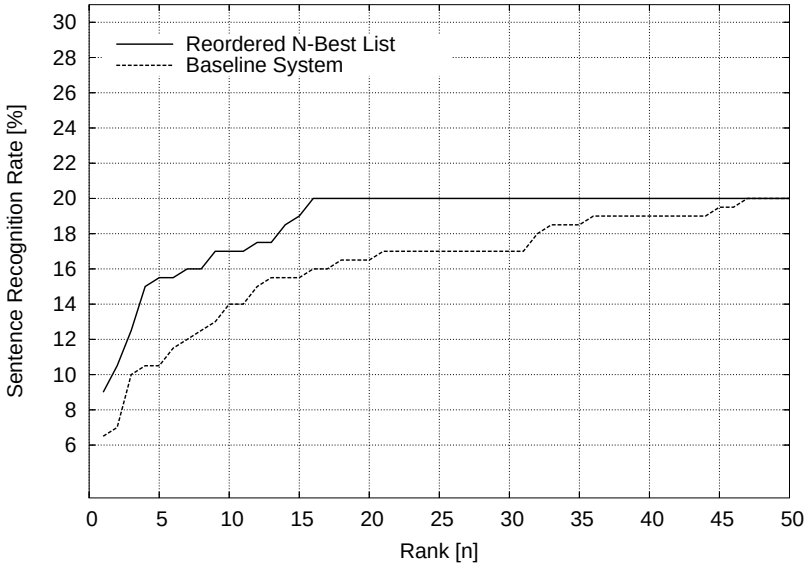

Figure 10.5
Sentence recognition rate on validation set, writer independent system.

the improvement of the combined system over the handwritten sentence recognizer is consistent for all ranks included in the n-best analysis. Since $n = 50$ has been chosen to validate the combined system, the handwritten sentence recognizer and the combined system will have the same performance when the first 50 ranks are included in the n-best analysis[5].

[5]A reordering of the n-best list does not have an effect on the performance

10.5 Test Set Results

The combination scheme has been applied in two sets of experiments. For the first experiments the 50-best sentence candidates have been involved. Further experiments include the 100-best sentence candidates provided by the handwritten sentence recognizer.

In the following two subsections the test set performances using the multi-writer task and the writer independent task are provided and compared with the performance of the baseline recognizer at the level of the best sentence candidate. Results for the 50-best and the 100-best analysis and provided in the last two subsections.

10.5.1 Results for the Multi-Writer Task

Table 10.2 summarizes the test set results of both the combined system including syntax analysis module and the baseline recognizer for the multi-writer task at the level of the best hypothesis.

Performance Measure	Baseline	Parsing	Significance
Sentence Recognition Rate	7.5%	8.0%	60%
Word Recognition Rate	76.7%	77.2%	75%
Word Level Accuracy	74.7%	75.6%	$> 99\%$

Table 10.2 *Test set results for multi-writer task.*

The highest level of significance is reached for the word level accuracy where it has been found that the absolute improvement of 0.9% is sigificant at the 99% level.

Using the word tags provided by the LPC corpus the quality of the word tags assigned by the syntax analysis module has been evaluated. For the 16 correctly recognized sentences produced by the combined system a tagging accuracy of 99.5% has been measured.

10.5.2 Results for the Writer Independent Task

Table 10.3 summarizes the test set results of both the combined system including syntax analysis and the baseline recognizer for the writer independent task at the level of the best hypothesis.

if the best result out of the first n ranks is considered.

Table 10.3
Test set results for writer independent system.

Performance Measure	Baseline	Parsing	Significance
Sentence Recognition Rate	11.0%	12.0%	70%
Word Recognition Rate	79.3%	79.4%	50%
Word Level Accuracy	76.8%	77.6%	91%

As for the multi-writer task, the most significant result is achieved for the word-level accuracy. A tagging accuracy of 99.0% has been measured for the 22 correctly recognized sentences for which a reference parse could be found in the LPC.

10.5.3 50-Best Analysis

Test set performance for the multi-writer tasks using the 50 best sentence candidates is reported in Figures 10.6 and 10.7. Fig. 10.8 and 10.9 show the n-best analysis for the writer independent experiments.

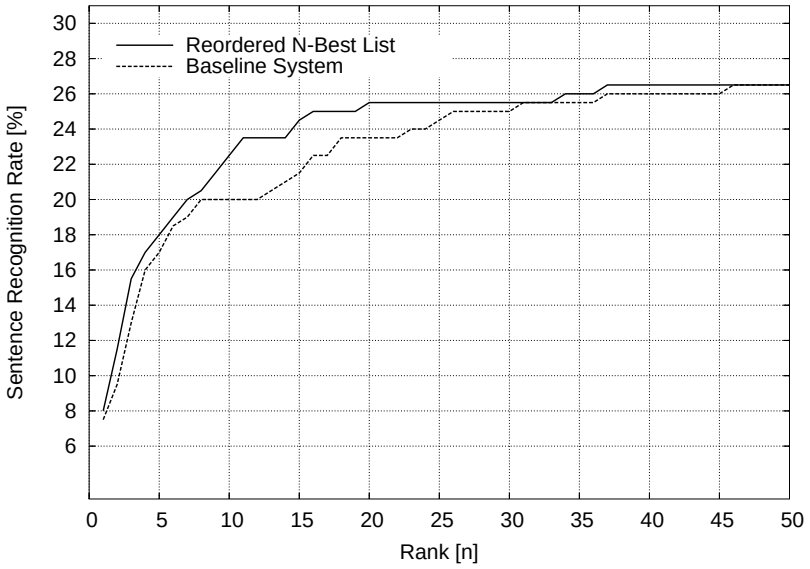

Figure 10.6
Test set sentence recognition rate, multi-writer system.

10.5.4 100-Best Analysis

In order to study the effect of the number of candidate sentences considered by the combination scheme additional experiments have

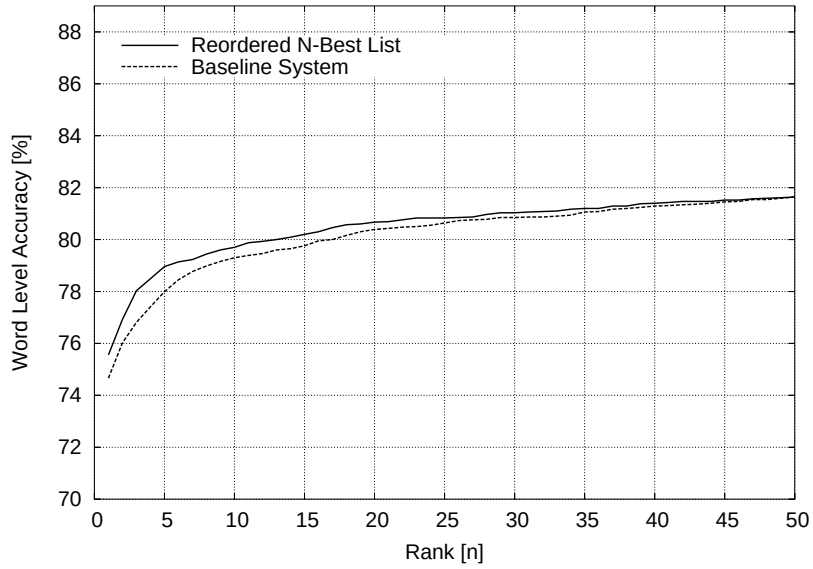

Figure 10.7
Test set word level accuracy, multi-writer system.

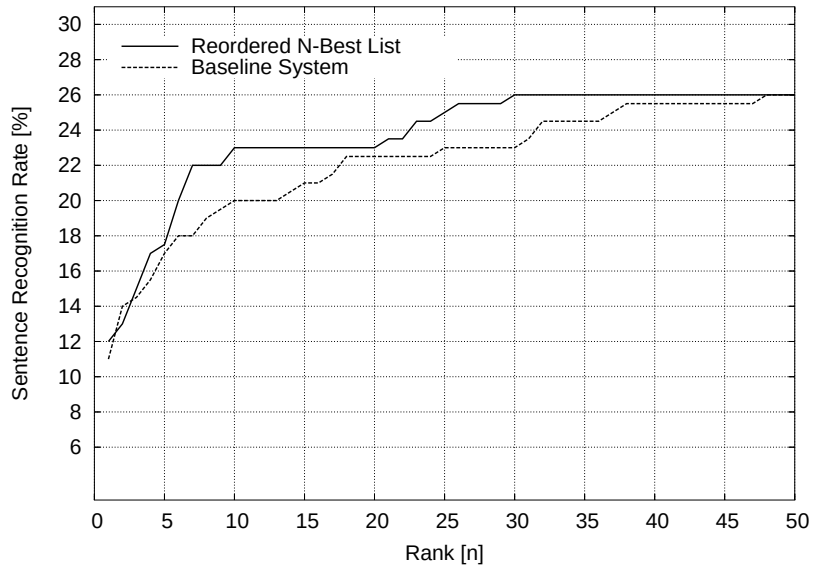

Figure 10.8
Test set sentence recognition rate, writer independent system.

been carried out which include the 100 best sentence candidates provided by the baseline recognizer.

Test set performance for the multi-writer tasks using the 100 best sentence candidates is reported in Fig. 10.10 and 10.11.

Although no further performance increase can be observed for the best sentence candidate the positive impact of the additional sentence candidates becomes obvious if more than the top 10 can-

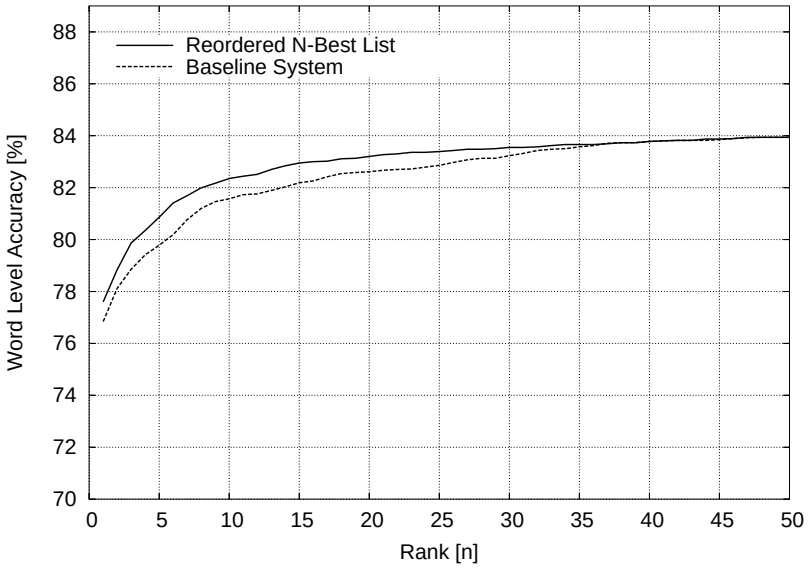

Figure 10.9
Test set word level accuracy, writer independent system.

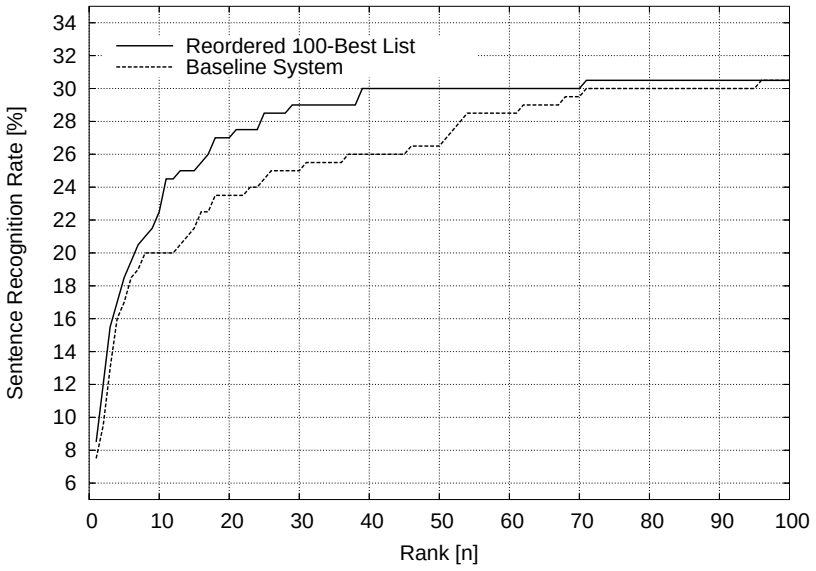

Figure 10.10
Test set sentence recognition rate, multi-writer system.

didates are considered. The performance improvement over the 50-best analysis is especially significant if the ranks 10 to 50 are considered (compare with Fig. 10.6 and 10.8).

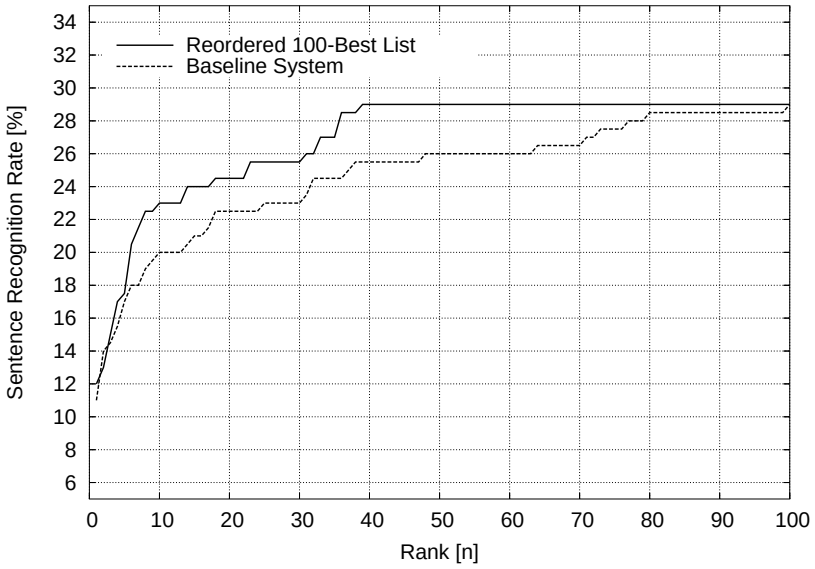

Figure 10.11
Test set sentence recognition rate, writer independent system.

11 Conclusions

A sequential coupling of a handwritten sentence recognition system and a syntax analysis module has been implemented and tested. The n-best sentence candidates provided by the recognizer are reordered in the syntax analysis module according to a sentence score determined for each sentence candidate. Using the proposed combination scheme, the sentence score is computed as the sum of the score provided by the recognizer and the weighted logarithm of the probability of the most probable parse.

A large scale stochastic context-free grammar (SCFG) for general text has been used to improve the performance of a recognition system for the first time in the domain of handwriting recognition. So far, SCFG were mostly used in the domain of speech recognition to reduce the word error rates of recognition systems designed for limited domains.

Based on the results of extensive experiments using the previously developed handwriting recognition system for general English sentences, it can be concluded that the sequential coupling of the recognizer and a syntax analysis module can improve the performance of the recognition system. Although the performance improvement for the top choice of the combined system was considerably lower on the test sets than for the validation sets, a significantly better n-best performance could be achieved for both the multi-writer and writer independent tasks. Finally, a very high tagging rate could be measured for the word tags assigned by the parsing step in the case of the correctly recognized sentences.

11.1 State of the Art

Based on the available literature about the use of syntax analysis, some general observations were made. For offline handwritten text recognition systems no integration of syntax analysis has been mentioned in literature to the knowledge of the author. Published works in the domain of OCR were restricted to artificially created OCR output and has not been tested using real machine print data as mentioned in [47][1].

The high performance improvements obtained by integration of syntactical knowledge published in the domain of speech recognition can be explained by the fact that relatively small grammars explicitly written for a specific task do not have to deal with the full amount of ambiguity present in natural language. Therefore, task-specific grammars allow for a much clearer distinction between grammatical and non-grammatical sentence candidates. This explanation is supported by the findings of other eperiments in the domain of speech recognition which involve broad coverage grammars. Such experiments are often based on the DARPA '93 HUB1 test setup and (absolute) word error reductions around 1% are typically reported.

11.2 The Stochastic Context-Free Grammar

The availability of the LPC provided the means to automatically extract SCFGs which were optimally aligned for the specified recognition tasks. Since the handwritten sentences in the IAM database are based on texts from the LOB corpus corresponding parse trees could be found in the LPC in many cases. Furthermore the production probabilities could be directly estimated from the bracketed sentences provided by the LPC.

Problems with the SCFG have been identified in experiments measuring the capability of the grammar to identify grammatically incorrect sentences. Due to the large number of productions, ex-

[1]The experiments mentioned in [73] are also based on artificially created word lattices. Therefore, the findings cannot be considered to be specifically related to handwriting recognition.

cessive over-generation has been observed for the SCFG extracted from the LPC. Furthermore the grammar does not enforce linguistic agreements between different constituents.

11.3 The Combination Scheme

The investigation of the minimum parse probability assigned to sentence candidates for which no parse could be found by the parser resulted in an interesting observation. For the optimal value of the parse scale factor it has been found that a grammatically correct sentence (according to the SCFGs used) should always be preferred over a grammatically incorrect sentence candidate.

Based on the comparison of preliminarily results [155] and the n-best results for $N = 50$ and $N = 100$ presented in Sec. 10.5 it can be concluded that performance improvements become more significant if a better baseline recognition system is used in the combination scheme and if a higher number of sentence candidates are available to the syntax analysis module.

IV Language Model Smoothing

Introduction 12

This part of the thesis investigates possible improvements of the baseline recognizer by the use of better n-gram language models. Specifically the effect of additional training texts using both natural and synthetic data is studied. Furthermore the impact of the order of the n-gram model is measured by comparing the performance of bigram and trigram language models trained under identical experimental conditions. The performance of the model is first evaluated using the perplexity measure on test sentences. In a second set of experiments the best performing models are then used in the context of the baseline recognizer introduced in Part II of the thesis.

In the following section a brief state of the art is presented. The methodology including statistical language modeling and the random sentence generation is covered in the next chapter. Chapter 14 describes the experiments and results and conclusions are drawn in chapter 15.

12.1 State of the Art

The successful use of synthetic data for the training of a bigram language model has been published in the context of the Berkeley Restaurant Project in the domain of speech recognition [64]. First, a pseudo-corpus has been compiled by using a stochastic context-free grammar in generation mode to produce 200,000 random sentences. This pseudo-corpus was then added to the regular corpus

containing 4,786 sentences to train a new bigram language model on the combined corpora. In an experiment using 364 test sentences randomly selected from the regular corpus[1] the performance of the two recognition systems was compared. As a result, the system using the new bigram language model was able to reduce the word error rate from 34.6% to 29.6%.

For the case of online handwriting recognition the effect of different statistical language models on the perplexity and the recognition rate of an online text recognition system has been described in [106]. A word bigram language model is compared to different word class bigram models. The word classes have been defined using both the available grammatical tags and a statistical clustering of the words. Finally, a combined language model using two different class bigram models has been used for comparison. It has been found that the combined word class model outperformed all other statistical language models in terms of both perplexity and the word recognition rate.

In offline handwriting recognition the use of statistical language models to improve the recognition performance of text line recognition systems has been reported in [95, 143]. In [95] the effect of unigram and bigram language models for medium sized lexicons is investigated. Using the closed vocabulary assumption, recognition rates of both unigram and bigram based recognizers are compared. For different sizes of the lexicon (2700-7700 words) the bigram based recognizer produced word recognition rates which were between 9% and 18% higher than the rates of the corresponding unigram based recognizers.

Trigram and bigram language models for medium to large lexicons (5-50,000 words) have been used in [143] to study the recognition performance of an offline handwritten text line recognition system[2]. In correspondence to the results published in [95] the system using the bigram language model clearly outperformed the system using a unigram model by 10% in average for large lexicons. However, no further improvement was achieved using trigram language models. In order to explain this counter intuitive findings the author suggests the following two reasons. First, the individual recognition of handwritten lines of text restricts the possible benefit of higher

[1] No information about the lexicon size is provided. It is also unclear, if the test sentences have been included in the training set.

[2] In contrast to [95], the closed vocabulary assumption has not been made.

order n-gram models (e.g. the full power of trigram model will only be available after the second word of each line). Second, the TDT-2 newswire corpus [23] which was used to train the bigram and the trigram language models is not aligned with the actual handwritten texts based on material from the LOB corpus [60].

Methodology 13

In the following sections different aspects of statistical language modeling and the generation of synthetic sentences for the training of statistical language models are covered.

The next section introduces the general concept of statistical language modeling. Sec. 13.2 defines the perplexity, a frequently used measure to assess the quality of a statistical language model. The most popular class of statistical language models, n-gram language models, is addressed in Sec. 13.3. Finally, the generation of synthetic texts based on random sentences produced by a stochastic context-free grammar is discussed in Sec. 13.4.

13.1 Statistical Language Modeling

Statistical language models try to capture the behavior of natural languages. It is not the goal of statistical language modeling to understand natural language or to find hidden structures behind the analyzed word sequences but to provide a probability distribution $p(s)$ over all possible sentences $s = (w_1, w_2, \ldots w_n)$[1]

[1]The techniques of statistical language modeling are not limited to sentences. Arbitrary units of word sequences may be defined (e.g. documents, texts or lines of written text).

Typical applications of statistical language modeling can be found in many natural language applications like speech recognition, information retrieval, handwriting recognition, etc. As will be explained below, statistical language models are mostly used to improve the performance of the above mentioned applications. An introduction to statistical language modeling can be found in [115] which also presents the established techniques in the field.

13.2 Perplexity of Statistical Language Models

The perplexity can be used to evaluate the quality of a statistical language model. The quality may be defined as the ability of the model to predict the next word in a sentence given the previous words in the same sentence. Because of this property, statistical language models can significantly improve the performance of recognition systems.

The perplexity $PP_T(M)$ of a language model M for a test text $T = (s_1, s_2, \ldots s_n)$ is defined using the following equation.

$$PP_T(M) = 2^{H_T(M)} \tag{13.1}$$

where $H_T(M)$ represents the (cross)entropy of the language model M for text T. The entropy $H_T(M)$ can be estimated on a test text T using the sentence probabilities provided by the language model M as shown below.

$$H_T(M) = -\frac{1}{n}\sum_{i=1}^{n}\log p(s_i) \tag{13.2}$$

In order to use perplexity values to measure the quality of language models the test texts should be as large as possible and no knowledge of the test text must be used for the construction of a language model. However, if different models are built using the same conditions for training and testing and the perplexities are only used to compare the performance of such models the closed vocabulary assumption may be safely made.

13.3 N-Gram Language Models

N-gram language models represent a simple approach to estimate sentence probabilities based on the observed frequencies of word sequences of length n found in a set of training texts. In the case of n-gram models the sentence probability $p(s)$ is decomposed into a product of conditional probabilities $p(w_i|w_{i-n+1}^{i-1})$ as follows

$$p(s) = \prod_{i=1}^{n} p(w_i|w_{i-n+1}^{i-1}) \tag{13.3}$$

where w_{i-n+1}^{i-1} represents the word sequence $(w_{i-n+1}, \ldots w_{i-1})$.

Most commonly bigram ($n = 2$) and trigram ($n = 3$) language models have been used in the domain of speech recognition. More recently, bigram language models have also been applied successfully in the domain of handwriting recognition [95, 143].

13.3.1 Relative N-Gram Frequencies

The conditional probabilities $p(w_i|w_{i-n+1}^{i-1})$ - sometimes called n-gram probabilities - are normally estimated using the relative frequencies $f(w_i|w_{i-n+1}^{i-1})$ of the corresponding n-grams extracted from large training corpora. For the computation of the relative frequencies the count $N(w_{i-n+1}^{i})$ is divided by $N(w_{i-n+1}^{i-1})$ where $N(.)$ represents the number of times the corresponding word sequence has been observed in the training corpus[2].

N-gram language models are particularly easy to build when the relative frequencies are directly used as estimates for the conditional probabilities.

(w_{i-2}, w_{i-1}, w_i)	$N(w_{i-2}^{i})$	$N(w_{i-2}^{i-1})$	$f(w_i\|w_{i-2}^{i-1})$
up the value	1	287	0.0035
the value of	39	48	0.8125
value of her	0	92	0.0000

Table 13.1 *Calculation of relative frequencies for a trigram language model.*

[2] A language model using relative frequencies implements a Maximum Likelihood estimation, since the resulting model maximizes the probability of the training corpus.

To illustrate this concept a part of a trigram language model using relative frequencies is provided in Tab. 13.1[3]. The first extreme case is represented by $p(of|the\ value)$ which means that the model is predicting the word 'of' with a 81% given the last two words 'the value'. On the other extreme $p(her|value\ of) = 0$ signifies that the model assigns a zero probability to word sequence 'value of her'. The sentence probability $p(s)$ for the sentence 'She put up the value of her money .' will therefore also be zero using this simple trigram language model. This would prevent a recognition system from finding the correct solution simply because a particular trigram did not occur in the training texts.

This zero frequency problem is the main reason why relative frequencies are not directly used as estimates of the N-gram probabilities in practice.

13.3.2 Language Model Smoothing

Model smoothing for n-gram language models is used to assign positive values to conditional probabilities of n-grams which have not been observed in the training text. To solve the zero frequency problem[4] a large number of smoothing techniques have been proposed in the literature which include discounting, backing off to lower order n-grams, linear interpolation of models of different order etc. As described in [20, 19] none of the different smoothing techniques seems to systematically outperform the others.

In this subsection only the Good-Turing smoothing technique [39] together with a backoff to lower order models [65] is described since it will be used for all the experiments reported in the next chapter[5].

The principle of the Good-Turing smoothing method lies in the reduction of a part of the probability mass of frequently observed n-grams which can then be assigned to n-grams which have not (or less frequently) been observed in a training corpus. This redistribution of probability mass is achieved through the manipulation of the observation counts of the n-grams as shown below.

[3]The counts are based on the reduced tagged LOB corpus used to train the language models for the writer independent test set.

[4]More generally, the problem of better estimating probabilities when there is not enough data.

[5]Descriptions of other smoothing techniques may be found in the literature [20, 63].

$$c^* = (c+1)\frac{N_{c+1}}{N_c} \tag{13.4}$$

where N_c represents then number of n-grams which have been observed c times in the training text. In the Good-Turing smoothing the conditional probabilities are then estimated from the relative frequencies obtained using the modified counts. This solves the zero frequency problem as follows. The modified count c^* for n-grams with a zero frequency in the training text will now be N_1/N_0 where N_0 represents the number of n-grams not observed in the training text[6] and N_1 the number of n-grams observed exactly once in the training.

As can be seen in Eq. 13.4 the Good-Turing smoothing will assign equal smoothed counts to all zero frequency n-grams. This is clearly not adequate since the same probability would be assigned to n-grams which are very unlikely from a linguistic point of view and to n-grams which would have been observed in other corpora.

Backoff n-gram modeling [65] is commonly used to assign specific probabilities for n-gram which have not been observed in the training text. The backoff probabilities for not observed n-grams are computed using the probabilities from the corresponding $n-1$-gram models. For the example $N(value\ of\ her) = 0$ provided in Tab. 13.1 the new model drops the first word and backs off to the count $N(of\ her)$ to estimate $p(her|value\ of)$.

The backoff probability $\hat{p}(w_i|w_{i-n+1}^{i-1})$ for an n-gram with a positive count can be computed by the following equation

$$\hat{p}(w_i|w_{i-n+1}^{i-1}) = \frac{c^*(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})} \tag{13.5}$$

where $c^*(w_{i-n+1}^{i})$ represents the modified count of the Good-Turing smoothing as defined in Eq. 13.4. Since $c^*(.)$ is generally lower than $c(.)$ the sum of all n-gram probabilities $\sum_{w_{i-n+1}^{i}} \hat{p}(w_i|w_{i-n+1}^{i-1})$ will be less than 1. The remaining probability mass can then be redistributed using the $n-1$-gram backoff model for the $n-grams$ which have not been observed in the training text as follows.

[6] N_0 can be computed as the number of all possible n-grams V^n for a vocabulary of a fixed size V minus the number of n-grams which have been observed at least once in the training text.

$$\hat{p}(w_i|w_{i-n+1}^{i-1}) = \alpha\ \hat{p}(w_i|w_{i-n+2}^{i-1}) \tag{13.6}$$

where α is a normalization factor depending on the word sequence w_{i-n+2}^{i-1} which ensures that the the backed off probabilities sum up to the probability mass which can be redistributed[7].

13.4 Generating Random Sentences from a SCFG

When a stochastic context-free grammar is used in generation mode, arbitrary amounts of (grammatically correct) random sentences can be produced. Such sentences can then be added to the already available text to extract a smoothed statistical language model[8].

This technique will not prevent the zero frequency problem (a corpus may contain grammatically incorrect sentences) but it can help to provide more robust estimates for the n-gram probabilities.

For the generation of a random sentence a rewriting process[9] is used in the following way. First, the start symbol S can be derived into a string of symbols by the application of productions which rewrite the nonterminal symbol S. From the resulting string the left-most nonterminal symbol can then be selected and rewritten by a suitable production found in the SCFG. If several productions are found, the selection of a production is based on its probability. This procedure can then be repeated until no more nonterminal symbols are found in the resulting string.

[7] Descriptions for the computation of the normalization factors α can be found in [20, 63].

[8] As an alternative to the use of random sentences a method to derive n-gram probabilities in closed form a SCFG is described in [129].

[9] See Sec. 8.2.1 for an example of the rewriting process.

Experiments and Results

14

14.1 Resources

For the first set of experiments English texts provided by the Brown and the Wellington corpus were used as additional material to train the language models. The two corpora are introduced in the following two subsections. The stochastic context-free grammar to produce the random sentences is mentioned in Sec. 14.1.3 and the set of recognition lattices used for the second series of experiments is specified in Sec. 14.1.4.

14.1.1 Brown Corpus

The Brown corpus [35] contains printed American English texts published in 1961 in the United States. The one million words provided by the corpus are divided into 500 text samples of 2000 words each. Text samples have been chosen for their representative quality of written American English.

14.1.2 Wellington Corpus

The Wellington corpus of New Zealand English [4] contains texts written in the years 1986-1990. It has been designed to be directly comparable with the LOB corpus of British English, the Brown

Corpus British English and the Macquaire corpus for Australian English.

14.1.3 Stochastic Context-Free Grammar

For the generation of the random sentences a stochastic context-free grammar (SCFG) for the large validation set of the writer independent task has been extracted as described in Sec. 10.1.

14.1.4 Recognition Lattices

The lattices from the writer independent system for both the large validation and the test set have been taken from the baseline recognition system described in Part II of this thesis.

14.2 Experimental Setup

This section describes the experimental setup for the investigation of the effect of additional training material and the order of the n-gram language model.

All experiments make both the segmented sentences assumption and the closed vocabulary assumption as defined in Sec. 6.1. For the second set of experiments the writer independent task as described in Sec. 6.1.3 is used. The baseline recognition systems are using both the bigram and the trigram language model extracted from the LOB corpus only. In the case of the bigram language model, the system corresponds exactly to the one described in Part II of this thesis.

The system using the modified language models always start with the sentences from the LOB corpus as described above. For each of the additional corpora (e.g. Brown, Wellington, combined Brown + Wellington, and random sentences generated by the SCFG) more and more sentences are added and the resulting corpora are then used to extract the smoothed language models.

For the first set of experiments the resulting language models are evaluated by computing the test set perplexities on the test sentences of the writer independent recognition task. In the second set of experiments the smoothed language models are used to rescore the recognition lattices of the baseline recognition system. The main advantage of the lattice rescoring technique is the significant speedup over the complete recognition of handwritten sentences. However, rescoring recognition lattices will generally not perform as good as plugging the new language model directly into the Viterbi decoding step.

Results of the perplexity experiments are provided in Sec. 14.3. The corresponding results for the handwritten sentence recognition system are documented in last two sections of this chapter.

14.3 Optimizing Perplexity

The first set of experiments investigates the effect of additional training material on the test set perplexity for the bigram and trigram language model.

Four different sources of additional sentences have been investigated to smooth a bigram and a trigram language model extracted from the sentences in the LOB corpus.

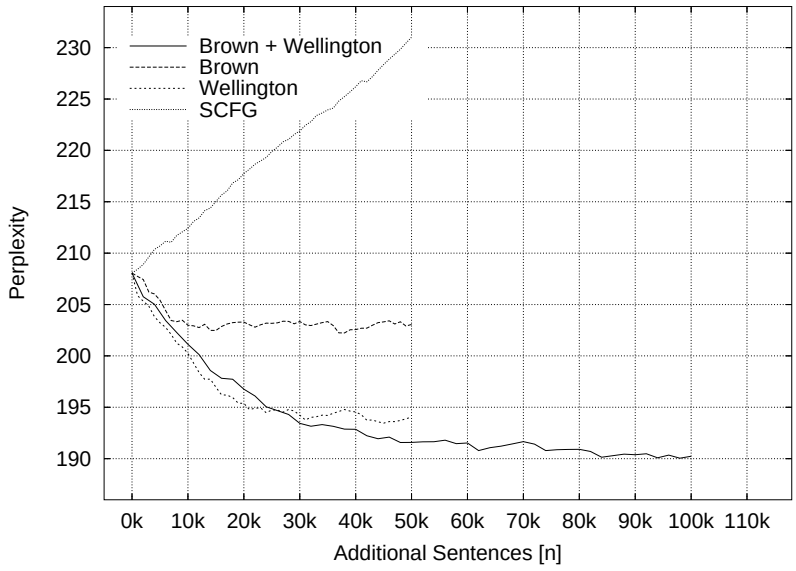

Figure 14.1
Test set perplexity for different bigram language models.

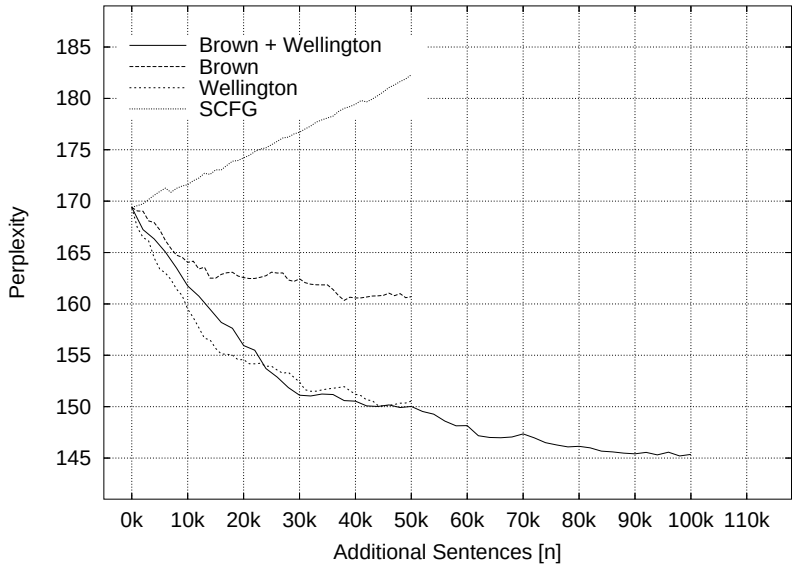

Figure 14.2
Test set perplexity for different trigram language models.

The results for the language models including more and more sentences from the different sources of additional text are provided in Fig. 14.1 for the case of bigram language models and in Fig. 14.2 for the trigram language models respectively. It can be concluded that adding random sentences to the LOB corpus to extract a smoothed bigram or trigram language model increased the test set perplexity in all cases.

Based on the results from the first set of experiments it can be concluded that for a minimal perplexity the trigram language model using both the sentences provided in the LOB corpus and the combined Brown + Wellington corpus should be used.

14.4 Optimizing System Performance

In the second set of experiments the best source of additional sentences and the optimal amount of additional material to optimize the system performance is determined. Instead of measuring the test set perplexity the newly created language models are used to rescore the lattices produced by the baseline sentence recognizer. From the rescored lattices the best candidate sentences are then used to measure the corresponding word level accuracy rates.

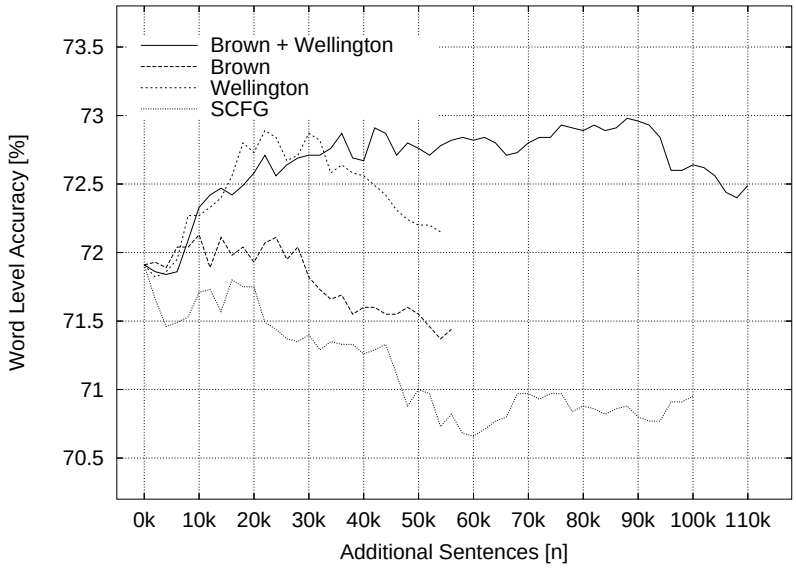

Figure 14.3 *Word level accuracy for different bigram language models, large validation set.*

Results for the validation set of the writer independent task are shown in Fig. 14.3 using different bigram language models.

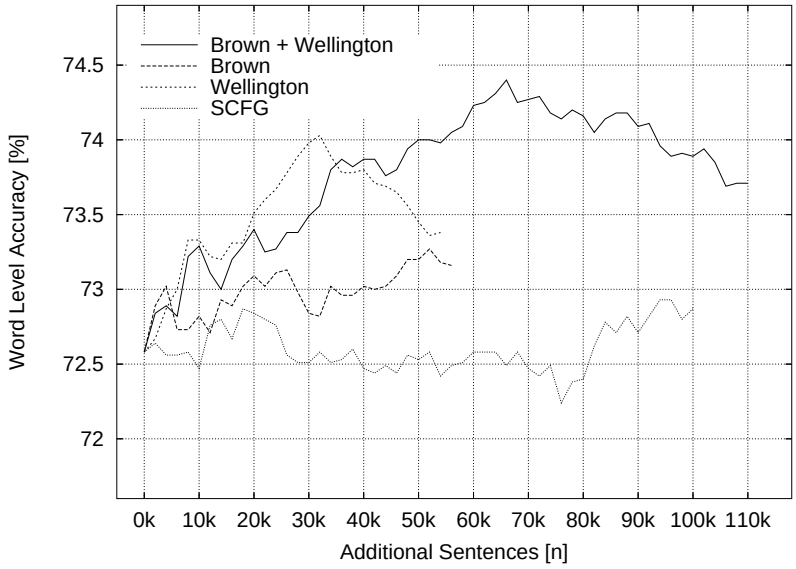

Figure 14.4 *Word level accuracy for different trigram language models, large validation set.*

The corresponding results for the trigram language models are provided in Fig. 14.4. Based on the results of the second set of experiments the following observations can be made.

- The use of the trigram language models to rescore the lattices shows a significant performance gain over the system using

corresponding bigram language models. This observation is not compatible with the results reported in [143]. However, the experimental conditions were different in several aspects. No closed vocabulary assumption was made and most lexicons used were considerably larger. Most notably, only text lines were recognized in [143] which are much shorter than complete sentences in most cases.

- For the systems using a bigram model, the random sentences generated by the SCFG have a negative effect on the system performance. In the case of trigram language model a positive effect of the random sentences cannot be observed. Therefore, the positive effect of synthetic training material to smooth a n-gram published in [64] cannot be confirmed.

- The sentences in the Wellington corpus seem to be better suited to smooth the language models than the material provided in the Brown corpus. However, the combination of both additional corpora produce the highest performance gain over the baseline system.

- Correlation between perplexity and word level accuracy is strong, when two different sources of additional data are compared (e.g. using additional sentences from the Wellington corpus leads to both better perplexity and better word level accuracy measures compared to the use of additional material from the Brown corpus).

- Perplexity and word level accuracy seem only weakly correlated when comparing two language models which use a different amount of additional sentences of the same source of additional data. Although the perplexity of both the bigram and the trigram language models is almost monotonically decreasing when more and more sentences from the combined Brown/Wellington corpus are used, a local maximum for the word level accuracy can be observed when only the first half of the combined corpus is added to the training sentences from the LOB corpus.

In order to maximize the word level accuracy of the sentence recognition system, a trigram language model using the sentences from the LOB corpus and the first 65,000 sentences of the combined

Brown + Wellington corpus will be extracted for the writer independent test set.

14.5 Test Set Results

This section reports the evaluation of the performance of the different recognition systems for the test set of the writer independent task. The first experiment validated the improved performance of the baseline system using the baseline trigram model extracted from the LOB corpus only.

Performance Measure	Baseline	Trigram	Significance
Sen. Recognition Rate	11.0%	12.0%	79%
Word Recognition Rate	79.0%	80.3%	98%
Word Level Accuracy	76.3%	78.2%	> 99%

Table 14.1 *Test set results using the trigram model for the writer independent system.*

Tab. 14.1 summarizes the results for the baseline system using a bigram language model and the system using the baseline trigram language model where only material from the LOB corpus was used for the training. It can be observed that the trigram language model did outperform the bigram based system for all performance measures. For both the word recognition rate and the word level accuracy the improvement is highly significant.

In a second experiment the sentence recognition system using a smoothed trigram language model trained on the LOB corpus and 65,000 additional sentences from the combined Brown + Wellington corpus was compared to the recognition system based on the trigram language model.

Performance Measure	Trigram	S. Trigram	Significance
Sen. Recognition Rate	12.0%	14.0%	95%
Word Recognition Rate	80.3%	81.8%	> 99%
Word Level Accuracy	78.2%	79.9%	> 99%

Table 14.2 *Test set results using the smoothed trigram model for the writer independent system.*

Tab. 14.2 provides the comparison of the system using the baseline trigram model (column 'Trigram') extracted from the LOB corpus

only and the system using the smoothed trigram language model (column 'S. Trigram') for the top choice. The achieved improvement over the baseline system using either the baseline bigram or the baseline trigram language model are highly significant for all performance measures as can be seen in column 'Significance'.

14.5.1 N-Best Analysis

The results of the n-best analysis of the smoothed trigram language models are shown in Fig. 14.5 for the sentence recognition rate and in Fig. 14.6 for the word level accuracy.

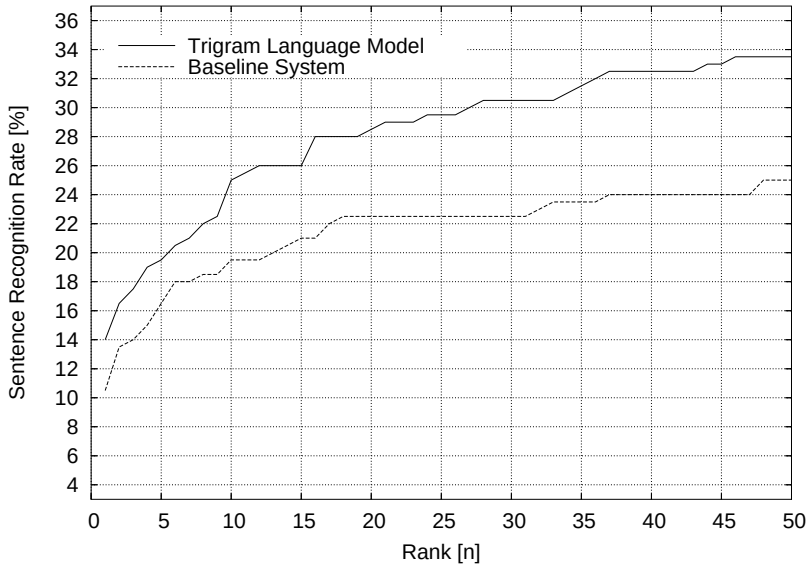

Figure 14.5 *Sentence recognition rate for the writer independent task on the test set.*

Please note that the system using the smoothed trigram language model is not reordering the n-best lists but rescoring complete recognition lattices. This is in contrast to the combination of the handwritten sentence recognition system and the syntax analysis module described in the Part III of this thesis.

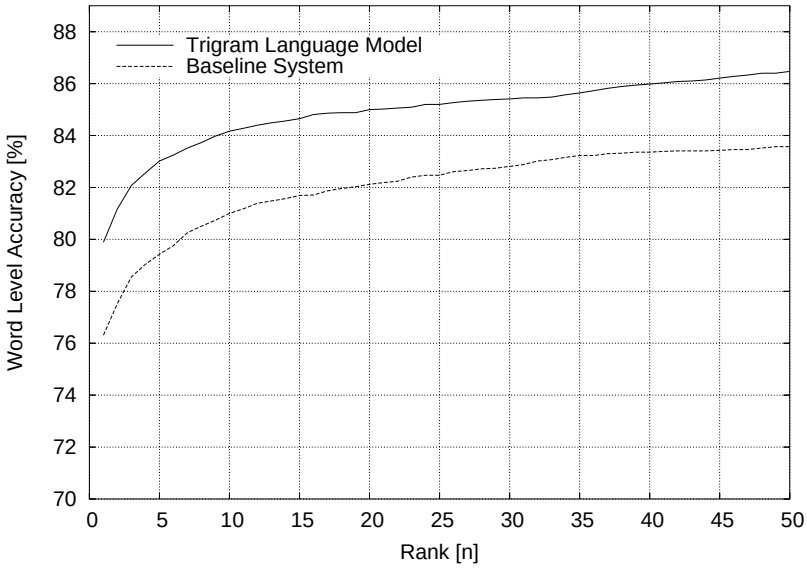

Figure 14.6
Word level accuracy for the writer independent task on the test set.

Conclusions 15

The use of different statistical language models for a handwritten sentence recognition system has been addressed in this part of the thesis. Specifically, the use of random sentences produced by a stochastic context-free grammar to smooth both bigram and trigram language models has been investigated.

15.1 Language Model Smoothing with SCFG

The results of the experiments based on the random sentences generated by the SCFG did not bring any improvements over the baseline system. This finding is in contrast to the results published in [64] where a similar pseudo-corpus was successfully used to smooth a bigram grammar in the context of the Berkeley Restaurant Project.

The discrepancy between these results could be explained by the fact that a rather small and task-specific SCFG was used for the experiments published in [64]. It can be assumed that such grammars will generate random sentences better aligned with the recognition task at hand than a large general-purpose grammar as the one used in this thesis.

15.2 Trigram Language Models

The trigram results documented in this part of the thesis suggest that the use of such models can significantly improve the system performance over a system incorporating a bigram language model only.

These findings do not correspond to the results documented in [143] where no improvement of the system using a trigram language model over the baseline system using a bigram language model could be observed. The following two reasons are believed to explain the different finding. First, the lines of handwritten text recognized by the system described in [143] contain roughly 10 words in average. Since a trigram model starts to be helpful only after the third word only 80% of the words in a text line could benefit from such language models. Second, the linguistic resources do not seem to be as well aligned in [143] as in the experimental setup chosen in the context of this thesis. Although the handwritten material from the SENIOR [120] database is based on texts from the LOB corpus no texts from this corpus were included for training of the language models. Instead, the TDT-2 corpus [23] has been used which contains transcriptions from several broadcast and newswire sources.

V Conclusions and Outlook

Conclusions 16

The main goal of the thesis was to investigate the use of syntax analysis using a large stochastic context-free grammar (SCFG) for general English sentences in the context of a state of the art, Hidden Markov Model (HMM) based offline handwriting recognition system.

In order to set up the necessary experimental framework a considerable amount of time has been spent on the preparation of the necessary resources. As a result, the first significant contribution of this thesis has been the creation of a segmented version of the IAM database. Additionally, a number of linguistic resources consistent with the IAM database were prepared which will speed up future experiments.

For the development of a state of the art recognizer, an existing offline recognition system for lines of handwritten text has been significantly improved in several aspects and enhanced for the recognition of complete handwritten sentences. First a new strategy to optimize the topology of character HMMs has been proposed and compared to existing techniques. In word recognition experiments the new strategy and a similar existing strategy increased the word recognition rate by 8% over the recognition rate obtained for the previously used character topology modeling. A further improvement has been achieved in the modeling of the emission probabilities of the HMM by using multi-Gaussian distributions instead of single Gaussians. On a medium sized text line recognition task, the word recognition rate could be improved by 40%. Finally, the integration

of the statistical language model has been fully optimized which increased the word level accuracy by 18-22%. The optimization of the language model integration has not been addressed in the domain of handwritten text recognition so far.

A new combination scheme for a sequential coupling of a handwritten sentence recognition system and a probabilistic parser is proposed for the integration of syntax analysis into the recognition process. Based on results of both multi-writer and writer independent recognition experiments it can be concluded that the use of a large scale SCFG for English sentences can significantly improve the word level accuracy of a recognizer. Although the measured improvement is not very large, this results can be seen as a substantial contribution, since no similar attempt has been made before in the field of handwriting recognition. The obtained results compare well with published experiments in the domain of speech recognition.

In the last part of the thesis, bigram and trigram language models have been trained with additional text. The effect of both natural text from additional corpora and from synthetic sentences generated by the SCFG have been investigated. Contrary to findings published in the literature, the synthetic sentences did not help to improve the bigram or trigram language models, whereas the trigram language models did improve the performance of the handwriting recognition system significantly.

To summarize the main result of the thesis, it can be stated that the use of syntax in a handwriting recognition system can help to improve the system performance. Since these results have been obtained using a general, broad coverage SCFG further improvements can be expected in more constrained domains. Such applications could make use of task-specific grammars or they could further disambiguate the output from the syntax analysis step by using some form of semantic knowledge.

17 Outlook

This chapter presents a number of issues which deserve further consideration. During the study of the literature and the optimization of the baseline recognizer it became obvious that a large number of opportunities exist to further improve the state of the art in handwritten text recognition. An incomplete list of possible future work is outlined below.

- The normalization of the text line images has been done on a global level for the recognizer presented. A local estimation of the main writing zones and the slant angle could contribute to more robust feature vectors.
- Although the HMM framework is now widely used for handwritten word and text recognition many aspects of the modeling remain open issues. Specifically, the optimization of the topology and the emission probabilities of HMM have been investigated by only a few researchers.
- Handwritten text recognition is still a new topic, and the few available publications suggest that statistical language models are critical for acceptable recognition performance. It would therefore be beneficial to study the literature in the domain of speech recognition, where the integration of statistical language models has been investigated for many years.
- The use of large, open vocabularies and rejection strategies would be an interesting area of further research.

Furthermore, the combination of the syntax analysis module and the handwriting recognition system should be investigated in more detail. Possible topics include the following.

- A significant improvement has been found when the 100-best list was considered compared to the 50-best list in the syntax analysis module. Lattice parsing helps to analyze a much larger number of recognition hypotheses in an economic way and could therefore result in further performance gains.
- The grammar which has been used in the syntax analysis module has been extracted in the most straightforward way from the Lancaster Parsed Corpus. No attempt has been made to study the effect of different grammars on the recognition performance of the entire system. The use of more compact grammars showing a lesser degree of over-generation as proposed in [17, 85] could prove to be better suited to improve the performance of the handwritten sentence recognition system.
- Grammars could be directly inferred from text corpora. Their performance could then be compared in the proposed combination scheme with the grammars used in the context of this thesis.

Finally, it has been observed that the use of image databases, recognition tasks and experimental setups is handled individually by most authors, which prevents a meaningful comparison of results between two publications in most cases. Publicly available databases should therefore also include recognition tasks and recommend an experimental protocol.

Bibliography

[1] F. Amaya, J.M. Benedí, and J.A Sánchez. Learning of stochastic context-free grammars from bracketed corpora by means of reestimation algorithms. In *VIII National Symposium on Pattern Recognition and Image Analysis, Bilbao, Spain*, pages 119–126, 1999.

[2] J.K. Baker. Trainable grammars for speech recognition. In J.J. Wolf and D.H. Klatt, editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, MIT, Cambridge, 1979.

[3] R. Bakis. Continuous speech word recognition via centisecond acoustic states. In *91st Meeting of Acoustical Society of America*, Washington DC, USA, 1976.

[4] L. Bauer. *Manual of Information to accompany The Wellington Corpus of Written New Zealand English, for use with Ditigal Computers*. Department of Linguistics, Victoria University, Wellington, New Zealand, 1993.

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

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

[7] U. Bhattacharya and B.B. Chaudhuri. A mayority voting scheme for multiresolution recognition of handprinted numerals. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 16–20, 2003.

[8] A. Biem. A model selection criterion for classification: Application to HMM topology optimization. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 104–108, 2003.

[9] H. Bourlard, H. Hermansky, and N. Morgan. Towards increasing speech recognition error rates. *Speech Communication*, 18:205–231, 1996.

[10] 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.

[11] A. Brakensiek, J. Rottland, and G. Rigoll. Confidence measures for an address reading system. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 294–298, 2003.

[12] H. Bunke. Recognition of cursive roman handwriting - past, present and future. In *7th Int. Conf. on Document Analysis and Recognition, Edinbourgh, Scotland*, volume 1, pages 448–459, 2003.

[13] H. Bunke, M. Roth, and E.G. Schukat-Talamazzini. Off-line handwriting recognition using hidden Markov models. *Pattern Recognition*, 55(1):75–89, 1995.

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

[15] J.-C. Chappelier and M. Rajman. A generalized CYK algorithm for parsing stochastic CFG. *Actes de TAPD*, pages 133–137, 1998.

[16] J.-C. Chappelier, M. Rajman, R. Aragüés, and A. Rozenknop. Lattice parsing for speech recognition. In *6e Conf. sur le Traitement Automatic du Langage Naturel (TALN99)*, pages 95–104, 1999.

[17] E. Charniak. Tree-bank grammars. CS-92-2, Dep. of Computer Science, Brown University, Providence RI, USA, 1996.

[18] C. Chelba. *Exploiting Syntactic Structe for Natural Language Modeling*. PhD thesis, Johns Hopkins University, Baltimore MD, USA, 2000.

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

[20] S.F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. TR-10-98, Center for Research in Computing Technology, Harvard University, Cambridge MA, USA, 1998.

[21] Z. Chi and S. Geman. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305, 1998.

[22] A. Choisy and A. Belaid. Coupling of a local vision by Markov field and a global vision by neural network for the recognition of handwritten words. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 2, pages 849–853, 2003.

[23] C. Cieri, D. Graff, M. Liberman, N. Martey, and S. Strassel. The TDT-2 text and speech corpus. In *DARPA Broadcast News Workshop*, 1999.

[24] World Wide Web Consortium. Extensible markup language (XML). http://www.w3c.org/XML/, 2003.

[25] C. Crowner and J.J. Hull. A hierarchical pattern matching parser and its application to word shape recognition. In *1st Int. Conf. on Document Analysis and Recognition*, volume 1, pages 323–331, 1991.

[26] D. D'Amato, E. Kuebert, and A. Lawson. Results from a performance evaluation of handwritten address recogntion systems for the united states postal service. In *7th Int. Workshop on Frontiers in Handwriting Recognition*, pages 189–198, Amsterdam, The Netherlands, 2000.

[27] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc.*, 39(1):1–38, 1977.

[28] S. Edelman, T. Flash, and S. Ullmann. Reading cursive handwriting by alignment of letter prototypes. *Int. Journal of Computer Vision*, 5:303–331, 1990.

[29] D. Elliman and N. Sherkat. A truthing tool for generating a database of cursive words. In *6th Int. Conf. on Document Analysis and Recognition, Seattle WA, USA*, pages 1255–1262, 2001.

[30] A. J. Elms, S. Procter, and J. Illingworth. The advantage of using an HMM-based approach for faxed word recognition. *Int. Journal on Document Analysis and Recognition*, 1(1):18–36, February 1998.

[31] Y. LeCun et al. Comparison of learning algorithms for handwitten digit recognition. In *Int. Conf. on Artificial Neural Networks, France*, pages 53–60, 1995.

[32] M. Feldbach and K.D. Tönnies. Word segmentation of handwritten dates in historical documents by combining semantic a-priori-knowledge with local features. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 333–337, 2003.

[33] J.D. Ferguson. Variable duration models for speech. In *Proc. of Symposium on Application of Hidden Markov Models to Text and Speech*, pages 143–179, October 1980.

[34] G.D. Forney. The Viterbi algorithm. *Proc. IEEE*, 61:268–278, 1973.

[35] W. N. Francis and H. Kucera. *Brown Corpus Manual, Manual of Information to accompany A Standard Corpus of Present-Day Edited American English, for use with Ditigal Computers*. Department of Linguistics, Brown University, Providence RI, USA, 1979.

[36] J. García-Hernandez, J.-A. Sánchez, and J.-M. Benedí. Performance and improvements of a language model based on stochastic context-free grammars. In *1th Iberian Conference on Pattern Recognition and Image Analysis*, pages 271–278, Puerto de Andratz, Spain, 2003.

[37] M.D. Garris. Design and collection of a handwritten sample image database. Technical report, NIST, 1992. Social Science Computer Review, volume 10, pages 196-214.

[38] R. Garside, G. Leech, and T. Váradi. *Manual of Information for the Lancaster Parsed Corpus*. Norwegian Computing Center for the Humanities, Bergen, 1995.

[39] I.J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264, 1953.

[40] N. Gorski, V. Anisimov, E. Augustin, O. Baret, and S. Maximor. Industrial bank check processing: the A2iA check reader. *Int. Journal on Document Analysis and Recognition*, 3:196–206, 2001.

[41] 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. Conf. on Document Analysis and Recognition*, pages 523–526, Bangalore, India, 1999.

[42] V. Govindaraju, R.K. Srihari, and S.N. Srihari. Handwritten text recognition. In A. L. Spitz and A. Dengel, editors, *Document Analysis Systems*, pages 288–304. World Scientific, 1995.

[43] V. Govindaraju and S. Tulyakov. Postal address block location by contour clustering. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 429–432, 2003.

[44] S. Guenter and H. Bunke. Optimizing the number of states, training iterations, gaussians in an HMM-based handwritten word reconizer. In *7th Int. Conf. on Document Analysis and Recognition, Edinbourgh, Scotland*, volume 1, pages 472–476, 2003.

[45] D. Guillevic and C.Y. Suen. Cursive script recognition to the processing of bank cheques. In *Int. Conf. on Document Analysis and Recognition 95, Montreal, Canada*, volume 1, pages 11–14, 1995.

[46] D. Guillevic and C.Y. Suen. HMM word recognition engine. In *Fourth Int. Conf. on Document Analysis and Recognition, Ulm, Germany*, volume 2, pages 544–547. IEEE, IEEE Computer Society Press, 1997.

[47] I. Guyon, R.M. Haralick, J.J. Hull, and I.T. Phillips. Data sets for OCR and document image understanding research. In H. Bunke and P.S.P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, pages 780–799. World Scientific, 1997.

[48] T.M. Ha and H. Bunke. Off-line handwritten numeral recognition by perturbation method. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5):535–539, 1997.

[49] T.M. Ha, M. Zimmermann, and H. Bunke. Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods. *Pattern Recognition*, 31(3):257–272, March 1998.

[50] A. Hauenstein and H.H. Weber. An investigation of tightly coupled time synchronous speech language interfaces using a unification grammar. In *Proc. of AAAI Workshop on Integration of Natural Language and Speech Processing*, pages 42–49, 1994.

[51] T. Hong and J.J. Hull. Text recognition enhancement with a probabilistic lattice chart parser. In *Int. Conf. on Document Analysis and Recognition*, pages 222–225, Tsukuba, Japan, 1993.

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

[53] J.J. Hull. Incorporating language syntax in visual text recognition with statistical model. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(12):1251–1256, 1996.

[54] J.J. Hull, A.C. Downton, M. Garris, C.Y. Suen, and K. Yamamoto. Databases of off-line handwritten english text. Technical report, IAPR TC 11, 1994.

[55] S. Impedovo, L. Ottaviano, and S. Occhiegro. Optical character recognition – a survey. *Int. Jounal of Pattern Recognition and Artificial Intelligence*, 5:1–24, 1991.

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

[57] F. Jelinek. Continuous speech recognition by statistical methods. *Proc. IEEE*, 64:532–536, 1976.

[58] F. Jelinek, L.R. Bahl, and R.L. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Trans. on Information Theory*, 21(3):250–256, 1975.

[59] S. Johansson, E. Atwell, R. Garside, and G. Leech. *The Tagged LOB Corpus, Users's Manual*. Norwegian Computing Center for the Humanities, Bergen, Norway, 1986.

[60] 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.

[61] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[62] A. S. Britto Jr, R. Sabourin, F. Bortolozzi, and C. Y. Suen. A two-stage HMM-based system for reconizing handwritten numeral strings. In *6th Int. Conf. on Document Analysis and Recognition, Seattle WA, USA*, pages 396–400, 2001.

[63] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2000.

[64] D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchman, and N. Morgan. Using a stochastic context-free grammar as a language model for speech recognition. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 189–192, Detroit MI, USA, 1995.

[65] S.M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE*

Trans. on Acoustics, Speech and Signal Processing, 35(3):400–401, 1987.

[66] G. Kaufmann. *Erkennung von Handschrift mittels Hidden Markov Modellen für das automatische Lesen von Checkbeträgen*. PhD thesis, Universität Bern, 1998.

[67] 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.

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

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

[70] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis. Skew angle estimation for printed and handwritten documents using the wigner-ville distribution. *Image and Vision Computing*, 20:813–824, 2002. Elsevier Science.

[71] E. Kavallieratou, N. Fakotakis, and G. Kokkinakis. An unconstrained handwriting recogntion system. *Int. Journal on Document Analysis and Recognition*, 4:226–242, 2002.

[72] E. Kavallieratou, K. Sgarbas, N. Fakotakis, and G. Kokkinakis. Handwritten word recognition based on structural charactersitics and lexical support. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 562–566, 2003.

[73] F.G. Keenan, L.J. Evett, and R.J. Whitrow. A large vocabulary stochastic syntax analyser for handwriting recognition. In *1st Int. Conf. on Document Analysis and Recognition 91, Saint-Malo, France*, pages 794–802, 1991.

[74] K. K. Kim, Y. K. Chung, and C. Y. Suen. Post-processing scheme for improving recognition performance of touching handwritten numeral strings. In *16th Int. Conf. on Pattern Recognition*, volume 2, pages 1051–4651, Quebec, Canada, 2002.

[75] K. Kita, T. Kawabata, and H. Saito. HMM continuous speech recognition using predictive lr parsing. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 703–706, 1989.

[76] K. Kita and W.H. Ward. Incorporating lr parsing into SPHINX. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 269–272, 1991.

[77] G. Koch, L. Heutte, and T. Paquet. Numerical sequence extraction in handwritten incoming mail documents. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 369–373, 2003.

[78] A. L. Koerich, Y. Leydier, R. Sabourin, and C. Y. Suen. A hybrid large vocabulary handwritten word recognition system using neural networks and hidden Markov models. In *8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 99–104, Niagra-on-the-Lake, Canada, August 2002.

[79] Electrotechnical Laboratory. *ETL-6 character image database*. Japan Electronic Industry Development Association, http://www.etl.go.jp/etlcdb/#English.

[80] C.-H. Lee and L.R. Rabiner. A frame-synchronous network search algorithm for connected word recognition. *IEEE Trans. on ASS*, 37(11):1649–1658, 1989.

[81] J. J. Lee, J. Kim, and J. H. Kim. Data-driven design of HMM topology for online handwriting recognition. In H. Bunke and T. Caelli, editors, *Hidden Markov Models: Applications in Computer Vision*, volume 45 of *Machine Perception and Artificial Intelligence*, pages 107–121. World Scientific, 2001.

[82] G. Leedham, S. Varma, A. Patankar, and V. Govindaraju. Separating text and background in degraded document images - a comparison of global thresholding techniques for multi-stage thresholding. In *8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 245–249, Niagra-on-the-Lake, Canada, 2002.

[83] S.E. Levinson. Continuously variable duration hidden markov models for automatic speech recognition. *Computer Speech and Language*, 1:29–45, March 1986.

[84] D. Li, A. Biem, and J. Subrahmonia. HMM topology optimization for handwriting recognition. In *26th Int. Conf. on Acoustics, Speech, and Signal Processing*, 2001.

[85] D. Linares, J.-A. Sánchez, J.-M. Benedí, and F.Torres. Learning of stochastic context-free grammars by means of estimation algorithms and initial treebank grammars. In *1th Iberian Conference on Pattern Recognition and Image Analysis*, pages 403–410, Puerto de Andratz, Spain, 2003.

[86] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition using state-of-the-art techniques. In *8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 320–325, Niagra-on-the-Lake, Canada, August 2002.

[87] C.-L. Liu, H. Sako, and H. Fujisawa. Integrated segmentation and recognition of handwritten numerals: Comparison of classification algorithms. In *8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 369–374, Niagra-on-the-Lake, Canada, August 2002.

[88] S. Madhvanath and V. Govindaraju. Local reference lines for handwritten phrase recognition. *Pattern Recognition*, 32(12):2021–2028, 1999.

[89] U. Mahadevan and S. N. Srihari. Parsing and recognition of city, state, and zipcodes in handwritten addresses. In *5th Int. Conf. on Document Analysis and Recognition*, pages 325–328, Bangalore, India, 1999.

[90] J. Mao, P. Sinha, and M. Mohiuddin. A system for cursive handwritten address recognition. In *14th Int. Conf. on Pattern Recognition*, pages 1285–1287, Brisbane, Australia, 1998.

[91] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. The Penn treebank: Annotating predicate argument structure. volume 19, pages 313–330, 1993.

[92] U.-V. Marti. *Offline Erkennung handgeschriebener Texte*. PhD thesis, University of Bern, Switzerland, Bern, 2000.

[93] U.-V. Marti and H. Bunke. Towards general cursive script recognition. In S.-W. Lee, editor, *Advances in Handwriting Recognition*, pages 203–212. World Scientific, 1999.

[94] U.-V. Marti and H. Bunke. Handwritten sentence recognition. In *15th Int. Conf. on Pattern Recognition*, volume 3, pages 467–470, Barcelona, Spain, 2000.

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

[96] U.-V. Marti and H. Bunke. The IAM-database: an English sentence database for off-line handwriting recognition. *Int. Journal on Document Analysis and Recognition*, 5:39–46, 2002.

[97] D. Menoti, D.L. Borges, J. Facon, and A. de Souza Britto Jr. Segmentation of postal envelopes for address block location: an approach based on feature selection in wavelet space. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 2, pages 699–703, 2003.

[98] M. Mohamed and P. Gader. Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(5):548–554, 1996.

[99] M.E. Morita, J. Facon, F. Bortolozzi, S. Garnes, and R. Sabourin. Mathematical morphology and weighted least squares to correct handwriting baseline skew. In *5th Int. Conf. on Document Analysis and Recognition 99, Bangalore, India*, pages 430–433, 1999.

[100] M.E. Morita, R. Sabourin, F. Bortolozzi, and C.Y. Suen. A recognition and verification strategy for handwritten word recognition. In *7th Int. Conf. on Document Analysis and Recognition, Edinbourgh, Scotland*, volume 1, pages 482–486, 2003.

[101] H. Ney. Stochastic grammars and pattern recognition. In P. Laface and R. De Mori, editors, *Speech Recognition and Understanding. Recent Advances*, pages 319–344. Springer Verlag, 1992.

[102] G. Nicchiotti and C. Scagliola. Generalised projections: a tool for cursive handwriting normalisation. In *5th Int. Conf. on Document Analysis and Recognition 99, Bangalore, India*, pages 729–732, 1999.

[103] K. Nitz, W. Cruz, H. Aradhye, T. Shaham, and G. Myers. An image-based mail facing and orientation system for enhanced postal automation. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 2, pages 694–698, 2003.

[104] A. Ogawa, K. Takeda, and F. Itakura. Balancing acoustic and linguistic probabilities. In *IEEE Conference on Acoustics, Speech and Signal Processing*, pages 181–184, 1998.

[105] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-9:pp. 62–66, 1979.

[106] F. Perraud, C. Viard-Gaudin, E. Morin, and P.-M. Lallican. N-gram and n-class models for on line handwriting recognition. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 2, pages 1053–1057, 2003.

[107] J.F. Pitrelli and M.P. Perrone. Confidence modeling for verificatio post-processing for handwriting recognition. In *8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 30–35, Niagra-on-the-Lake, Canada, August 2002.

[108] J.F. Pitrelli and M.P. Perrone. Confidence-scoring post-processing for off-line handwritten-character recognition verification. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 278–282, 2003.

[109] R. Plamondon and S. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22:63–84, 2000.

[110] P.Y.Simard, D. Steinkraus, and J.C. Platt. Best practices for convolutional nerual networks applied to visual document analysis. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 2, pages 958–962, 2003.

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

[112] T.M. Rath and R. Manmatha. Features for word spotting in historical manuscripts. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 218–222, 2003.

[113] B. Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276, 2001.

[114] G. Roeloff. Portable network graphics, a turbo-studly image format with lossless compression. http://www.libpng.org, 2003.

[115] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proc. of the IEEE*, 88:1270–1278, 2000.

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

[117] M.-P. Schambach. Determination of the number of writing variants with an HMM based cursive word recognition system. In *7th Int. Conf. on Document Analysis and Recognition, Edinbourgh, Scotland*, volume 1, pages 119–123, 2003.

[118] M.-P. Schambach. Model length adaptation of an HMM based cursive word recognition system. In *7th Int. Conf. on Document Analysis and Recognition, Edinbourgh, Scotland*, volume 1, pages 109–113, 2003.

[119] J. Schürmann. *Pattern Classification*. John Wiley and Sons, Inc., 1996.

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

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

[122] J.C. Simon and O. Baret. Cursive word recognition. In Impedovo and Simon, editors, *From Pixels to Features III: Frontiers in Handwriting Recognition*, pages 241–260. Elsevier Science, 1992.

[123] B.-K. Sin and J. H. Kim. Ligature modeling for online cursive script recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(6):623–633, 1997.

[124] R.K. Srihari, S. Ng, C.M. Baltus, and J. Kud. Use of language models in on-line sentence/ phrase recognition. In *3rd Int. Workshop on Frontiers in Handwriting Recognition*, pages 284–294, Buffalo NY, USA, 1993.

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

[126] T. Steinherz, E. Rivlin, and N. Intrator. Off-line cursive word recognition - a survey. *Int. Journal on Document Analysis and Recognition*, 2:90–110, 1999.

[127] G. Stemmer, V. Zeissler, E. Nöth, and H. Niemann. Towards a dynamic adjustment of the language weight. *Lecture Notes in Computer Science*, 2166:323–328, 2001.

[128] A. Stolcke. *Bayesian Learning of Probabilistic Language Models.* PhD thesis, University of California, Berkeley CA, 1994.

[129] A. Stolcke and S. Omohundro. Inducing probabilistic grammars by Bayesian model merging. In R.C. Carrasco and J. Oncina, editors, *Grammatical Inference and Applications*, volume Second Int. Colloquium on Grammar Inference, pages 106–118, Alicante, Spain, 1994.

[130] A. Stolke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. Technical Report TR-93-065, International Computer Science Institute, Berkeley CA, USA, 1993.

[131] A. Stolke and S. Omohundro. Best-first model merging for hidden Markov model induction. Technical Report TR-94-003, International Computer Science Institute, Berkeley CA, USA, 1994.

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

[133] C.I. Tomai, B. Zhang, and V. Govindaraju. Transcript mapping for historic handwritten document images. In *8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 413–418, Niagra-on-the-Lake, Canada, August 2002.

[134] M. Tomita, editor. *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*. Kluwer Academic Publishers, 1986.

[135] S. Uchida and H. Sakoe. An off-line character recognition method employing model-dependent pattern normalization by an elastic membrane model. In *5th Int. Conf. on Document Analysis and Recognition*, pages 499–502, Bangalore, India, 1999.

[136] S. Uchida and H. Sakoe. A handwritten character recognition method based on unconstrained elastic matching and eigen-deformations. In *8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 72–77, Niagra-on-the-Lake, Canada, August 2002.

[137] S. Uchida and H. Sakoe. Handwritten character recognition using elastic matching based on a class-dependent deformation model. In *7th Int. Conf. on Document Analysis and Recognition, Edinbourgh, Scotland*, pages 163–167, 2003.

[138] S. Uchida, E. Taira, and H. Sakoe. Nonuniform slant correction using dynamic programming. In *6th Int. Conf. on Document Analysis and Recognition, Seattle WA, USA*, pages 434–438, 2001.

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

[140] A. Vinciarelli. *Offline Cursive Handwriting: From Word to Text Recognition*. PhD thesis, University of Bern, Switzerland, Bern, 2003.

[141] A. Vinciarelli and S. Bengio. A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22:1043–1050, 2001.

[142] A. Vinciarelli and S. Bengio. Off-line cursive word recognition using continuous density HMMs trained with pca and ica featurew. In *16th Int. Conf. on Pattern Recognition*, volume 3, pages 81–84, Quebec, Canada, August 2002.

[143] A. Vinciarelli, S. Bengio, and H. Bunke. Offline recognition of large vocabulary cursive handwritten text. In *7th Int. Conf. on Document Analysis and Recognition, Edinbourgh, Scotland*, volume 2, pages 1101–1105, 2003.

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

[145] Q. Wang, T. Xia, C.L. Tan, and L. Li. Directional wavelet approach to remove document image interference. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 2, pages 736–740, 2003.

[146] W. Wang, A. Brakensiek, A. Kosmala, and G. Rigoll. Multi-branch and two-pass HMM modeling approaches for off-line cursive handwriting recognition. In *6th Int. Conf. on Document Analysis and Recognition, Seattle WA, USA*, pages 231–235, 2001.

[147] Q. Xu, L. Lam, and C.Y. Suen. Automatic segmentation and recognition system for handwritten dates on candian bank cheques. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 2, pages 704–708, 2003.

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

[149] B.A. Yanikoglu and P.A. Sandon. Off-line cursive handwriting recognition using style parameters. PCS-TR93-192, Dartmouth College, Hanover NH, USA, 1993.

[150] S. J. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, editors. *The HTK Book (for HTK Version 3.2).* http://htk.eng.cam.ac.uk/. Cambridge University Engineering Department, 2002.

[151] S. J. Young, N. H. Russell, and J. H. S. Thornton. Token passing: a conceptual model for connected speech recognition systems. CUED technical report F INFENG/TR38, Cambridge University, 1989.

[152] D. H. Younger. Recognition of context-free language in time n^3. *Information and Control*, 10(2):189–208, February 1967.

[153] M. Zimmermann and H. Bunke. Automatic segmentation of the IAM off-line handwritten English text database. In *16th Int. Conf. on Pattern Recognition*, volume 4, pages 35–39, Quebec, Canada, August 2002.

[154] M. Zimmermann and H. Bunke. Hidden Markov model length optimization for handwriting recognition systems. In *8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 369–374, Niagra-on-the-Lake, Canada, August 2002.

[155] M. Zimmermann, J.-C. Chappelier, and H. Bunke. Parsing n-best lists of handwritten sentences. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, volume 1, pages 572–576, 2003.

Curriculum Vitae

Personal details

First Name:	Matthias
Second Name:	Zimmermann
Date of Birth:	March 31^{st}, 1969
Address:	Riedmatt 16
	CH-6300 Zug
Nationality:	Swiss

Education

1999 - 2003	Ph.D. in Science (expected by November 2003), at University of Bern, Switzerland
2001	Postgraduate Course in Speech and Lanaguage Engineering, Ecole Politechnique Federale de Lausanne, Switzerland
1998	Internship at IBM Almaden Research Center, San Jose, USA
1990 - 1996	Master of Scienence, Study of Computer Science at University of Bern, Switzerland
1982 - 1989	Matura, Study at Kantonsschule Zug, Switzerland

Employment

2003	Research Assistant at University of Bern, Switzerland
2000 -	Specialist at xbrain.ch GmbH Hünenberg, Switzerland
1999 - 2000	Analyst at Credit Suisse Bern, Switzerland
1996 - 1997	Software Quality Manager at PostFinance Zollikofen, Switzerland

Publications

Journal Articles

T.M. Ha, M. Zimmermann and H. Bunke, "Off-line Handwritten Numeral String Recognition by Combining Segmentation-Based and Segmentation-Free Methods", *Pattern Recognition*, volume 31(3), pages 257-272, 1998

M. Zimmermann and J. Mao, "Lexicon Reduction using Key Characters in Cursive Handwritten Words", *Pattern Recognition Letters*, volume 20, pages 1297-1304, 1999.

Conference Papers

T.M. Ha, G. Kaufmann, M. Zimmermann and H. Bunke, "Handwriting Recognition for a Post Check Reading System", *1st Int. Conf. on Knowledge Based Intelligent Electronic Systems*, Adelaide Australia, pages 54-63, 1997

M. Zimmermann and H. Bunke, "Hidden Markov Model Length Optimization for Handwriting Recognition Systems", *8th Int. Workshop on Frontiers in Handwriting Recognition*, Niagra-on-the-Lakes Canada, pages 54-63, 2002

M. Zimmermann and H. Bunke, "Automatic Segmentation of the IAM Off-line Handwritten English Text Database", *16th Int. Conf. on Pattern Recognition*, Quebec Canada, volume 4, pages 35-39, 2002

M. Zimmermann, J.-C. Chappelier and H. Bunke, "Parsing N-Best Lists of Handwritten Sentences", *7th Int. Conf. on Document Analysis and Recognition*, Edinbourgh Scotland, volume 1, pages 572-576, 2003