Handwriting Recognition in Historical Documents

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Andreas Fischer

von Bern BE

Leiter der Arbeit:

Prof. em. Dr. Horst Bunke

Institut für Informatik und angewandte Mathematik
Universität Bern

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Bern, den 13. März 2012

Der Dekan:
Prof. Dr. Silvio Decurtins

Abstract

This thesis investigates pattern recognition methods for handwriting recognition in historical documents. The aim of these methods is to automatically extract textual content from digitized manuscript images. Based on their textual content, millions of historical manuscript images could be integrated in digital libraries, which would help to preserve our cultural heritage by making it readily accessible to researchers and the public.

Two state-of-the-art strategies are pursued to model and recognize characters, words, and sentences. First, a generative strategy using hidden Markov models (HMM) and secondly, a discriminative strategy using a special form of recurrent neural networks (NN). The learning-based systems are generic in the sense that they can learn character appearance models for arbitrary alphabetical languages as long as a number of training samples are provided. They operate at the level of text lines avoiding prior word and character segmentation which is prone to errors for touching characters, broken characters, variable word spacing, and difficult image conditions stemming, e.g., from paper texture, damaged parchment, faded ink, and ink bleed-through.

In order to evaluate the performance of the recognition systems, a data set of several historical scripts and languages was created. The IAM historical document database (IAM-HistDB) includes Latin texts from the 9th century written in Carolingian minuscules (Saint Gall database), medieval German texts from the 13th century written in Gothic minuscules (Parzival database), and longhand English texts from the 18th century (George Washington database). Altogether, over hundred annotated manuscript pages were used for experimental evaluation in this thesis and were made publicly available on the Internet.

Four subproblems of handwriting recognition in historical documents are addressed in this thesis, namely ground truth creation, automatic transcription, keyword spotting, and transcription alignment. Ground truth creation consists of annotating text elements on manuscript images with their transcription. Such annotations are necessary to train and evaluate recognition systems. In order to reduce the cost of manual annotation, a semi-automatic ground truth creation procedure has been developed that allowed to create the IAM-HistDB with little human effort.

For automatic transcription, single word recognition as well as text line recognition with statistical language models are investigated. Compared with benchmarks for modern handwriting, encouraging results are reported with respect to content-based indexing of historical documents in digital libraries, especially for the Carolingian script of the Saint Gall database and the Gothic script of the Parzival database. General improvements of the recognition systems include a novel algorithm for NN-based text line recognition, the application of linear and non-linear feature selection, and the introduction of novel structural features based on handwriting graphs.

The proposed structural features are obtained by dissimilarity space embedding of handwriting graphs with respect to a set of automatically selected character prototypes. By means of this embedding, a bridge between the high representational power of graphs and the large repository of algorithmic tools in statistical pattern recognition could be established. It allows, in principle, to use arbitrary structural handwriting models in statistical handwriting recognition.

Keyword spotting is a promising alternative to automatic transcription for identifying search terms in historical documents, especially if no reliable lexicon and language model are available. In this thesis, we contribute two novel algorithms for keyword spotting based on character models, one using HMM and the other using NN. They are magnitudes faster than lexicon-based transcription and provide promising spotting results for historical as well as for modern documents. We show that the proposed sub-word model approach outperforms classical word template matching. For the George Washington database, we are able to top the best spotting results reported in the literature so far.

Finally, we have investigated a novel alignment problem which aims at annotating manuscript images fully automatically based on inaccurate transcriptions. By making use of electronic text editions of historical documents created by human experts, training samples for handwriting recognition can be extracted with almost no human effort. By considering inaccurate text editions, we enlarge the scope of admissible texts since human experts tend to write out abbreviations, correct mistakes, insert punctuation marks, change the capitalization, or even rephrase whole sentences for better readability. Two HMM-based alignment systems are introduced and a semi-supervised scenario is outlined that has great potential to initialize handwriting recognition systems at low cost for mass digitization of historical documents.

Acknowledgments

First of all I would like to express my gratitude to Prof. Em. Dr. Horst Bunke who has guided me through several theses and provided me with his priceless knowledge, experience, and wisdom in research.

I would also like to thank Prof. Dr. Rolf Ingold for acting as co-referee and for his support in the HisDoc project. This synergy project of the Swiss National Science Foundation (CRSI22_125220) has brought together three research groups and I further thank Prof. Dr. Jacques Savoy, Nada Naji, and Micheal Baechler for the successful collaboration. Special thanks go to Susanne Thüler for her careful administrative support.

Many people have contributed to the database of historical manuscripts used in this thesis. I would like to thank Prof. Dr. Michael Stolz and Dr. Gabriel Viehhauser for providing the Parzival transcriptions, Prof. Dr. Ernst Tremp for his kind permission to include the manuscript images in the online database, Max Bänziger for providing the aligned Patrologia Latina edition, Markus Wüthrich for the first stock of data, and Amrei Schroettke, Chatrina Casutt, and Matthias Zaugg for their patient annotation work.

I am very grateful for countless discussions, disputes, whiteboard art, and coffee breaks with my colleagues who contributed invaluable parts to this thesis. Many thanks to Dr. Volkmar Frinken, Emanuel Indermühle, Dr. Kaspar Riesen, Dr. Alicia Fórnes, Dr. Marcus Liwicki, Angelika Garz, Samuel Schürch, Andreas Keller, Jürg Nietlispach, Elias Gerber, Gabe Jackson, Anna Schmassmann, Christine Müller, and Nicolas Lenz.

Very special thanks go to my family Thomas Fischer, Martin Fischer, and Silvia Meier who supported me in each and every way.

Finally, I thank my partner Sara Grimm for all her love and inspiration.

Contents

Chapter 1

Introduction

1.1 Motivation

In order to preserve the cultural heritage represented by handwritten historical manuscripts, libraries all around the world digitize large numbers of documents. Storing scanned manuscript images saves the world's cultural heritage from being lost due to the degradation of paper and parchment. After digitization, however, the libraries are facing a challenging and yet unsolved problem: to make the textual content of millions of document images readily accessible to researchers and the public.

Such an access could be established in digital libraries with search engines similar to the ones on the Internet. To that end the historical scripts have to be transcribed into machine-readable text. Manual transcription is not feasible in reasonable time considering the large amount of document images. Instead, there is a growing interest in pattern recognition methods that allow for automatic handwriting recognition [6].

This thesis addresses the open question how current handwriting recognition systems perform in the case of historical documents. The goal is the creation of reading machines that can generate electronic manuscript transcriptions fully automatically. While this ultimate goal is far from reached by state-of-the-art systems, intermediate goals, which seem more feasible today, include the support of human experts with interactive systems, the identification of specific keywords within manuscripts, and the alignment of existing electronic texts with manuscript images. Recognition methods that can provide satisfactory solutions to these problems are promising candidates for integrating current research systems into industrial systems. They also are considered in this thesis.

In the following, handwriting recognition in historical documents and the aims of this thesis are introduced in detail. First, the basic concepts underlying machine reading are presented in Section 1.2. Especially for readers that are not familiar with pattern recognition in general or with

handwriting recognition in particular, some of the main terminology in pattern recognition is introduced in the context of recognizing handwritten text. Afterwards, the general state-of-the-art in modern handwriting recognition is discussed in Section 1.3. Next, the current challenges and techniques that are specific to historical handwritings are elaborated in Section 1.4. Based on this background the scope, aims, and contributions of this thesis are formulated in Section 1.5. Finally, an outline of the thesis is given in Section 1.6.

1.2 Machine reading

For more than half a century, automatic handwriting recognition has been an intriguing and still largely unsolved problem in computer science [32, 129, 144]. The goal to realize machine reading systems that can match or even surpass the impressive ability of humans to read handwritten text is still far from being reached.

Deciphering handwritings can also be very difficult for humans, for instance when it comes to reading handwritten notes taken years ago, hastily written shopping lists, and notoriously illegible medical prescriptions. When confronted with unknown languages, reading letter by letter is highly demanding and prone to errors even if the letters are known. Mastering historical scripts and languages, in particular, may require years of intensive study, not only in linguistics but also in history to get an understanding of the general context which is often required to decipher unreadable parts of a manuscript.

In the context of artificial intelligence, pattern recognition methods for handwriting recognition aim at mimicking human perception and intelligence in order to realize machine reading. Most importantly, the decision rules that identify the patterns of interest, i.e., characters, words, and whole sentences, are not manually defined in some algorithm. Instead, they are *learned from training samples* given by handwritten texts together with their machine-readable transcription. In analogy to human learning, decision rules are inferred from such training samples in order to recognize other handwritten texts automatically afterwards.

Following the learning paradigm has two advantages. First, it avoids the problem of manually defining decision rules which seems hardly feasible considering the complexity of the task. Second, it can potentially provide very general machine reading systems that can be used with any script and language provided that training samples are available.

In the following, machine learning is introduced in a more formal way. The representation of textual patterns is discussed in Section 1.2.1 and machine learning methods are addressed in Section 1.2.2. While (almost) no mathematical formulas are employed, the vocabulary of the field of pattern

recognition is used and references to introductory works are given. Afterwards, Section 1.2.3 explains the fundamental difference of recognizing handwritten text line images compared to individual character and word images. Finally, Section 1.2.4 discusses the integration of language models in addition to text appearance models.

1.2.1 Pattern Representation

In order to process handwritten text algorithmically, it is represented first using mathematical models. In *statistical pattern recognition*, objects are represented by *feature vectors*. For images of handwritten text, typical features include contour position, stroke direction, and center of mass. Due to the rich mathematical structure vector spaces provide, a huge amount of pattern recognition algorithms could be devised over the last decades using statistical object representation [59].

In *structural pattern recognition*, objects are described symbolically with *strings*, *trees*, and *graphs*. Graphs are the most general models that consist of nodes which are linked with edges. The nodes describe subparts of an object while the edges capture binary relationships of the subparts. Both nodes and edges may be labeled with additional information such as symbols and feature vectors. For handwritten text, a natural graph model could represent start and end positions of individual strokes with nodes and link them with edges labeled with the length and curvature of the stroke. Because of its high representational power, a large number of algorithms have been developed for graph-based object representation [49].

1.2.2 Machine Learning

After representing handwritten text images with a suitable model, machine learning is used to infer text appearance models from given writing samples. Depending on the information available for the samples, clustering, supervised learning, and semi-supervised learning can be distinguished.

Clustering When presented with handwriting images only, without knowledge of their textual content, so called *clustering* methods can be applied initially to identify groups of similar patterns [117]. For character images, the resulting *clusters*, i.e., sets of images belonging together, ideally represent the underlying alphabet. For word images, a dictionary would be expected that contains an image cluster for each distinct word within the writing samples. A natural similarity for feature vectors is their proximity in the vector space. Similarity measures can also be defined in various ways for graphs such that clustering can be performed directly in the graph domain [255].

Clusters of similar patterns provide very intuitive models of *pattern classes*, for instance the class of the letter *A* or the class of the word *Parzival*. Clusters that are labeled with their class can already be used for *classification* of new handwriting images. Using a classical nearest neighbor rule [52], arbitrary handwriting images are assigned to the class whose cluster contains the most similar image. Again, this classification rule is only dependent on some suitable definition of similarity which is feasible for both feature vectors and graphs.

Supervised Learning As soon as class labels are available for some handwriting samples, more sophisticated class models can be inferred by means of *supervised learning*. Supervision refers to the presence of class labels that need to be provided by a "teacher", e.g., by manually labeling samples. In contrast, clustering is applied to unlabeled patterns and is therefore called *unsupervised learning*. Most supervised techniques rely on statistical pattern representation, where each pattern is represented by a single feature vector in a metric vector space. In this case, class models can be seen as a partition of the vector space into distinct regions for each class, i.e., all patterns within a region are assigned to the class of that region.

Following the *discriminative* approach, the class boundaries of the vector space partition are modeled directly, e.g., by means of hyperplanes. Learning, in that case, consists of optimizing the parameters of the boundaries, e.g., the position and orientation of hyperplanes, such that the risk of misclassification is minimized. During *training*, the parameters are fitted to a *training set* of labeled pattern samples. However, especially if the training set is small and not representative with respect to the whole pattern space, *overfitting* might occur. This leads to poor *generalization* performance when classifying other patterns than the training samples. For handwriting recognition, consider for example a single training image of the number 1 that is written like the number 7. If the class model is fitted too closely to this single training sample it will classify many 7s as 1s afterwards. Often, the problem of overfitting is alleviated by using reasonable heuristics. Two prominent heuristics are establishing boundaries in low density regions in the vector space and using labeled samples of an independent *validation set* to estimate the generalization ability of the class model.

In contrast, the *generative* approach models the distribution of the different pattern classes in the vector space rather than their boundaries. Usually, a parametric density family such as Gaussian mixtures is employed and the corresponding parameters, e.g., means and variances of Gaussians, are fitted to the training set, e.g., using expectation maximization [55]. Taking into account the a priori probability of the classes as well as class specific misclassification risks, the vector space is subdivided into distinct regions for each class such that the Bayes risk of misclassification is minimal [59].

For the task of classification, the generative approach can be used in the same way as the discriminative approach because a vector space partition is established eventually. However, the estimated density distributions provide additional information about the different classes that can be used, e.g., to generate artificial patterns by means of random sampling.

In summary, classification can be condensed into a single formula of the predictor function f_θ

$$f_\theta(x) = y \tag{1.1}$$

which assigns a class label $y \in \mathcal{L}$ to a pattern $x \in \mathcal{P}$ based on class model parameters θ. The model parameters are optimized on labeled training samples with respect to minimizing the risk of misclassification. The label space $\mathcal{L}$ is usually given by a finite set of symbols. The pattern space $\mathcal{P}$ typically is $\mathbb{R}^n$ for statistical representation and the graph domain for structural representation. For statistical representation, the predictor function corresponds with a vector space partition that is either established in a discriminative fashion by means of parametrized class boundaries or in a generative fashion by means of parametrized class densities.

An algorithmic implementation of the predictor function is, in essence, the recognition system, which is also called *classifier* or *recognizer*. For performance evaluation, the classifier is applied to an independent *test set* of samples that were not used during training. Corresponding classification *accuracies* and *error rates* estimate how well the trained recognizer will perform in real-world applications. For accurate estimates of the performance, the test set should, of course, be representative with respect to the envisaged real-world scenario.

Semi-Supervised Learning Apart from supervised and unsupervised learning, an intermediate form of learning, called *semi-supervised learning*, can be distinguished [41]. In addition to labeled training samples, this form of learning makes use of unlabeled samples in order to improve the trained class models, e.g., by moving class boundaries into low density areas or by improving class density estimations. An analogy to human learning is given by a child that first learns from a teacher how handwritten numbers look like and afterwards looks at other handwritings to refine its idea of a number without help from a teacher. For handwriting recognition, this form of learning is appealing, in particular, because the manual effort to create labeled writing samples is high while unlabeled images of handwritings are abundantly and readily available [82].

1.2.3 Sayre's Paradox

So far, a scenario was considered where each pattern is represented by a single feature vector or graph. This approach is suitable, e.g., for printed

documents that allow for segmenting text images into individual characters before recognition. Taking into account known fonts for printed documents, optical character recognition (OCR) can be realized with near-perfect recognition accuracy for modern fonts.

For handwritten text, however, standard OCR methods are not applicable since a recognition-free segmentation of text images into characters is usually not feasible prior to recognition. Difficulties arise for touching characters and broken characters. Also, there is no "font" for handwritten characters whose shapes can differ greatly even for a single writer. Character recognition is needed to segment the text reliably. That is, text segmentation into characters requires recognition and recognition requires segmentation. Also known as *Sayre's paradox* [205], this contradiction demands for more sophisticated recognition systems that perform segmentation and recognition at the same time.

Instead of segmenting handwritten text images into characters, an over-segmentation into parts of characters is typically considered [36], resulting in a sequence of small text images. Correspondingly, the predictor function

$$f_\theta(\mathbf{x}) = \mathbf{y} \tag{1.2}$$

maps a sequence of feature vectors or graphs $\mathbf{x} = x_1, \ldots, x_N$[1] with length N to a sequence of character labels $\mathbf{y} = y_1, \ldots, y_M$ with length $M \leq N$, where $x_i \in \mathcal{P}$ for $1 \leq i \leq N$ and $y_j \in \mathcal{L}$ for $1 \leq j \leq M$. During recognition, several consecutive character parts can be assigned to the same character label providing, in result, not only a recognized sequence of character symbols, but also a segmentation of the handwriting image into characters.

Only a few recognition systems exist that are able to cope with sequences. A well-established generative approach is given by Hidden Markov Models (HMM) [177]. HMM are also widely used in the related field of speech recognition [179]. Very recently, a discriminative approach based on Neural Networks (NN) [96] could successfully be applied to handwriting recognition. Both HMM and NN based recognizers are *segmentation-free* in the sense that they do not depend on character segmentation prior to recognition. Instead, they apply over-segmentation and assign the character parts to character labels as an integral part of training and recognition.

Unlike characters, complete handwritten words can be segmented prior to recognition with high accuracy in many cases. Hence, classical recognition can be performed with respect to Equation 1.1. However, when considering large vocabularies, obtaining a large number of training samples for each word is a major obstacle. Working with sub-word models such as characters, instead, has two advantages. First, the number of class models that need to be trained is substantially smaller[2]. Since characters are shared

[1] In this thesis, the bold typeface refers to sequences.

[2] Considering a word with four letters and an alphabet with 26 letters, 26^4 word models

among different words, more training instances per model are available in a given handwritten text. Secondly, words can be recognized even if they do not appear in the training set by concatenating trained character models accordingly.

1.2.4 Language Models

Character models discussed so far are *appearance models* of handwritten text reflecting the different writing styles of individual characters. However, when relying on appearance models only, the resulting sequence of characters is prone to errors. For instance, invalid words can be returned that are not part of the language under consideration. Hence, *language models* are additionally taken into account in order to constrain the recognition process. In its basic form, recognition is limited to a sequence of words from a lexicon of valid words.

The implicit assumption of this basic approach is that all words sequences have same probability. Of course, this assumption does not hold true for natural language where words do not occur arbitrarily in a text. *Statistical language models* aim at predicting the probability of a word based on its previously recognized word history. For instance, after the word "Los", "Angeles" is more probable than "Francisco". The corresponding word sequence probabilities are estimated from large text corpora such as electronic newspapers and books in case of modern languages. The use of statistical language models has proven to increase the recognition accuracy significantly and they have become a standard component of modern reading machines [153].

Only very few attempts have been made so far to extend the statistical language modeling approach, e.g., by integrating stochastic context-free grammars (SCFG) based on grammatical word tags such as noun, pronoun, and verb form [261]. The additional gain in terms of recognition accuracy is still very limited.

The ability of humans to truly *understand* handwritten text is beyond the reach of current machine reading systems and in difficult cases, where deep context understanding is required to decipher an illegible word, they are expected to fail. Instead, an intermediate goal is to provide accurate results for cleanly written text, which would already allow an automatic access to a large number of handwritten documents.

---

would need to be taken into account when disregarding dictionary limitations. Exponentially fewer character models can be used instead.

1.3 Modern Handwriting

Before handwriting recognition in historical documents is addressed, this section reviews the state of the art in modern handwriting recognition. First, the capabilities and limits of current recognition systems are discussed in Section 1.3.1. Afterwards, the process of learning-based handwriting recognition is detailed in Section 1.3.2.

1.3.1 State of the Art

In case of printed documents, commercial systems for optical character recognition (OCR) with near-perfect recognition accuracy exist. For handwritten documents, however, standard OCR methods are not applicable because a recognition-free segmentation of text images into characters is usually not feasible (see Section 1.2.3). Besides the segmentation problem, the lack of a "font" for handwriting renders character modeling more difficult since the character shapes can differ greatly even for a single writer.

Commercial handwriting recognition systems that achieve near-perfect recognition performance are currently only available for restricted tasks with small vocabularies such as bank check reading [94, 114] and postal address reading [221, 64]. In these scenarios, the number of words is limited and redundancies between handwritten numbers and words can be used to achieve near-perfect results. Also, there are commercial products available for *on-line recognition* [176], where temporal information is given about the writing process based on special input devices, for instance for Personal Digital Assistants (PDAs). In contrast, *off-line recognition* is solely based on handwriting images. Because of the lack of temporal information it is a harder task than on-line recognition.

For large vocabularies underlying natural language, the recognition accuracy for off-line handwritten text is still far from being perfect, although strong progress could be observed in the past two decades [245]. This progress can be illustrated for the IAM database [154][3], which provides an established benchmark data set for off-line Roman cursive handwriting recognition. Over 600 writers have contributed to this data set that consists of over 13,000 handwritten text lines from the LOB corpus [120] for British English. Shortly after the introduction of the IAM database ten years ago, a word recognition accuracy of about 60% was reported [153]. Meanwhile, the best reported word recognition accuracy for the IAM database is about 74% [65].

For script specific surveys on handwriting recognition, see for instance [32] for Roman scripts, [145] for Arabic scripts, and [222] for Chinese scripts.

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

1.3.2 Recognition Process

The procedure of learning-based handwriting recognition consists of two consecutive steps, namely training and recognition. At training stage, character appearance models and language models are learned from training samples. At recognition stage, scanned documents are segmented into text images which are then transcribed into machine-readable text. The two stages are described in the following. For a survey on the whole document analysis process, we refer to [162].

Training The current state of the art relies on two sources of information to train recognition systems for unconstrained modern handwritings.

1. Text images labeled with their machine-readable transcription.
2. Electronic text corpora.

Labeled text images are used to train character appearance models (see Section 1.2.2). For segmentation-free recognition, text lines can be considered that are labeled with a sequence of characters (see Section 1.2.3). Such training samples can be obtained efficiently for modern handwriting by means of special forms that are filled in by contemporary writers. For instance, the writers can be asked to copy given texts into individual boxes for text lines which can be retrieved easily after scanning the forms.

Text corpora are given by electronic media such as electronic newspapers and books, which are abundantly available for modern languages. From these texts, lexicons of valid words can be collected and statistical language models can be estimated (see Section 1.2.4).

Recognition After scanning handwritten documents, text images need to be extracted in a first step. This includes layout analysis, text and non-text separation, and text line segmentation. Although word segmentation is feasible in many cases it is not necessary for state-of-the-art recognition systems that are able to process whole text lines.

In a second step, the text lines are recognized based on the trained character appearance models and the language model. The result is a sequence of lexicon words in a machine-readable form, for instance in ASCII characters.

1.4 Historical Handwriting

As for modern documents, the recognition of printed historical documents is less challenging than the recognition of handwritten historical documents. Yet there are currently no commercial solutions available for mass digitization of historical prints. Compared with modern prints additional difficulties

include special fonts and bad image conditions due to the partial decay of paper. There are, however, ongoing projects such as IMPACT[4] that aim at making historical prints accessible in digital libraries in the near future.

The mass digitization of historical handwritings, on the other hand, can be seen as one of the hardest problems in handwriting recognition today. While modern handwriting recognition includes a rather narrow space in time, the variety of layouts, scripts, and languages in historical documents is immense[5]. With respect to the recognition process, there are two main challenges.

1. The extraction of text images from scanned manuscript pages.
2. The acquisition of training samples for handwriting recognition.

The extraction of text images is rendered difficult by a large variety of layouts and bad image conditions stemming for instance from damaged parchment, faded ink, and ink bleed-through. The acquisition of training samples cannot be done by means of special forms but is based on actual manuscripts instead. With respect to the challenge of text extraction this can often only be achieved in a semi-automatic fashion with considerable human effort. Furthermore, special historical scripts and languages cannot be transcribed by laypersons. Expert knowledge is required instead. Even in case of an existing electronic text, an explicit alignment between the text images and the electronic text has to be performed. As for language modeling, large electronic text corpora with a consistent orthography are often unavailable for historical languages.

Despite the discouraging conditions for learning-based handwriting recognition, the research activity in this field has grown strongly in the past decade and there are promising reports in the literature that demonstrate how current pattern recognition methods can be effectively applied for handwriting recognition in historical documents[6].

In the following, we discuss the state of the art in historical handwriting recognition in more detail. The problem of text extraction is addressed in Section 1.4.1, the acquisition of training samples in Section 1.4.2, and handwriting recognition in Section 1.4.3.

1.4.1 Text Extraction

For text extraction, typical challenges include special layouts, ornaments, colored initial letters, faded ink, ink bleed-through, ink stains, partial decay

[4] `http://www.impact-project.eu/`

[5] Even the range of writing supports is wide, including for instance stone [151] and palm leafs [38].

[6] The first international workshop specifically dedicated to historical document analysis was held 2011 in Beijing and has instantly attracted a large number of researchers. For more information, see `http://www.comp.nus.edu.sg/~hdocp/`.

of parchment, and distorted parchment. First approaches towards automatic layout analysis in historical documents include [9, 89]. An extensive survey on text line segmentation methods can be found in [140]. Also, recognition-free word segmentation has been attempted as described, e.g., in [150].

An important aspect of text extraction in general is the separation of text foreground from non-text elements such as ornaments, but also from visible structure of the writing support such as wrinkles and stains. In [224], different algorithms for *binarization*, i.e., an explicit image separation into text foreground and background, are compared for historical documents.

1.4.2 Ground Truth Creation

In order to create training samples for handwriting recognition, text images have to be extracted from scanned manuscript pages and the images have to be annotated with their transcription. Such annotations or labels are also called the *ground truth* of a pattern recognition problem. The ground truth is usually created with human supervision to ensure that clean, error-free training samples are obtained. Besides its use to train recognizers, ground truth is also needed for evaluating the performance of a recognizer.

Several *interactive systems* have been proposed in the literature that aim at reducing the human effort for annotating historical manuscripts. Examples include the DEBORA system used for Renaissance documents [24], the DMOS annotation system used for old civil status registers and military forms [50], and the STATE [93] and CATTI [191] systems used for old Spanish manuscripts. There is a trend towards multimodal interaction that facilitates human computer interaction by using touch screen devices and allowing for handwritten input using special pens [236]. Also, handwriting recognition systems can be employed as soon as the first training samples become available. They can support manual transcription by suggesting recognition results to the user.

If a transcription is already available, automatic *transcription alignment* with text images can be used for ground truth creation. The typical scenario takes line transcriptions into account that are aligned in order to segment and label individual words within text line images. In some cases, an accurate alignment is feasible without handwriting recognition, for instance by means of gap metrics [223]. In more difficult cases, handwriting recognition can be applied [115, 233].

The creation of data sets for historical document analysis requires much human effort and there are only very few data sets publicly available for research today[7]. One of the first data sets for handwriting recognition in historical documents was the George Washington database [134, 183] containing 20 pages of George Washington's letters. This database has already

[7]Unfortunately, copyright constraints can also prohibit the free distribution of research data in this field.

been used by several research groups and can be seen as a benchmark data set for handwriting recognition, in particular for keyword spotting (see Section 1.4.3). With the development of interactive annotation systems, more databases are being currently created that will hopefully provide a larger scope of manuscripts for benchmarking in the future. A promising data repository in this context are the GIDOC[8] data sets [171, 215], which were made freely available on the Internet, very recently, and provide transcriptions of several hundred old Spanish manuscripts.

1.4.3 Handwriting Recognition

Besides the lack of training data, some further difficulties have to be considered for historical handwriting recognition. Character appearance models have to take an additional variability into account that stems from bad image conditions. For instance, the quality of text foreground detection can differ greatly even within a single manuscript. Furthermore, language models cannot easily be transferred to different manuscripts in case of non-standardized orthography and special character symbols. On the other hand, historical scripts often show a regular, neat writing style. In fact, some medieval manuscripts almost look printed. In this case, handwriting recognition is expected to achieve high recognition accuracy even for a limited amount of training samples.

Due to the difficulties in text extraction and acquisition of training data, only few attempts towards *automatic transcription* of historical documents have been made so far. Yet there are some promising reports. The advantage of the regular writing style in a medieval Latin manuscript was used in [61] by means of character template images. A small amount of manually extracted character images could successfully be used for character modeling with HMM. In [167], character cavities were categorized for segmentation-free recognition of old Greek manuscripts. However, this approach is specifically designed for the old Greek script. A more general character-based approach was presented in [67] where so-called alphabet soups were used in conjunction with HMM in order to transcribe the George Washington data set. On the same data set, word modeling with HMM was pursued in [134] and recognition with conditional random fields has been attempted in [69]. Fully-fledged HMM-based transcription including language modeling has been used for recognizing old Spanish manuscripts in the context of interactive transcription in [236].

Automatic transcription assumes the availability of training samples and a lexicon of valid words. This condition is rather untypical for historical handwriting recognition. Even if a few training samples are available, a reliable lexicon might not be easy to obtain for special historical languages. In

[8] `http://prhlt.iti.upv.es/page/projects/multimodal/idoc/`

this scenario, *keyword spotting* has been proposed in [149] as a promising alternative to automatic transcription for the identification of search terms in historical manuscripts. In [149], template images of a given search term are matched against segmented word images in the manuscript. The most similar words, or manuscripts that contain the words, are then returned sorted by similarity. Based on different statistical features and similarity measures, various variants have been developed, for instance in [139, 183, 195]. An advantage of this approach is that template images are easily accessible by means of clustering (see Section 1.2.2) and no character modeling is needed based on transcribed training samples. However, by using template word images the approach is limited to search terms for which at least one template is available, usually in the same manuscript that is being searched.

1.5 Problem Statement

This thesis investigates pattern recognition methods for handwriting recognition in historical documents. Four subproblems are addressed, namely ground truth creation, automatic transcription, keyword spotting, and transcription alignment (see Section 1.4).

In the following, the scope of this thesis is discussed in Section 1.5.1 and general recognition methods are commented in Section 1.5.2. Finally, Section 1.5.3 lists the contributions of this thesis.

1.5.1 Scope

Experimental evaluation is performed with respect to three historical document collections. The oldest manuscript originates in the 9th century and contains Latin texts written in Carolingian minuscules. The second is a 13th century manuscript written in a medieval German language with Gothic minuscules. Finally, George Washington's letters from the 18th century are considered, written in longhand. Overall, the experimental evaluation is constrained to the Roman alphabet. Yet the character shapes as well as the text extraction conditions differ greatly among the three document collections as illustrated in Figures 2.1– 2.3 in Chapter 2. In addition to the historical manuscripts, the IAM database [154] is used for comparison with modern English handwriting. Samples from this database are shown in Figure 2.4.

In order to focus on handwriting recognition, text extraction is not covered in depth. It is addressed only in the context of ground truth creation. Otherwise, experimental evaluations are performed on manually corrected text segmentations. Therefore, the reported results should be regarded as upper bounds. In real-world applications, any error in text extraction can lead to additional errors in handwriting recognition.

1.5.2 Methods

We pursue two state-of-the-art strategies to model and recognize characters, words, and sentences. First, a generative strategy using hidden Markov models (HMM) [177] and secondly, a discriminative strategy using a special form of recurrent neural networks (NN) [96]. The recognition systems are segmentation-free in the sense that no segmentation of text line images into words and characters is required prior to training and recognition.

1.5.3 Contributions

The main contributions of this thesis are listed in the following. The presented work has been published in large parts in various international conference papers and journals, which are referenced here.

1. Ground truth creation: By means of an efficient semi-automatic procedure [73] the IAM-HistDB was created that includes Carolingian Latin texts from the 9th century (Saint Gall database), Gothic German texts from the 13th century (Parzival database), and longhand English texts from the 18th century (George Washington database). They were made publicly available on the Internet[9].

2. Automatic transcription: Two state-of-the-art systems were applied to historical script recognition, one based on HMM [79, 254] and one based on NN [79]. General improvements were achieved with feature selection [70] and a new algorithm for NN-based recognition.

3. Keyword spotting: Two novel keyword spotting systems are proposed, one based on HMM [75, 76] and one based on NN [83, 86]. The new sub-word model approach has proven to outperform classical template matching.

4. Transcription alignment: A new HMM-based alignment between text editions and manuscript images is proposed that can be used for creating a handwriting recognition database automatically [72, 74].

5. Graph-based handwriting recognition: A novel framework is proposed that allows to use handwriting graphs in conjunction with statistical recognition [71, 78].

Not included in this thesis are related collaborative works on semi-supervised learning [84], keyword spotting adaptation from modern to historical documents [85], and template-based keyword spotting using blurred shape models [80].

[9] `http://www.iam.unibe.ch/fki/databases/iam-historical-document-database`

1.6 Outline

The remainder of this thesis is organized as follows. First, the data sets under consideration are introduced in Chapter 2. For each data set, its resources, contents, and characteristics are discussed, some sample images are shown, and key references are given.

Next, Part I presents methods for handwriting recognition in general. Statistical representation is discussed in Chapter 3, structural representation in Chapter 4, HMM-based recognition in Chapter 5, and NN-based recognition in Chapter 6. Mostly, these chapters contain reviews of state-of-the-art methods and promising trends. New methods include graph-based handwriting recognition in Chapter 4 and a new algorithm for NN-based recognition in Chapter 6.

Afterwards, Part II is dedicated to historical manuscript recognition. Ground truth creation is presented in Chapter 7, automatic transcription in Chapter 8, keyword spotting in Chapter 9, and transcription alignment in Chapter 10. New methods as well as experimental evaluations are provided.

Finally, conclusions are drawn in Chapter 11 alongside with a discussion of future research directions.

Chapter 2

Data Sets

In this chapter, the handwriting data sets considered throughout this thesis are introduced. Three novel historical data sets include the Saint Gall database (SGDB), the Parzival database (PARDB), and the George Washington database (GWDB). For comparison with modern handwriting, the well-established IAM database (IAMDB) is taken into account as well for experimental evaluation.

The three historical data sets are part of the IAM historical document database (IAM-HistDB), which contains several hundred labeled manuscript pages. The SGDB, PARDB, and GWDB, including over hundred manuscript pages altogether, cover the parts that were already used for experimental evaluation and have been made publicly available online at `http://www.iam.unibe.ch/fki/databases/iam-historical-document-database`.

While the PARDB is completely new, the document collections underlying the SGDB and GWDB have been used before with a different ground truth. The 20 page images of George Washington's letters underlying the GWDB, in particular, were used in various articles and conference papers on handwriting recognition by several research groups and allow for comparison with other results reported in the literature.

The four data sets are illustrated in Figures 2.1– 2.4. For each database, an example page is displayed alongside with different writing styles, text extraction challenges, and special characters typically encountered in the manuscripts. The figures are commented in the corresponding database sections.

The main characteristics of the data sets are summarized in Tables 2.1– 2.3. Note that since the IAM-HistDB is a work in progress that includes error corrections in the manuscript annotations, the text statistics indicated in earlier publications may differ slightly from Table 2.3, which corresponds to the version 1.0 of the SGDB, PARDB, and GWDB published online. For the IAMDB, version 3.0 is taken into account.

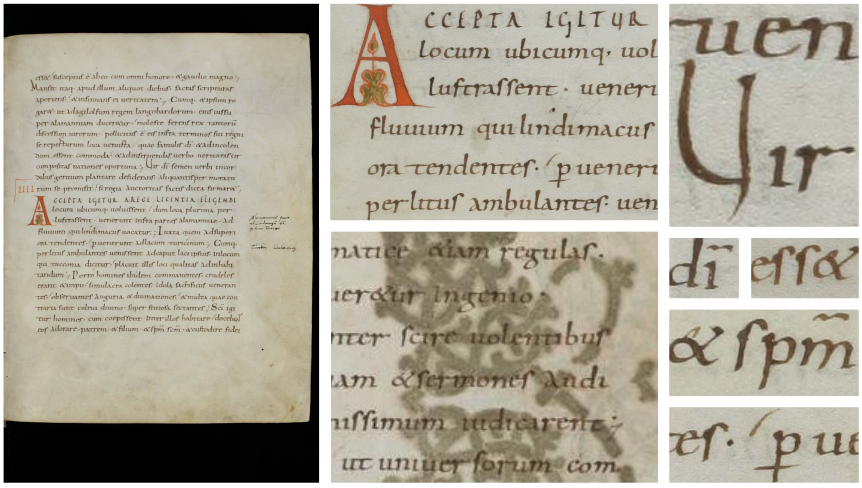


Figure 2.1: Saint Gall Database (SGDB)

Figure 2.2: Parzival Database (PARDB)

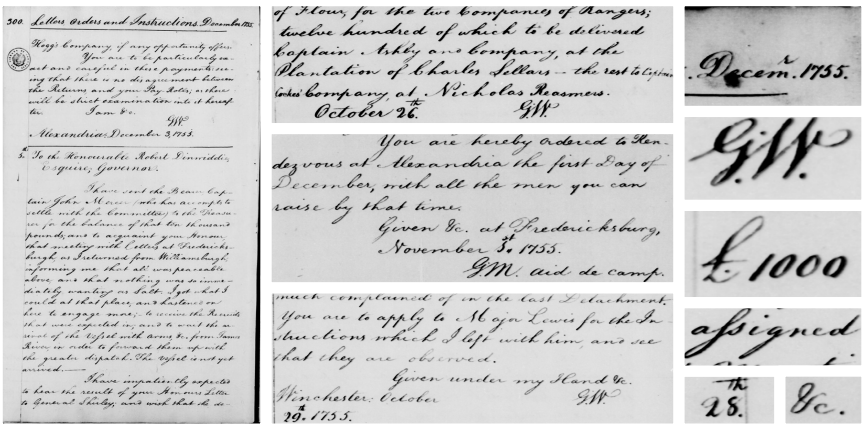


Figure 2.3: George Washington Database (GWDB)

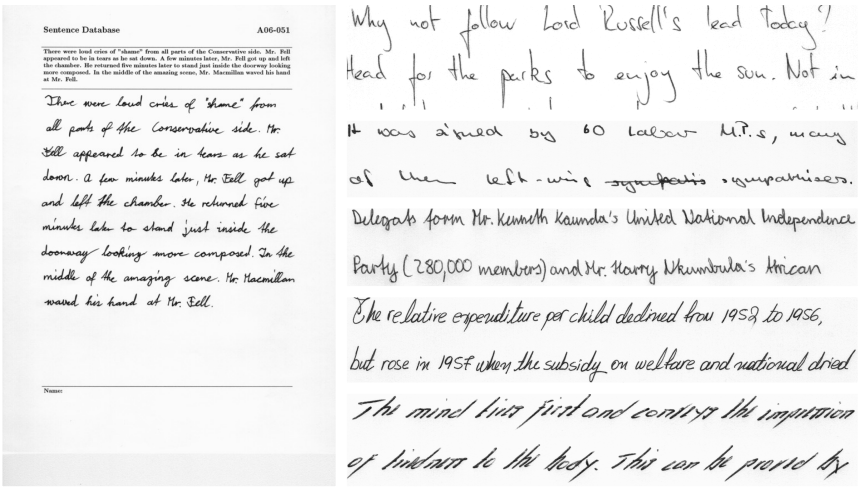


Figure 2.4: IAM Database (IAMDB)

Table 2.1: Data sets.

Database	Century	Language	Writers	Pages
Saint Gall (SGDB)	9th	Latin	1	60
Parzival (PARDB)	13th	German	3	47
George Washington (GWDB)	18th	English	2	20
IAM (IAMDB)	20th	English	657	1539

Table 2.2: Page formats.

Database	Physical Format	Digital Format[†]
SGDB	24.0 x 30.0 cm, ink on parchment	3328 x 4993 px, Color JPEG
PARDB	21.5 x 31.5 cm, ink on parchment	2000 x 3008 px, Color JPEG
GWDB	Ink on paper	2034 x 3286 px, Grayscale JPEG
IAMDB	21.0 x 29.7 cm, pen on paper	2479 x 3542 px, Grayscale PNG

[†]Including digitization borders.

Table 2.3: Text statistics.

Database	Text Lines	Word Instances	Word Classes	Characters
SGDB	1410	11597	4890	49
PARDB	4477	23478	4934	93
GWDB	656	4894	1471	82
IAMDB	13353	115320	13542	78

2.1 Saint Gall Database

2.1.1 Resources

The Saint Gall database (SGDB) is based on a medieval Latin manuscript from the 9th century that contains the hagiography *Vita sancti Galli* by Walafrid Strabo. In his work, Strabo describes the life of Saint Gall who gave his name to the Abbey of Saint Gall, Switzerland. The Abbey Library holds a manual copy of the work within the Cod. Sang. 562, which was written by a (probably) single experienced hand in Carolingian script with ink on parchment.

The digital images of the manuscript were made available online by the e-codices project[1], a virtual manuscript library from the Medieval Institute of the University of Fribourg, Switzerland. The documents were digitized with a resolution of 300dpi and are available as high quality JPEG images. A text edition of the manuscript was attached at page level to the e-codices images by the affiliated Monumenta project[2] based on the Patrologia Latina edition[3].

2.1.2 Contents

The SGDB currently includes 60 manuscript pages[4] as indicated in Table 2.1. Both images and text edition were obtained from the Monumenta webpage and have been processed in several steps into a handwriting recognition database (see Chapter 7). The version 1.0 of the SGDB consists of:

- Colored page images together with text line locations.
- Binary, normalized text line images together with their transcriptions, word locations, and word labels from the text edition.

Statistics of the textual content are given in Table 2.3. Note that the transcriptions, i.e., the actual word spellings in the handwriting, deviate from the word labels given in the text edition, e.g., in case of abbreviations. Text line binarization was performed to separate the text foreground from the parchment background and normalization is an image preprocessing step that aims at standardizing the handwriting appearance (see Chapter 3).

2.1.3 Characteristics

Typical properties of the SGDB for handwriting recognition are illustrated in Figure 2.1. In general, the layout is very regular. Each page is written

[1] `http://www.e-codices.unifr.ch`
[2] `http://www.monumenta.ch`
[3] J.-P. Migne PL114, 1852
[4] Cod. Sang. 562, pages 3–50 and 54–65.

in a single column that contains 24 text lines. Text line separation is eased by linear skew, i.e., text line inclination, as well as large spacing between consecutive lines. Furthermore, the text foreground is clearly distinguishable from the parchment background. Finally, only a small number of 49 characters have to be modeled. Besides 25 lower case letters and 22 upper case letters, the special letter "&" and the punctuation character "." are used in the database. Carolingian minuscules are predominant (about 90%), while upper case letters are only used to emphasize the structure of the text.

However, there are several challenges for machine reading when working with the SGDB. Automatic layout analysis has to take marginal notes on the border of the page into account, which were added later to the manuscript and are not part of the main text. Also, colored initial letters have to be separated from the main text body (Figure 2.1, left column). For text line extraction, small holes in the parchment and ink bleed-through render the task more difficult. An example of ink bleed-through is shown that is caused by a large ornament on the back page that interferes with the handwriting on the front page (second column, bottom). Furthermore, text lines can overlap in case of large ascenders and descenders (third column, top).

For character and word modeling, abbreviations have to be taken into account. Samples are shown for "dei" and "esset" (third column, middle)[5]. Furthermore, word breaks at the end of a line, which occur for about a third of all lines, are not specially marked. Instead, the words are directly continued on the next line. Finally, word spacings are frequently omitted (right column, second image from bottom where "&" is an individual word). In other cases, dots and strokes are inserted between two consecutive words in order to improve separability (right column, bottom), yet they are not necessarily punctuation marks with respect to the sentences.

2.1.4 Key References

The page images underlying the SGDB have already been used in [9] for the task of layout modeling. We have presented the SGDB in [72] in the context of transcription alignment (see Chapter 10) and have published it online at the same time as part of the IAM-HistDB.

2.2 Parzival Database

2.2.1 Resources

The Parzival database (PARDB) is based on a medieval German manuscript from the 13th century that contains the epic poem *Parzival* by Wolfram von Eschenbach, one of the most significant epics of the European Middle Ages.

[5]Note that special abbreviation marks next to the letters are not annotated in the SGDB. Instead, the ground truth only contains letter labels.

There exist several manual copies of the poem that differ in writing style and dialect of the language. For the PARDB, the Cod. 857 is considered, which is held by the Abbey Library of Saint Gall, Switzerland. It was written in Middle High German by several writers with ink on parchment using Gothic minuscules.

The manuscript was digitized with a resolution higher than 200 dpi by the German Language Institute of the University of Bern, Switzerland. Based on the TUSTEP software[6] for managing transcriptions of Latin and non-Latin manuscripts, an electronic manuscript edition was created and published on CD-ROM[7]. The actual word spelling in the handwriting is represented in the electronic edition by special character symbols, e.g., in case of abbreviations.

2.2.2 Contents

The PARDB currently consists of 47 manuscript pages[8] including three writers as indicated in Table 2.1. Based on the electronic manuscript edition, the version 1.0 of the PARDB includes:

- Colored page images.
- Binary, normalized text line images together with their transcriptions.
- Binary, normalized word images together with their transcriptions.

Statistics of the textual content are given in Table 2.3. Note that for the PARDB, the ground truth creation procedure presented in Chapter 7 was not yet available and the text locations within the images were, unfortunately, not recorded[9]. For the ease of online distribution, the original page images, which were used to extract text line and word images, have been compressed to high quality JPEG.

2.2.3 Characteristics

A visual impression of the PARDB is given in Figure 2.2. The poem is written in two columns of pairwise rhyming lines. Besides the main text body, it contains ornaments, richly colored initial letters, and marginal notes on the page borders that were added later to the manuscript (Figure 2.2, left column). Writing samples are given for each of the three writers (second column, from top to bottom). The handwriting of the first writer appears on the first four pages of the PARDB (Cod. 857, pages 6–9), followed by

[6] `http://www.zdv.uni-tuebingen.de/tustep/`

[7] `http://www.parzival.unibe.ch/`

[8] Cod. 857, pages 6–10, 33–39, 43–44, 124–127, 143–144, and 262–288.

[9] For the SGDB, in contrast, such locations allow to change binarization and other text line preprocessing steps before recognition.

the second writer (Cod. 857, parts of page 9) and the third writer, whose handwriting constitutes the large part of the database. While the three writing styles are quite similar, the appearance of the handwriting changes frequently throughout the manuscript due to differences in ink and parchment condition. The first pages of the physical manuscript, in particular, were exposed to the ravages of time resulting in stains and wrinkles on the page images that interfere with the text foreground (second column, top). Also, ink-bleed through from the colored initial letters as well as holes and stitches[10] interfere with the handwriting (third column, first and second image).

The handwriting style is very neat and regular. However, the manuscript is rather densely written such that consecutive characters often overlap and touch each other. While Gothic minuscules are predominant, there are many special characters, e.g., used for abbreviations, that account for the total of 93 characters. Typically, they consist of lowercase letters in combination with special symbols or other lowercase letters written on top (third column, last four images). Unlike the SGDB, such special characters are encoded explicitly in the ground truth of the PARDB.

2.2.4 Key References

We have introduced the PARDB in [79] and have used it afterwards for several handwriting recognition tasks, e.g., automatic transcription [78], keyword spotting [76, 86], and transcription alignment [74]. The PARDB was published online alongside with this thesis as part of the IAM-HistDB.

2.3 George Washington Database

2.3.1 Resources

The George Washington database (GWDB) is based on English letters written by George Washington and his associates in the 18th century with ink on paper. They were digitized and made available online as high quality greyscale JPEG images by the Library of Congress alongside with transcriptions[11].

2.3.2 Contents

The GWDB currently includes 20 page images[12] as indicated in Table 2.1. The corresponding letters were written between October and December 1755

[10] Parchment was expensive in medieval times and hence damages were repaired.

[11] George Washington Papers at the Library of Congress from 1741–1799, Series 2, to be found at `http://memory.loc.gov/ammem/gwhtml/gwseries2.html`

[12] Letterbook 1, pages 270–279 and 300–309

by George Washington and one of his aide-de-camp, George Mercer, during the French and Indian War. Based on the online edition provided by the Library of Congress, the version 1.0 of the GWDB consists of:

- Binary, normalized text line images together with their transcriptions.
- Binary, normalized word images together with their transcriptions.

Statistics of the textual content are given in Table 2.3. As for the PARDB, the GWDB does not contain text locations, since it was generated before the ground truth creation procedure presented in Chapter 7 became available.

2.3.3 Characteristics

Some sample images of the GWDB are shown in Figure 2.3. The layout is somewhat less regular than for the medieval data sets. Besides the main text, the letters contain page numbers, rulers, stamps, and signatures (Figure 2.3, left column). While the writing styles of George Washington (second column, top) and his aide-de-camp (second column, middle) are very similar, there are differences in handwriting appearance due to the ink and paper conditions. Some parts of the text have faded (second column, bottom) and stains interfere with the handwriting in a few cases (right column, top).

The longhand English handwriting is neatly written. Besides punctuation marks, some of the special characters include signatures, numbers, currency symbols, double "s" letters, "th" written on top of numbers, and the frequently used "etc." symbol (right column).

2.3.4 Key References

The 20 pages underlying the GWDB have been used in several studies before, e.g., in [134] for word recognition and in [183] for keyword spotting. The ground truth that was made publicly available by Rath et al. contained automatically determined word locations. In contrast, the GWDB includes ground truth at line level that was used, e.g., in [76, 86] for keyword spotting. The GWDB was published online alongside with this thesis as part of the IAM-HistDB.

2.4 IAM Database

2.4.1 Resources

The IAM database (IAMDB) is based on modern English handwritings that were collected about a decade ago at our institute[13]. The writers were

[13]Institute of Computer Science and Applied Mathematics (IAM), University of Bern, Switzerland.

presented with texts from the Lancaster-Oslo/Bergen corpus (LOB) [120], which contains a variety of printed British English texts published in 1961. Using arbitrary modern pens and special acquisition forms, the texts were written on paper in many different personal handwriting styles. Unfortunately, no information about age, gender, cultural and educational background, etc. is available. Yet, the handwriting collection contains a large scope of texts, writing instruments, and handwriting styles, which is ideal for testing recognition systems for modern handwritings. The forms were processed in several steps into a handwriting database available at `http://www.iam.unibe.ch/fki/databases/iam-handwriting-database`.

2.4.2 Contents

The IAMDB currently includes 1539 page images and 657 writers as indicated in Table 2.1. The version 3.0 of the IAMDB consists of:

- Grayscale page images.
- Grayscale text line images together with their transcriptions and word locations.
- Grayscale word images together with their transcriptions.

Statistics of the textual content are given in Table 2.3. Note that besides text line segmentation, the IAMDB also contains sentence segmentation and grammatical word tags such as noun, verb, etc. Binarization thresholds as well as angles for slant normalization, i.e., the inclination of the letters, are suggested (see Chapter 3).

2.4.3 Characteristics

Writing samples from the IAMDB are shown in Figure 2.4. In the upper part of the special acquisition forms, a few sentences of the LOB corpus are displayed. The writers were asked to copy them in the separated area below and to use rulers in order to write in straight lines with convenient spacing in between (Figure 2.4, left column). Hence, the problem of text extraction is less difficult compared with historical documents. Also, the separation between text foreground and background is usually not problematic due to the contrast between the modern pens and the paper. Finally, the alignment between the LOB text and the handwriting is, in most cases, one to one, if the text was copied correctly.

However, there are handwriting styles that contain large descenders and ascenders, which can be problematic for text line segmentation (right column, top). Also, the handwritten text might not correspond perfectly with the LOB text in case of mistakes. In particular, the handwriting can contain stroke through text (right column, second image from top).

Character modeling is very challenging for the IAMDB, since the writing styles differ greatly (right column). Also, special characters such as punctuation marks have to be taken into account. On the other hand, unlike for many historical languages, large text corpora, e.g., the LOB corpus itself, can be used for language modeling with consistent orthography.

2.4.4 Key References

The IAMDB was introduced in [152] and has been improved in the following by corrections and by adding additional ground truth. The text segmentation scheme is presented in [258] and a thorough description of the database can be found in [154]. The IAMDB has been frequently used for experimental evaluation of handwriting recognition systems by several research groups. For HMM-based recognition, an overview of current transcription accuracies can be found in [177]. In this thesis, the IAMDB serves as a reference database for comparing historical handwriting recognition with modern handwriting recognition.

Part I

Handwriting Recognition

Chapter 3

Statistical Representation

Before machine learning and eventually recognition can be applied to handwriting images, text patterns are represented by mathematical models. In statistical pattern recognition, a fixed set of features, which are usually real-valued, is used to describe handwriting images. Typical features include, e.g., contour positions, center of mass, number of loops, etc. With respect to n features, images are represented with feature vectors $x \in \mathbb{R}^n$.

In case of OCR for printed documents, individual character images are taken into account that can be segmented reliably prior to recognition with respect to gaps between characters and known fonts. For handwriting, however, segmentation into characters is not feasible reliably before recognition in case of touching characters and variable handwriting styles (see Section 1.2.3). Instead, over-segmentation is typically applied to extract a sequence of features $\mathbf{x}$ given by

$$\mathbf{x} = x_1, \ldots, x_T;\ x_i \in \mathbb{R}^n;\ 1 \leq i \leq T \tag{3.1}$$

which captures a series of T character parts that are grouped to characters during recognition (see Chapters 5 and 6).

From a signal theory oriented point of view, it would be desirable to reconstruct the original movements of the writing instrument from a given handwriting image. In this scenario, the character parts x_i in Equation 3.1 correspond with individual strokes. However, in contrast to on-line handwriting recognition, where temporal information about the writing process is available due to special input devices [176], the reconstruction of this on-line information from a scanned or photographed image is a very challenging problem. Several attempts have been made in this direction, mainly in the 90's. Examples can be found in [22, 30, 58, 175].

In the past decade, however, a workaround based on a sliding window approach, which works very well in practice, has emerged as the predominant method, especially for HMM-based recognition [177]. Examples are given in [18, 27, 63, 65, 207, 245, 252]. In the sliding window approach, a small

analysis window is moved in writing direction over the image of an individual text line and a set of features are extracted at each window position. Here, the individual feature vectors x_i in Equation 3.1 correspond to the character parts captured by the analysis window.

Similar to the time frames considered for speech recognition, the sliding window approach provides spatial frames of handwritten text. These text frames do not necessarily correspond with meaningful strokes or graphemes, which are segmented explicitly as an additional image preprocessing step in some handwriting recognition systems [36]. Similarly to reconstruction of on-line information from images, most segmentation-based approaches stem from the 90's, often combined with rule-based recognition rather than learning-based methodology. In this thesis, we only consider the sliding window approach for statistical representation. Instead of pre-segmenting characters into meaningful entities, the task of constructing characters, words, and sentences is completely delegated to the recognition system. While this segmentation-free approach is more demanding for the recognition system, it does not suffer from explicit character segmentation errors.

In this chapter, we review statistical handwriting representation with respect to the sliding window approach. First, the scope of this review is commented on in Section 3.1. Next, image preprocessing is discussed in Section 3.2. The sliding window approach and common statistical feature sets are presented afterwards in Section 3.3. Finally, feature post-processing is reviewed in Section 3.4 with respect to feature selection that aims at improving the feature vector space prior to recognition.

3.1 Scope

The primary input for statistical handwriting representation is given by pre-segmented text line images and the output is a sequence of feature vectors. In order to cope with variance in the appearance of the handwriting, a text line normalization method is presented in Section 3.2.2 based on the method of Marti and Bunke [153]. Concrete layout analysis and text line extraction methods employed in this thesis are discussed in Chapter 7, where interactive solutions are presented in the context of ground truth creation.

For large vocabularies underlying natural language, whole word recognition is hardly feasible [32]. Hence, the statistical representation of complete words is not taken into account. Instead, words are treated as special cases of text lines and word images are represented with the sliding window approach as a sequence of feature vectors as well. Complete word representation is briefly reviewed, however, in Chapter 9 in the context of template-based keyword spotting. For an article on the role of complete word recognition, also called holistic recognition, we refer to [148].

Single character recognition is not taken into account as well, since a

reliable character segmentation prior to recognition is not feasible in general for handwritten documents [36][1]. Hence, statistical representation of single character images is not considered. For a survey on feature extraction methods for character recognition, we refer to [239].

Two feature sets extracted from binary images are presented in Sections 3.3.2 and 3.3.3. The first is the geometric descriptor proposed by Marti and Bunke [153] and the latter is the zoning descriptor proposed by Vinciarelli et al. [245]. Both feature sets are well-established for modern documents and are applied in this thesis to historical documents. Primarily, we use the geometric features, while the zoning features serve as an additional reference descriptor for experimental evaluation in Chapter 8.

In the context of feature selection, standard principal component analysis (PCA) and independent component analysis (ICA) are presented in Sections 3.4.1 and 3.4.2, respectively. In addition, a state-of-the-art non-linear feature space transformation by means of kernel PCA (KPCA), introduced by Schölkopf et al. [211], is presented in Section 3.4.3. Feature selection is considered for experimental evaluation in Chapter 8. To the knowledge of the author, KCPA has not been investigated for segmentation-free handwriting recognition before.

3.2 Image Preprocessing

Before serialization with a sliding window, an input document image has to be segmented into text lines. Afterwards, a standard practice in image preprocessing is given by text line normalization. It aims at standardizing the appearance of the handwriting prior to feature extraction. In the following, text line extraction is discussed in Section 3.2.1 and text line normalization in Section 3.2.2.

3.2.1 Text Line Extraction

Text line extraction involves layout analysis to identify text blocks consisting of consecutive lines. Afterwards, segmentation methods are needed for separating the text lines from each other within the text blocks. For a survey on text line extraction methods, we refer to [140]. Results of a recent competition can be found in [90].

After text line extraction, binarization is often applied to the text lines in order to separate the text foreground from the document background [178, 240]. For historical documents, in particular, binarization is a difficult problem due to paper or parchment degradation artifacts that can result in con-

[1]Note that there might be some special cases of historical manuscripts, for which character segmentation is feasible. In that case, OCR-like methods are possibly better suited for recognition, since an accurate segmentation into characters simplifies the problem of recognition.

siderable background noise [226]. Layout analysis, text line extraction, and binarization methods employed in this thesis are detailed in Chapter 7 in the context of interactive ground truth creation.

Another common image preprocessing step is given by thinning or skeletonization of binary images. The aim of these procedures is to obtain a stroke of one pixel width while maintaining the original stroke connectivity [133]. In Chapter 4, the skeletonization method used in this thesis is illustrated in the context of graph-based handwriting representation.

3.2.2 Text Line Normalization

Handwritten text images typically contain considerable variations in the appearance that impose difficulties for feature extraction. First, the general orientation and size of the handwriting can differ depending on the page layout and the digitalization device. Secondly, a variation in orientation and size of the individual characters is encountered, especially for multiple writer scenarios. Therefore, most handwriting recognition systems perform some form of text line normalization prior to feature extraction in order to standardize the appearance. Standardization is achieved by a series of rotation, shearing and scaling transformations. The parameters needed for these transformations, e.g., the rotation angle, are typically estimated on a binarized version of the text line image.

The orientation of the baseline, i.e., the skew, often deviates from the horizontal direction and needs to be corrected by rotation. Popular methods for skew angle estimation include contour minima interpolation [26] and entropy maximization of horizontal projection profiles [247]. Another orientation property of the handwriting that varies from one writer to another is the inclination of the individual characters, i.e., the slant. It can be removed by means of a shear transform. To that end, the mean direction of straight line segments is commonly used as an estimation of the slant [33, 246].

For normalizing the size of Roman scripts, the estimated midpoint or x-height of the handwriting, i.e., the height of the lowercase letter “x”, can be used in combination with the baseline for vertical scaling into three disjoint areas of equal height (upper, middle, and lower area) [153]. Another known practice for normalizing the size of the handwriting is to apply horizontal scaling with respect to the average character width [203].

In this thesis, we apply the well-known normalization method proposed by Marti and Bunke [153]. It is illustrated in Figure 3.1 for text line images of all four data sets under consideration (see Chapter 2). Normalization includes skew correction, followed by slant correction, x-height normalization and width normalization. While the final image operations can be applied to arbitrary images, the information needed for the normalization operations is extracted from binary images.

First, the skew angle is determined by linear regression of the bottom-

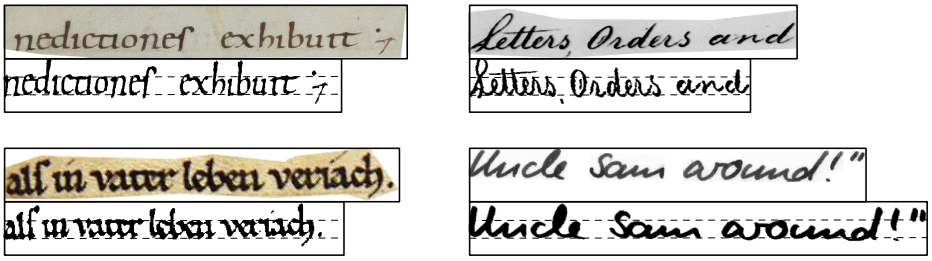


Figure 3.1: Text Line Normalization

most black pixels of each image column. Then, the skew of the text line is removed by rotation.

Afterwards, the slant is estimated by the mean direction of straight line segments and is removed by shearing. The line segments are given by straight line approximations of contour pixels. The contour pixels are extracted with respect to horizontal differences in pixel intensity in order to emphasize vertical contours.

For x-height normalization, vertical scaling is applied to obtain three writing zones of the same height, i.e., lower, middle, and upper zone separated by the lower and upper baseline. For the lower baseline, the regression result from the skew correction is used, and the upper baseline is found by linear regression of the upper-most black pixels of each image column. Then, for vertical scaling, the height of the baselines in the middle of the image is taken into account. In Figure 3.1, the three resulting writing zones are marked in the normalized images with dotted separating lines.

Finally, width normalization is performed with respect to the number of foreground crossings of a horizontal line through the middle zone of the handwriting. The number of foreground crossings per 100 pixels is set to a predefined value by means of horizontal scaling.

For more details on the different normalization steps, we refer to [153]. Note that depending on the type of document there might be good reasons to avoid some of the text line normalization steps, since any errors made at this stage may impede the overall recognition process. In particular, no slant correction is performed for the PARDB and the SGDB, since the letters are already rather upright.

3.3 Feature Extraction

For handwriting serialization using a sliding window, an analysis window is moved in writing direction over the preprocessed text line image, e.g., from left to right for Roman scripts and from right to left for Arabic scripts [63].

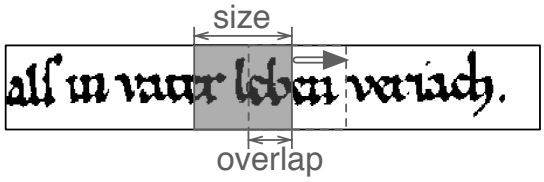


Figure 3.2: Sliding Window

At each position of the sliding window, a feature vector is extracted from the subimage captured by the window. Taking into account widows of different size and overlap, this results in a feature vector sequence description of the normalized text line image according to Equation 3.1. An illustration of the sliding window approach is shown in Figure 3.2. Note that the spatial frames resulting from sliding window extraction have a uniform width. An alternative approach has been proposed, e.g., in [31], where a sequence of handwriting skeleton edges with variable width is extracted in the context of HMM-based recognition. In Chapter 4, we propose another solution with dynamic width in the context of graph-based handwriting representation.

A variety of statistical features for handwritten text have been proposed in the literature. For a survey on features for single character recognition, we refer to [239]. Most single character descriptors are principally suited for describing sliding window frames as well. However, sliding windows are typically rather narrow and do not capture complete characters. Hence, local character properties such as contour positions and vertical foreground-background transitions are more frequently taken into account than global properties such as cavities and loops.

In the following, sliding window features are discussed in general in Section 3.3.1. Afterwards, two feature sets used for experimental evaluation in this thesis are described in more detail in Sections 3.3.2 and 3.3.3.

3.3.1 Sliding Window Features

In a recent report, binary pixel values of a window spanning a few image columns are proposed to be used directly as features for recognition with Bernoulli HMM [92]. Most feature sets, however, extract information at a higher level from the window frame. There are two arguments commonly put forward in favor of high-level information. First, the amount of data can be reduced and hence the number of model parameters that need to be estimated on learning samples is smaller. Secondly, more sophisticated features can focus on essential properties of the handwriting that are independent of the writing instrument and writing style. However, information is also lost when the raw pixel information is condensed into a heuristic set of features.

Especially if there are large amounts of training samples available, working with raw pixel intensities might be favorable[2].

Narrow windows of one image column have been proposed, e.g., in [153, 158]. In [158], the positions of the vertical black-white transitions in binary images are primarily used as features. In addition to a certain maximum of transitions, the gradients of the first and last transition, i.e., the contours, are used as features. They are calculated with respect to the successive image column. In [153], the number of black-white transitions is considered alongside with the contour positions, contour gradients, and some descriptors of the distribution of black pixels within the image column. A subset of these features is employed in [181] in the context of keyword spotting and an augmentation to several image columns with similar features can be found in [252], where some feature derivatives over the window columns are taken into account as well.

Wider windows of several image columns are used, e.g., in [13, 189, 231, 235, 245]. A small fraction of the image height, i.e., $\frac{1}{15}$, is used in [13] as the window width. Binary images and considerable window overlap are taken into account and each window is divided into 20 vertical cells. Features include the cell intensity, i.e., the fraction of black pixels, and its vertical and horizontal derivatives with respect to neighboring cells. Also the slope and correlation of strokes with respect to neighboring cells is taken into account. Similarly, square cells are used in [235] for gray-scale images. Here, a smoothed gray-scale value and its derivatives with respect to neighboring cells are used as features. A 4×4 window grid is used in [245]. The features are given by the fraction of black pixels in each cell and the correlation between neighboring cells is established by normalizing the feature values over the complete window. Based on similar grids and inspired by the success of the SIFT descriptor [147] in computer vision, pixel intensity gradients within a cell are considered as features in [189, 231] with respect to a partition of the circle into, e.g., 8 directions. In [189], binary images are smoothed with a Gaussian filter in order to calculate pixel intensity gradients.

The window widths discussed so far are typically given by a small fraction of the (normalized) image height and encompass only a small part of a character. The width of the sliding window is usually treated as a parameter in the systems mentioned and the final optimized value is not always indicated in the corresponding publications. However, it seems safe to say that the sliding windows, optimized with respect to the recognition accuracy of a validation set, are not much wider than an individual stroke. The fact that narrow windows perform well for recognition is somewhat astonishing, since much of the 2-dimensional nature of the handwriting is not captured in such small sliding windows. It indicates that the subsequent recognition, e.g., by

[2] "There is no data like more data." Yet in the context of historical handwriting recognition, training data is rather difficult and costly to obtain (see Section 1.4).

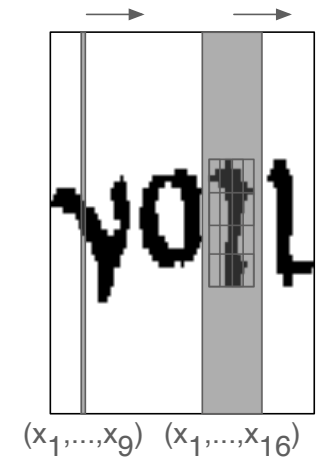


Figure 3.3: Statistical Features

means of HMM, is well-suited for reconstructing the complete characters from small parts. A recent report indicating benefits from larger windows can be found in [200], where the window width was considerably enlarged for integrating Gradient, Structure, and Concavity features (GSC), which were proposed for single character recognition in [66]. Integrating GSC features with a larger window has proven beneficial in [200] for HMM-based Arabic handwriting recognition.

3.3.2 Geometric Features

For statistical handwriting representation, the geometric features proposed by Marti and Bunke in [153] are primarily considered in this thesis. Using a sliding window width of one pixel, nine geometric features are extracted, i.e., $n = 9$ and T equals the width of the text line image in Equation 3.1. An illustration is shown in Figure 3.3 (left).

Three global features capture the fraction of black pixels, the center of gravity, and the second order moment. The remaining six local features consist of the position of the upper and lower contour, the gradient of the contours with respect to the subsequent sliding window, the number of black-white transitions, and the fraction of black pixels between the contours. For more details on the geometric features, we refer to [153].

3.3.3 Zoning Features

As a further reference descriptor, the zoning features proposed by Vinciarelli et al. in [245] are taken into account in this thesis. A sliding window with a width of 16 pixels is used that is adjusted to the area actually containing

foreground pixels at each window position. The window width is $\frac{2}{15}$ of the normalized image height, which corresponds roughly to twice the width of an individual stroke. The zoning features are then given by the fraction of foreground pixels in 4×4 regular window zones, i.e., $n = 16$ in Equation 3.1. An illustration is shown in Figure 3.3 (right).

The window center is moved column by column over the text line image, i.e., T equals the width of the text line image in Equation 3.1, resulting in large overlaps of the sliding window. At the start and at the end of the text line, the area outside the image covered by the sliding window is assumed to be background, i.e., white. For more details on the zoning features, we refer to [245].

3.4 Feature Selection

The feature sets introduced in Sections 3.3.2 and 3.3.3 evidently provide a different view on the handwriting image. In general, it cannot be predicted prior to recognition which heuristic descriptor performs best for a given handwritten document. In some cases, working with raw pixel intensities might even outperform sophisticated feature-based approaches if the heuristics used in the descriptor, e.g., extracting geometric properties, do not capture relevant information for separating different classes in the feature space. However, there are general methods for feature selection[3] applicable to any feature set that aim at improving class separability in the feature space, thereby often reducing the feature space dimension.

Performing machine learning in lower dimensions is desirable not only in terms of increased computational speed, but also in the context of the "curse of dimensionality" [15]. Due to the rapid increase in volume when adding extra feature space dimensions, large amounts of training samples are needed for modeling pattern classes accurately in sparse, high-dimensional feature spaces. As pointed out in Section 1.4, working with small training sets is important for handwriting recognition in historical documents, since training samples are difficult and expensive to obtain. Hence, reducing the statistical pattern representation to few dimensions is desirable.

Standard methods for feature selection include removing linear correlations among the features with Principal Component Analysis (PCA) [121] or Linear Discriminant Analysis (LDA) [155] and, as a more general approach, finding independent feature sources by means of Independent Component Analysis (ICA) [113]. For handwriting recognition, the sequential nature of the feature descriptors constrains the set of applicable feature selection methods. Because only the text line transcription is provided for the feature vector sequences of the training samples, but not letter labels for each

[3]Note that the term "feature selection" is used in a broad sense in this thesis and includes feature space transformation.

individual feature vector x_i in Equation 3.1, supervised algorithms that are based on class labels such as LDA are not applicable. Instead, feature selection is limited to unsupervised methods such as PCA and ICA. In [244], e.g., PCA and ICA were used to improve the zoning features presented in Section 3.3.3 for HMM-based cursive handwriting recognition.

The main restriction of the feature selection methods mentioned so far is their assumption of linearity. PCA does not take into account nonlinear correlations among the features and ICA performs a linear transform in order to maximize the independency of the features. With the introduction of kernel methods [159], nonlinear transformations have come within the reach of efficient computability. Kernelizable algorithms are executed in high dimensional feature spaces based only on the dot product. The dot product can often be computed efficiently even for nonlinear feature space mappings.

Since the introduction of kernel PCA (KPCA) about a decade ago [211], this powerful nonlinear technique has been successfully applied to several pattern recognition tasks including, for example, face detection [141] and palmprint recognition [62]. In [218], KPCA was used for Chinese character recognition. Here, images of single characters are recognized that are, in contrast, not available for Roman cursive handwriting recognition where whole text lines are considered without character segmentation. In the domain of speech recognition, which is closely related to segmentation-free handwriting recognition, a successful application was reported in [229] for an HMM-based recognition system.

In this section, several unsupervised feature selection methods are discussed that can be used to improve the statistical descriptors introduced in Section 3.3. PCA is addressed in Section 3.4.1, ICA in Section 3.4.2, and KPCA in Section 3.4.3.

3.4.1 Principal Component Analysis

Principal component analysis (PCA) [121] is a standard tool in statistics for coping with linear data correlation. Depending on its application area, it is sometimes also called the Karhunen-Loéve transform (KLT) or the Hotelling transform. A tutorial on PCA can be found, e.g., in [219].

In the context of statistical pattern representation, PCA applies a linear transform

$$y = Wx \tag{3.2}$$

with an $m \times n$ transformation matrix W to the original feature vectors $x \in \mathbb{R}^n$ in order to remove linear correlations amongst the new features $y \in \mathbb{R}^m$[4]. With respect to the new vector basis, given by the row vectors

[4]Note that feature vectors of dimension n are expressed as $n \times 1$ matrices in the context of matrix multiplication in this thesis.

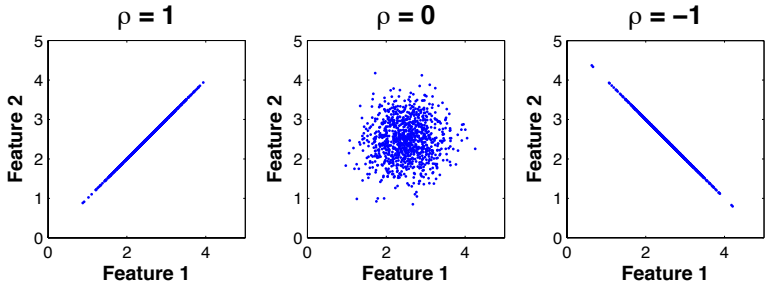


Figure 3.4: Linear Correlation

$w \in \mathbb{R}^n$ of W, the new features are also called *principal components*. The goal is to reduce the statistical pattern representation to $m \leq n$ uncorrelated principal components that capture most of the variance. Under the assumption that a high variance carries important information for class separation, components with low variance can be regarded as noise and can be dropped from the pattern representation without loosing important information. In the following, the goal of PCA is formalized and a standard solution based on linear algebra is briefly presented.

Considering two real-valued features $v_1, \ldots, v_N$ and $w_1, \ldots, w_N$ of N pattern samples, *sample mean* m_v, *sample standard deviation* s_v, *sample covariance* $cov_{v,w}$, and the *sample correlation coefficient* $r_{v,w}$ are given by

$$m_v = \frac{1}{N} \sum_{i=1}^{N} v_i \tag{3.3}$$

$$s_v = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (v_i - m_v)^2} \tag{3.4}$$

$$cov_{v,w} = \frac{1}{N-1} \sum_{i=1}^{N} (v_i - m_v)(w_i - m_w) \tag{3.5}$$

$$r_{v,w} = \frac{cov_{v,w}}{s_v s_w} \tag{3.6}$$

They estimate the mean μ_v, standard deviation σ_v, covariance, and Pearson's correlation coefficient $\rho_{v,w}$ of the underlying random process. The features are *uncorrelated* if and only if $\rho_{v,w} = 0$.

Figure 3.4 shows samples that are linearly correlated and uncorrelated. In case of maximum correlation $|\rho| = 1$, one feature can be expressed as a

linear function of the other. In that case, one dimension is redundant and the 2-dimensional vector basis can be replaced by a 1-dimensional vector basis that points in the direction of the straight line of correlation. This new vector basis still describes the distribution of the samples accurately and simplifies the representation. In case of minimum correlation $\rho = 0$, the data is "most random" and reducing a dimension would lead to a considerable loss of variance.

Note that Pearson's correlation coefficient only takes linear correlations into account. That is, even if $\rho_{v,w} = 0$, there might be non-linear correlations among the features (see Section 3.4.3). Also, this correlation measure assumes that the mean and the standard deviation are sufficient statistics of the underlying random process, e.g., in case of Gaussian distributions[5].

With respect to the correlation coefficient, the goal of PCA can be formulated as finding a transformation matrix W in Equation 3.2 such that no pairwise linear correlations according to Equation 3.6 exist in the new pattern representation $y \in \mathbb{R}^m$. Furthermore, the $m \leq n$ uncorrelated principal components have to be ordered by variance such that the first component captures most of the variance.

A standard solution is given by eigenvalue decomposition of the covariance matrix in the original feature space. First, the N pattern samples $x_1, \ldots, x_N$ with $x_i \in \mathbb{R}^n$ are centered by subtracting the mean given in Equation 3.3. Next, the $n \times n$ covariance matrix

$$C_X[i,j] = cov_{i,j} \tag{3.7}$$

between all features $1 \leq i, j \leq n$ is calculated with respect to Equation 3.5. Due to the centering of the data, $C_X = \frac{1}{N-1} XX^t$ holds with respect to the $n \times N$ data matrix X and its transposed X^t. Using eigenvalue decomposition, the symmetric and positive semi-definite matrix C_X can be written as

$$C_X = E^t D E \tag{3.8}$$

where D is a diagonal matrix, the rows of E correspond to the orthonormal eigenvectors of C_X, and the diagonal elements $\lambda_1, \ldots, \lambda_n$ of D are the eigenvalues of the eigenvectors. For $m = n$ and $W = E$, we obtain a diagonal covariance matrix

$$C_Y = \frac{1}{N-1} YY^t = \frac{1}{N-1} (EX)(EX)^t = EC_X E^t = E(E^t D E)E^t = D \tag{3.9}$$

in the new feature space, i.e., the (sample) covariance and hence the correlation coefficient is zero as required.

In summary, the eigenvector matrix E of the centered covariance matrix C_X provides a solution for the PCA transformation W, i.e., the resulting

[5] Fortunately, the assumption of Gaussian distribution often holds true in case of large-sample statistics due to the Central Limit Theorem.

principal components are uncorrelated. With respect to the orthogonality of the new eigenvector basis, an orthogonal transform (rotation and mirroring) of the original vector basis is obtained. Since the eigenvalues λ_i correspond with the sample variances s_i^2 in the new feature space (Equation 3.9), the principal components can be ordered such that $\lambda_1 \geq \ldots \geq \lambda_n$ as required. By choosing only the first $m \leq n$ components, the dimensionality is reduced while maximum variance is captured. An example is shown in Figure 3.5 (top), where the first principal component (y-axis) captures most of the variance and the second (perpendicular) component could be regarded as noise.

3.4.2 Independent Component Analysis

In contrast to PCA, Independent component analysis (ICA) [113] aims not only at finding linearly uncorrelated components, but also statistically independent components. They are expected to capture distinct properties of the patterns that are well-suited for classification. As for PCA, feature transformation is constrained to a linear transform given in Equation 3.2. The goal is to determine the parameters of the transformation matrix W such that statistical independence is maximized in the new feature space. In analogy to PCA, a second goal is to reduce the feature space to $m \leq n$ dimensions that capture most of the variance.

A classical example motivating the application of ICA is given by the *cocktail party problem* [44]. It refers to the ability of humans to separate several speakers at a cocktail party. In a more technical scenario, m speakers are registered with n microphones and the goal is to separate the speakers based on the registered speech signals. Assuming a linear mixing model of the acoustic signals and ignoring noise, the recorded signals $x \in \mathbb{R}^n$ are given by

$$x = My \tag{3.10}$$

where M is the $n \times m$ *mixing matrix* and $y \in \mathbb{R}^m$ are the original speech signals. For $m \leq n$, the goal is to find the inverse $W = M^{-1}$ (or the pseudo-inverse in case of $m < n$) of the unknown mixing matrix according to Equation 3.2. The transformation matrix is also called *demixing matrix*, which allows to separate the recorded signals into individual speakers. Since only the recorded signals are available but neither the original sources nor the mixing matrix, this task is called *blind source separation*. Assuming statistical independence of the sources[6] in addition to linearity, ICA provides a possible solution to the problem. With respect to the new vector basis, given by the row vectors of W, the new features are called *independent components*.

[6]Note that no assumptions about the underlying density distributions are required.

Note that for both M and y being unknown, only the direction of the independent components can be retrieved but not the variance of the components, because any scaling of a component of y can be compensated by scaling a row of M and vice versa[7]. Hence, an ordering of the independent components cannot be established as required. However, when first obtaining uncorrelated components using PCA before finding independent components, the same dimensionality reduction can be obtained as for PCA.

Note further that statistically independent features are uncorrelated, but the reverse does not hold true in general. For joint normal distributions, however, uncorrelated features are already independent. In this case, the application of ICA cannot further improve the feature space compared to PCA.

Several approaches to ICA have been proposed in the literature that differ in the objective function, also called *contrast function*, used for optimizing the parameters of W. Well-known principles include, e.g., maximum likelihood estimation [173], mutual information minimization [48], network entropy maximization ("infomax") [14], and negentropy maximization for individual components [111]. For a survey on ICA, see [112]. In this thesis, the FastICA algorithm presented in [111] (see [168] for a tutorial) is considered, which is one of the most widely used methods. It is introduced in the remainder of this section.

For FastICA, the *negentropy* $J(v)$ of a single feature v is used as a contrast function to obtain individual independent components. Based on the *differential entropy* $H(v)$, it is given by

$$J(v) = H(v_{Gauss}) - H(v) \tag{3.11}$$

$$H(v) = -\int p(v) \log p(v) dv \tag{3.12}$$

where $p(v)$ is the density function and v_{Gauss} is a Gaussian variable that has the same mean and variance as the feature v. From information theory it is known that a Gaussian variable has the largest entropy amongst all random variables of equal variance [53], i.e., it is "most random". Hence, negentropy is zero if v has a normal distribution and is greater than zero otherwise. Another well-known measure of *non-Gaussianity* is given by the kurtosis. However, negentropy is preferred since the kurtosis can be very sensitive to outliers [108]. In practice, an approximation of $J(v)$ is used [111].

Non-Gaussianity can be used for finding elements of the new vector basis, i.e., the rows of the transformation matrix W, as follows. Considering one row vector $w \in \mathbb{R}^n$ of W and one new feature v of y in Equation 3.2, v is given by

$$v = w^t x = w^t M y = \alpha^t y \tag{3.13}$$

[7]For the same reason, the sign of the direction remains ambiguous.

with respect to some coefficients $\alpha \in \mathbb{R}^m$ and taking into account Equation 3.10. That is, the feature v is a linear combination of the independent components with respect to some coefficients $\alpha_1, \ldots, \alpha_m$. For an optimal solution of W, all but one coefficients are zero, in which case v is an independent component. At this point, the assumption of statistical independence can be used in the context of the Central Limit Theorem, which states, under some conditions, that the distribution of a sum of independent random variables tends towards normal distribution. In the considered scenario, each non-zero coefficient α_i contributes another random variable, i.e., an independent component, to the feature v, which is given by a sum of independent components. Consequently, having only one non-zero coefficient corresponds with maximum non-Gaussianity, i.e., negentropy. By optimizing w with respect to maximum negentropy $J(v)$, an individual row vector of W can be found.

The FastICA algorithm proceeds as follows. First, the samples are centered with respect to their mean (Equation 3.3). In a second preprocessing step, feature correlations are removed and unit variance is established, also known as *whitening*. Standard PCA can be used to find uncorrelated features. At this point, the principal components can be ordered by variance and the dimensionality can be reduced as for PCA. Since the variance of the independent components cannot be determined, unit variance is established afterwards by scaling with the standard deviation in each feature dimension (Equation 3.4). The whitening has the effect that independent components can now be found by means of an orthogonal transform [111], which reduces the search space considerably.

After whitening, individual rows of W are found with respect to negentropy maximization of the corresponding feature. Instead of gradient descent, Newton's procedure is followed to achieve faster convergence. Subsequent components are found iteratively in orthogonal directions in a Gram-Schmidt-like procedure. For more details on FastICA, we refer to [111].

An illustration of the algorithm is given in Figure 3.5 for Gaussian and non-Gaussian features. In case of normal distributions (top), statistical independence is already established after applying PCA and ICA cannot further improve the feature space. The first principal component (y-axis) is found in the direction of maximum variance and the second component in perpendicular direction, corresponding with a rotation of the (already centered) data. For the non-Gaussian sample (bottom), which is given by a uniform distribution on a parallelogram, maximum independence is achieved. After whitening, independent components are retrieved by rotation.

Finally note that there are several recent approaches towards non-linear ICA, e.g., in [8, 103]. They are outside the scope of this thesis. However, the principles of kernel methods used for kernel ICA [8] are introduced in Section 3.4.3.

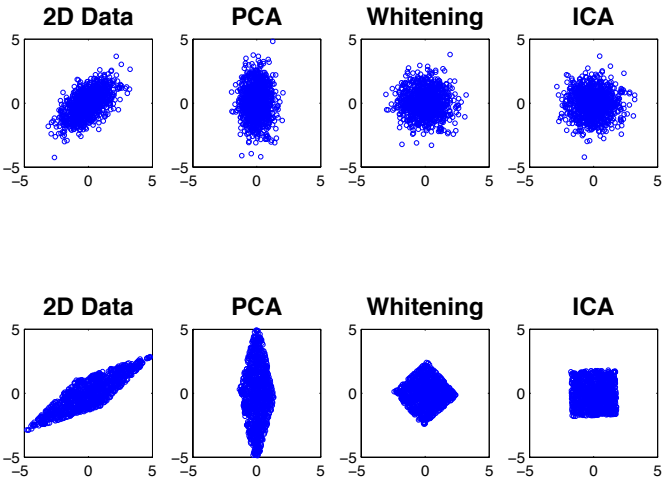


Figure 3.5: ICA

3.4.3 Kernel PCA

The feature selection and dimensionality reduction methods discussed so far, i.e., PCA and ICA, are limited by their assumption of linearity. In Figure 3.6, several cases are shown where the covariance of two features are zero, i.e., the features are not linearly correlated. Yet there are, evidently, non-linear correlations amongst the features. For instance, the circular sample in Figure 3.6 (middle) is generated by two independent random processes. The position on the circle is uniformly distributed over the circumference and the deviation from the circle has a normal distribution. Retrieving these two random processes from the given data samples in terms of features would be ideal for subsequent pattern classification.

While PCA provides an efficient algorithm with clear statistical properties for resolving linear feature correlations, doing the same in the general non-linear case is much harder and is an open field in research. In particular, methods are sought after that provide general solutions for a large scope of non-linear correlations without the need to assume specific structures, e.g., circular correlations, beforehand.

One approach to the problem is to accept non-linear feature correlations in the pattern representation and to focus on performing the classification task with non-linear class boundaries, which can be provided by both

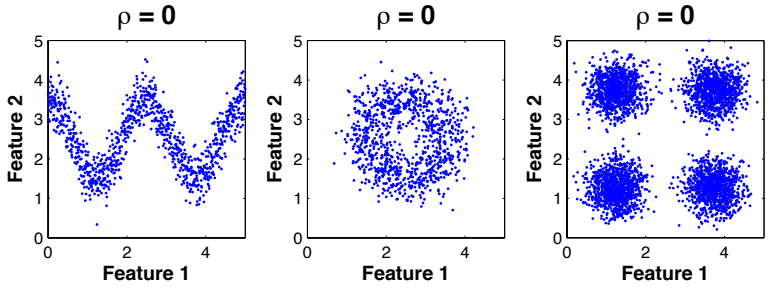


Figure 3.6: Non-Linear Correlation

generative and discriminative classifiers (see Section 1.2.2)[8]. In particular, multilayer neural networks (see Chapter 6) are known to be capable to approximate any function with arbitrary accuracy [104]. However, depending on the availability of training samples, there is always the risk of overfitting, i.e., adapting parametrized pattern distributions or class boundaries too closely to the training set, especially if a large amount of parameters are employed for non-linear class separation. Another problem is given by the greedy nature of many parameter optimization strategies, e.g., gradient descent in case of neural networks, which are prone to find only local optima for the parameters rather than global ones.

Another very general approach to the problem, which has become a state-of-the-art practice in the past decade, is given by so-called *kernel methods*. Based on the mathematical framework of reproducing kernels [7] from functional analysis, very efficient and flexible tools for non-linear machine learning and pattern recognition could be devised [210, 217]. Most prominently, the Support Vector Machine (SVM) was developed [23], which has demonstrated the possibility to efficiently combine non-linear feature mapping and linear classification in order to cope with non-linear pattern analysis. In the field of pattern recognition, applications of SVM include, e.g., face recognition [101], speech recognition [220], handwritten character recognition [198][9], and general object recognition [40]. The success of SVM has contributed to a more widespread investigation of kernel methods, e.g., in the context of structural pattern recognition [88], where the application

[8]Even the nearest neighbor rule provides a non-linear partition of the vector space into different classes, i.e., a Voronoi partition.

[9]Note that the typical application scenario for SVM is given by objects that are represented by a single feature vector. In the sequential case of continuous speech and cursive handwriting, SVMs are less frequently applied and if so, usually in hybrid approaches in combination with HMMs.

to non-vectorial patterns demonstrates the great flexibility of the approach. In the remainder of this section, we will discuss the basic principles of kernel methods and their application to non-linear feature selection by means of kernel PCA (KPCA) [211].

The underlying idea of kernel methods is to map the features into higher dimensional spaces, where their subsequent analysis is expected to be easier, e.g., tractable by linear algorithms. The intuition behind is that linear separability is improved with increasing dimensionality of the vector space when considering patterns in general position. This intuition is formalized in Cover's theorem [51]. For non-linear feature space mappings $\phi(x) = y$ with respect to the domains

$$\phi : \mathcal{P} \rightarrow \mathcal{F} \tag{3.14}$$

a linear algorithm executed in the new feature space $\mathcal{F}$ can be seen, in effect, as a non-linear algorithm in the original pattern space $\mathcal{P}$.

An important property of kernel methods, which makes such a feature transform appealing for pattern recognition, is that an explicit mapping is avoided and, instead, algorithms are performed implicitly by means of a *kernel function* $\kappa : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}$ given by

$$\kappa(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle \tag{3.15}$$

as the dot product $\langle ., . \rangle$ in $\mathcal{F}$, which can often be calculated efficiently even for very high (possibly infinite) dimensional feature spaces. A (complete) vector space over a field, $\mathbb{R}$ in our case, that is endowed with a dot product is a Hilbert space. The dot product can be used to define the standard norm $||y|| = \sqrt{\langle y, y \rangle}$ and hence the notion of distance in a vector space. Any statistical pattern recognition algorithm that can be expressed only in terms of dot products can now replace the standard dot product of $\mathbb{R}^n$ with another dot product and thus, be performed in some other feature space $\mathcal{F}$ implicitely, i.e., without performing the mapping ϕ and without knowing the actual feature values $y = \phi(x)$. Algorithms that can be rewritten in terms of only dot products, are called *kernelizable* and the shortcut that avoids an explicit mapping is called *kernel trick*.

For *valid kernel functions*, it is guaranteed that there is, in fact, some Hilbert space $\mathcal{F}$, in which the kernel function corresponds with the dot product [210]. Kernel validity can be ascertained, e.g., based on the *kernel matrix*

$$K[i, j] = \kappa(x_i, x_j) \tag{3.16}$$

of N patterns $x_1, \ldots, x_N$. A symmetric kernel function is valid if its kernel matrix is positive semi-definite for any choice of N patterns. In that case, the kernel function indeed corresponds with the dot product of some Hilbert space $\mathcal{F}$. For valid kernels, several closure properties hold, e.g., the sum and

Table 3.1: Standard kernel functions in $\mathbb{R}^n$.

Kernel	Definition	Parameter
RBF kernel	$\kappa(x,y) = \exp(-\gamma \cdot \|\|x-y\|\|^2)$	$\gamma > 0$
Polnomial kernel	$\kappa(x,y) = (\langle x,y\rangle + c)^d$	$c \geq 0, d \in \mathbb{N}$
Sigmoid kernel	$\kappa(x,y) = \tanh(\alpha \cdot \langle x,y\rangle + \beta)$	$\alpha > 0, \beta < 0$

product of two valid kernels are valid, which can be used to derive new kernels. An important observation at this point is that the pattern space $\mathcal{P}$ is not constrained to $\mathbb{R}^n$. Instead, valid kernels can also be defined for structural patterns, e.g., patterns represented by graphs (see Chapter 4). Hence, kernel methods offer an elegant bridge between structural and statistical pattern recognition by making structural patterns amenable to statistical algorithms [164], under the condition that the algorithms are kernelizable.

Rewriting algorithms in terms of only dot products is non-trivial in general and not feasible for all pattern recognition algorithms. Yet, the distance in $\mathcal{F}$, for instance, can be rewritten easily as

$$\begin{aligned} ||\phi(x_1) - \phi(x_2)||^2 &= \langle \phi(x_1) - \phi(x_2), \phi(x_1) - \phi(x_2)\rangle \\ &= \langle \phi(x_1), \phi(x_1)\rangle + \langle \phi(x_2), \phi(x_2)\rangle - 2\langle \phi(x_1), \phi(x_2)\rangle \\ &= \kappa(x_1,x_1) + \kappa(x_2,x_2) - 2\kappa(x_1,x_2) \end{aligned} \tag{3.17}$$

only in terms of the kernel function. Since the distance in a feature space is a fundamental property for pattern recognition, a wide range of algorithms are kernelizable. In particular, PCA is kernelizable as shown in [211]. As a result, the principal components $p_1, \ldots, p_N$ of $\phi(x) \in \mathcal{F}$ for an arbitrary pattern $x \in \mathbb{R}^n$ are given by

$$p_i = \sum_{j=1}^{N} \alpha_j^i \kappa(x_j, x) \tag{3.18}$$

where $x_1, \ldots, x_N$ are the training samples and α^i is an eigenvector of the kernel matrix. That is, the principal components in $\mathcal{F}$ can be expressed only in terms of the kernel function κ. Note that the centering of the data around the mean vector in $\mathcal{F}$ can also be kernelized and the principal components can still be ordered such that the first $m \leq N$ components capture most of the variance in $\mathcal{F}$. The final feature representation of KPCA is given by $(p_1, \ldots, p_m) \in \mathbb{R}^m$, where the dimensionality can even be higher than the original one in case of $n < N$.

The application of kernel methods requires the choice of a suitable kernel, which captures important properties of the underlying patterns. In the

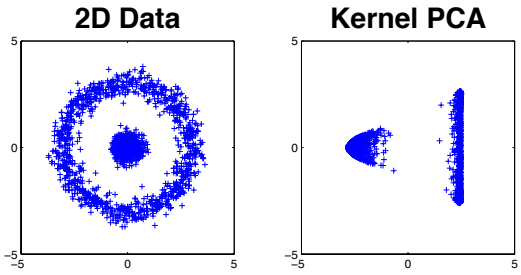


Figure 3.7: KPCA

literature, several standard kernels have been proposed, some of which are listed in Table 3.1. In this thesis, we employ the well-known radial basis kernel (RBF) for KPCA and contribute empirical evidence in Chapter 8 for its successful use in the context of segmentation-free handwriting recognition.

An illustration of KPCA with an RBF kernel is shown in Figure 3.7. In this example, two circular structures are displayed in $\mathbb{R}^2$ (left hand side), one with a normal distribution around the origin and the other with the circular distributions from Figure 3.6 (middle). Using an RBF kernel with an appropriate value of γ, which determines the influence range of the training samples, the two first principal components in $\mathcal{F}$ are displayed (right hand side) that provide an astonishingly intuitive result. It looks like a profile view on the data after pushing the origin into a third dimension. The first principal component (x-axis) captures the distance from the origin, i.e., the most important property for separating the central structure from the outer ring, which can be established with a straight line for the transformed data.

Note that, while the illustration of the KPCA transform is straightforward with respect to the new principal components, the characterization and interpretation of possible feature mappings ϕ and feature spaces $\mathcal{F}$, even for the well-known RBF kernel, is an ongoing issue in current research [225].

Chapter 4

Structural Representation

Statistical handwriting recognition based on real-valued feature vectors $x \in \mathbb{R}^n$, as discussed in Chapter 3, clearly is the predominant text representation approach nowadays. A crucial advantage of the statistical approach is given by the rich mathematical structure vector spaces provide. This is the conditio sine qua non for current state-of-the-art recognition methods, including generative HMM-based approaches (see Chapter 5) and discriminative NN-based methods (see Chapter 6). However, the use of feature vectors implicates two limitations. First, all objects are represented with a fixed number of features regardless of their complexity. Also, binary relationships among different parts of an object cannot be represented directly.

Both constraints can be overcome in structural pattern recognition using graph-based representation. Here, objects are described with a flexible set of nodes capturing subparts. Nodes can be linked with edges to represent binary relationships and both nodes and edges can be labeled with additional information. Special cases of graphs are given by trees, strings, and also feature vectors, which can be regarded as individual nodes with one vectorial attribute. Graph-based representation is a natural choice for describing, e.g., molecules as linked atoms, social networks as linked persons, and the Internet as linked webpages. Due to their high representational power, graphs have played an important role in pattern recognition in the past decades and a considerable amount of graph-based algorithms could be devised. For an extensive survey, see [49].

However, there is a major drawback of graph-based representation. Most of the basic mathematical operations provided by vector spaces are not available or not defined in a standardized way for such a general data structure and hence, the algorithmic processing capabilities are limited. Most algorithms concentrate on similarity measures between graphs, which make them directly amenable, e.g., to clustering [206] and nearest neighbor classification. Yet, current state-of-the-art handwriting recognition systems require more mathematical structure and hence graphs are only rarely used for

handwriting recognition nowadays.

In this chapter, a novel graph-based approach to handwriting recognition is presented that is based on a recently introduced vector space embedding method [187] in order to make handwriting graphs amenable to statistical handwriting recognition. Handwriting graphs are obtained from keypoints in skeleton images. Serialization is achieved, as for statistical representation, with a sliding window resulting in a sequence of graphs given by

$$\mathbf{g} = g_1, \ldots, g_T;\ g_i \in \mathcal{G};\ 1 \leq i \leq T \tag{4.1}$$

with respect to the graph domain $\mathcal{G}$. The graphs are then embedded in a vector space with the embedding function $\phi : \mathcal{G} \rightarrow \mathbb{R}^n$ based on the dissimilarity $d : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$ to n character prototypes $p_1, \ldots, p_n$ given by

$$\phi(g_i) = (d(g_i, p_1), \ldots, d(g_i, p_n)) \tag{4.2}$$

This results in a structural descriptor $\phi(g_i) \in \mathbb{R}^n$ and a sequence of feature vectors $\phi(g_1), \ldots, \phi(g_T)$ that is amenable to any statistical handwriting recognition system. Dissimilarity is given by graph edit distance (GED) [29], a very general method able to match arbitrarily labeled graphs. By normalization, similarity measures are obtained and we call the final structural descriptor Graph Similarity Features (GSF). They were first published in [78].

The proposed approach provides a bridge between structural and statistical pattern recognition and combines the advantages of both fields. Powerful graph-based representation can be used to describe handwriting images and statistical state-of-the-art recognition systems can be used to perform handwriting recognition. Prerequisite is the availability of a set of prototype characters, which can be obtained automatically from the training set by means of transcription alignment and prototype selection.

The remainder of this chapter is structured as follows. First, related work is discussed in Section 4.1. Then, the proposed structural representation is introduced step by step. The handwriting graphs obtained from skeleton images are presented in Section 4.2. Next, graph dissimilarity given by GED is discussed in Section 4.3 and automatic character prototype selection is elaborated in Section 4.4. The principal algorithm of GSF is presented afterwards in Section 4.5 including graph sequence extraction and dissimilarity space embedding. Finally, a global picture of the methodology underlying GSF is drawn in Section 4.6 and possible extensions are discussed in an outlook.

4.1 Related Work

The structural GSF descriptor is related to other approaches in the literature in several ways. In the following, we discuss its relation to dissimilarity space embedding, to handwriting recognition based on character prototypes, and to other structural handwriting representations proposed in the literature.

4.1.1 Dissimilarity Space Embedding

The idea for GSF is inspired by the work of Pekalska and Duin [170], which have raised the argument that dissimilarities are somewhat more fundamental than features, since they constitute the very notion of pattern classes. In [170], the effectiveness of dissimilarity space embedding (DSE) has been shown for various applications of statistical pattern recognition. In the work of Riesen and Bunke [187], DSE was successfully transferred to structural pattern recognition using the flexible GED as the underlying graph dissimilarity. An important advantage of DSE for structural pattern recognition is given by the fact that graphs become amenable to the rich repository of statistical pattern recognition methods based on their vectorial description. Empirical evidence for the successful use of DSE in combination with various statistical classifiers can be found, e.g., in [77], where we report significant improvements over classification in the graph domain for several data sets.

In general, the transition from graphs to feature vectors is expected to come with a loss of information, since a fixed number of vectorial features cannot cope with an arbitrary number of nodes, edges, and labels. In particular, straight-forward approaches such as counting the number of nodes and edges only take a small part of the underlying structural information into account. An example can be found in [43], where parts of Arabic words (PAW) are represented by graphs extracted from skeletonized handwriting images. Based on a 4×4 grid, the number of nodes and edges of a certain type are counted in each cell and are accumulated in a fixed-size feature vector for SVM-based classification. The resulting graph features performed significantly worse than the statistical Gabor features proposed in the paper.

In contrast to descriptive features like node and edge counts, the DSE approach is based on an actual graph matching that takes the complete graph structure into account. Indeed, the resulting dissimilarity features have proven to be robust with respect to the loss of structural information for a number of applications [187]. Besides DSE, another general approach to extract vectorial descriptors from graphs is given by spectral methods [253]. However, unlike DSE based on GED, current spectral methods cannot cope with arbitrarily labeled graphs.

Apart from vector space embedding, another known option for bridging the gap between structural and statistical pattern recognition is given by means of graph kernels [88, 164]. As discussed in Section 3.4.3 in the context of kernel PCA, obtaining valid kernels for structural data allows for performing statistical classification implicitly in a high-dimensional vector space. However, this approach is limited to kernelizable algorithms that are based only on dot products, since the actual feature values are not known in general. For the state-of-the-art recognizers considered in this thesis, however, the feature values are needed, which makes vector space embedding the first choice.

From the DSE point of view, the GSF descriptor contributes an embedding method suited for sequential graph problems as opposed to the standard scenario where each object is represented with a single graph. Empirical evidence is given in Chapter 8 that the method shows high potential for the task of handwriting recognition.

4.1.2 Character Prototypes

The idea of using character prototypes for handwriting recognition is well-known. An early example form the 90's can be found in the work of Edelman et al. [60]. Here, word contours are traced in order to extract interest points, e.g., start points, end points, intersections, sharp turns, etc. Handwriting images are then represented by point sets in $\mathbb{R}^2$. Afterwards, character prototypes are matched against complete words based on affine transforms. A special alignment procedure finally returns the most likely sequence of character prototypes with consistent spatial ordering.

Similarly, character prototypes are used by Rocha and Pavlidis in [188] for recognizing printed documents including hand-printed text. Interest points are extracted directly from gray-scale images. They represent small ridges and saddles on the gray-scale surface. Printed text is then represented with relative neighborhood graphs (RNG) [238], where nodes capture interest points and edges are inserted based on relative neighborhood in $\mathbb{R}^2$. The most likely sequence of character prototypes is finally found by means of error-tolerant subgraph matching, taking into account overlapping and broken characters.

More recently, character prototypes were used in the context of handwriting recognition in historical documents by Edwards et al. in [61]. Pixel intensities of character prototype images are used as character emission models of generalized HMM. HMM-based recognition (see Chapter 5) is then performed to find an optimal alignment between the character prototypes and complete text line images. This approach is still closely related to the previous examples. Instead of performing statistical learning, an alignment of text images with character prototypes is achieved directly. For historical documents, such a learning-free approach is, of course, very appealing because training data is difficult and costly to obtain (see Section 1.4). However, there are two limitations. First, character templates are needed for the complete alphabet, which already assumes a certain amount of labeled images, i.e., a training set, being available. Secondly, single templates do not capture the variance in handwriting appearance encountered even within a single manuscript.

In contrast to the learning-free methods mentioned, the proposed GSF approach does not attempt to find an optimal alignment of character prototypes with handwriting images directly. Instead, the local dissimilarity between text line graphs and character graphs is used as a structural descrip-

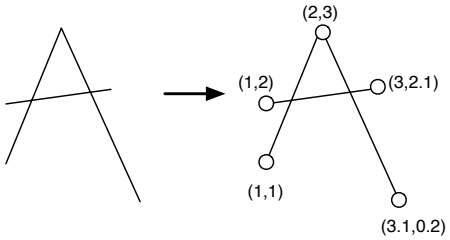


Figure 4.1: Line Drawing

tor[1]. The character prototypes are considered only as a visual vocabulary that contains valuable structural information for the handwriting recognition task. For single author manuscripts, in particular, a set of characters is expected to represent the most important structures, even if it does not cover the whole alphabet. This information is accessed by means of dissimilarities and statistical learning is applied afterwards in the same way as for any other vectorial descriptor, i.e., statistical character models are trained that capture variances in the handwriting appearance (see Chapters 5 and 6). In Chapter 9, we provide empirical evidence that even for single author manuscripts and a small amount of training data, the statistical learning approach dominates template matching in the context of keyword spotting.

With respect to the sliding window approach discussed in Section 3.3, the use of character prototypes requires a change in methodology. Instead of using a fixed-size sliding window, we consider a dynamic window width. At each window center position, the match against each prototype character is performed within the scope of the prototype width. For instance, the scope for matching an "m" against a text image at a given position is larger than the scope for matching an "i".

4.1.3 Structural Representation

Graph-based representation of handwriting is a very natural way to capture the 2-dimensional nature of handwritten text. In the literature, they are frequently used to represent and match single characters. A straight-forward letter representation can be found, e.g., in [187]. Artificial drawings with straight lines are considered for 15 capital letters of the Roman alphabet including *A, E, H, I,* etc. Graph nodes capture start and end points of the lines, labeled with their coordinates in $\mathbb{R}^2$, and unlabeled graph edges between start and end points represent the lines as shown in Figure 4.1.

Alternatively, stroke segments can be represented by graph nodes as proposed, e.g., in [5] for on-line Roman digits. For a given sampling length,

[1] In fact, after feature normalization, the similarity is taken into account instead of the dissimilarity for the final GSF descriptor.

the on-line strokes are segmented and the corresponding nodes are labeled with the start and end positions of the segments in $\mathbb{R}^2$. Unlabeled edges represent links between stroke segments. Another example of representing strokes with nodes can be found in [228], where straight lines in off-line Chinese character images are represented with graphs. Node labels include the stroke length, orientation, centroid position, and junction characterization. Edge labels include the distance between start, end, and centroid position of two strokes. Character matching is established by means of Hopfield NN [227].

Special interest points of on-line characters are represented by graph nodes in [37], which are linked by unlabeled edges. A set of 11 interest points are deduced from Catastrophe theory including, e.g., interior points, end points, bumps, and angles, which are proposed for general shape matching. In [2], interest points are extracted from off-line Arabic characters including, e.g., dots, intersections, and loops. Following the writing direction from right to left, the nodes are inserted in a tree representing the handwriting. A special error-tolerant matching and alignment procedure has been proposed in [1] for the handwriting trees in order to recognize complete text lines.

In the handwriting graphs considered in this thesis, skeleton endpoints and intersections are represented by nodes, which are labeled with their position in $\mathbb{R}^2$. Unlabeled edges link two points that are directly connected in the skeleton image. In order to capture the skeletons more densely, we take additional, equidistant points on the skeleton also into account. The additional graph nodes do not correspond to special interest points such as angles or bumps, but instead capture the general shape of the skeleton with a certain density. An advantage of this approach is that it is not dependent on interest point classification along the skeleton edges which can be prone to errors. A similar strategy was used, e.g., in [16] in the context of general shape matching, where word shapes are densely represented by contour points.

Astonishingly, the experimental evaluation in Chapter 8 indicates that using edges in the proposed skeleton graphs actually decreases the overall recognition performance of the GSF descriptor. Instead, an optimum is achieved for skeletons densely covered with nodes, but without using edges. As a positive side-effect, using no edges in the handwriting graphs allows for calculating GED in polynomial time, since it is reduced to an assignment problem (see Section 4.3).

4.2 Graph Extraction

The proposed handwriting graphs are extracted from thinned text images after binarization and normalization. Text line extraction and binarization are detailed in Chapter 7 in the context of interactive ground truth creation

and text line normalization is presented in Section 3.2.2. The subsequent processing steps discussed in this section are not constrained to text line images, but can also be applied to words, characters, or parts of characters.

In the following, a well-established definition of graphs and subgraphs is given in Section 4.2.1. Afterwards, handwriting skeletons are discussed in Section 4.2.2 and the final handwriting graphs are presented in Section 4.2.3.

4.2.1 Graph Definition

A *graph* is defined as a set of *nodes* that are linked with *edges*. Let L_V and L_E be label alphabets for nodes and edges, respectively. A graph $g \in \mathcal{G}$ in the *graph domain* $\mathcal{G}$ is the four-tuple $g = (V, E, \alpha, \beta)$, where

- V is a finite set of nodes
- $E \subset V \times V$ is the set of edges
- $\alpha : V \rightarrow L_V$ is the node labeling function
- $\beta : E \rightarrow L_E$ is the edge labeling function

Common label alphabets are numbers, symbols, or vectors. An edge $(u, v) \in E$ is *directed* from node u to v $(u, v \in V)$. If there exists an edge (v, u) for all edges $(u, v) \in E$ with the same edge label, the direction of an edge is irrelevant and the graph is called *undirected*. Figure 4.1 illustrates the graph representation of a line drawing using nodes for endpoints and undirected edges for lines. The nodes are labeled with the coordinates of the endpoints.

A *subgraph* is given by a subset of nodes and edges. Formally, graph $g' = (V', E', \alpha', \beta')$ is a subgraph of $g = (V, E, \alpha, \beta)$ if

- $V' \subseteq V$
- $E' \subseteq E$
- $\alpha'(v) = \alpha(v) \; \forall \; v \in V'$
- $\beta'(e) = \beta(e) \; \forall \; e \in E'$

Note that a more rigorous definition of *induced subgraphs* requires $E' = E \cap V' \times V'$ instead of the second condition. The subgraphs used in this thesis meet this requirement as well.

4.2.2 Handwriting Skeletons

Before graph extraction, the binary normalized text images are thinned in order to obtain a stroke of one pixel width while maintaining the original stroke connectivity [133]. The obtained skeletons represent the basic shape

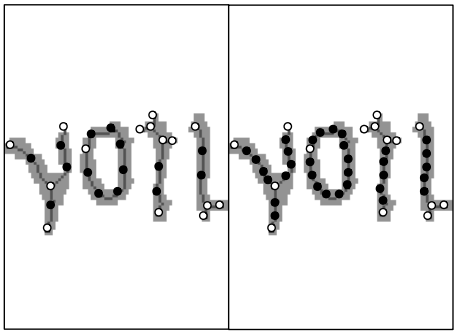


Figure 4.2: Handwriting Graphs

of the handwriting disregarding the thickness of the strokes, which is dependent on the writing instruments and the local binarization result. Such a dependency is not desirable, in general, because it would constrain a trained recognizer to a specific text appearance condition that may not be met in other manuscripts or even within the same manuscript.

In this thesis, the 3×3 thinning operator proposed in [97] is applied. Based on two sub-iterations on a checkerboard pattern, one pixel wide medial curves are extracted while preserving connectivity[2]. In Figure 4.2, the skeleton of the word "von", taken from the Parzival database, is illustrated within the text foreground. In general, the resulting skeletons capture the general shape of the characters satisfactorily for the data sets under consideration. In particular, only few additional branches are inserted in case of small bumps in the contour. Such additional branches are not desired since they are dependent on the writing instruments and the local binarization result. Examples can be seen at the stroke endings of the (broken) letter "n" in Figure 4.2.

On the one hand, possible enhancements include pruning of unstable skeleton parts [216] in order to increase the independence of the stroke thickness. On the other hand, shape contours [16] could be taken into account instead of skeletons when facing manuscript collections written with similar writing instruments.

4.2.3 Handwriting Graphs

Handwriting graphs are derived from skeleton images by representing keypoints with nodes labeled with their image coordinates $(x, y) \in \mathbb{R}^2$. Keypoints initially include endpoints with one direct neighbor, intersections with more than two direct neighbors, and the left-most, upper-most point of circular structures. In Figure 4.2, they are illustrated with white circles.

[2] An implementation is given by Matlab's `bwmorph` function.

Afterwards, equidistant additional connection points are added along the skeleton. Given a skeleton connection between two keypoints in form of a pixel chain $(x_1, y_1), \ldots, (x_m, y_m)$, connection points are inserted at regular distance D along that chain. Hereby, the distance $d(i, j)$ between two chain points (x_i, y_i) and (x_j, y_j) with $i < j$ is given by the sum

$$d(i,j) = \sum_{k=i}^{j-1} ||(x_{k+1}, y_{k+1}) - (x_k, y_k)|| \tag{4.3}$$

of Euclidean distances along the chain. Depending on the connection point distance D, handwriting graphs with different resolutions are obtained. In Figure 4.2, connection points are illustrated with black circles for $D = 9$ on the left and $D = 5$ on the right hand side. Both keypoints and connection points are added to the handwriting graph as nodes labeled with their coordinates $(x, y) \in \mathbb{R}^2$.

Finally, the graph nodes are linked with unlabeled, undirected edges whenever their corresponding skeleton points are directly linked along the skeleton. The edges are not illustrated in Figure 4.2 in order to improve the visibility of the underlying skeleton. For sparsely covered skeletons, the graph is basically given by linked endpoints and intersections. For densely covered skeletons, the edges seem somewhat obsolete since the proximity of the node labels already implicates their linkage. An advantage of an equidistant coverage of the skeletons is given by the fact that no classification of special interest points such as sharp turns is needed.

4.3 Graph Edit Distance

The structural descriptor presented in this chapter is based on a dissimilarity measure between graphs. In general, *exact* and *error-tolerant* graph matching can be distinguished. Exact matching is used to identify identical graphs or subgraphs in terms of nodes, edges, and labels. Unfortunately, no algorithm with polynomial time complexity is known for unconstrained graph isomorphism. Subgraph isomorphism, in particular, is known to be NP-hard [87]. Instead, tree search algorithms with exponential time complexity are frequently used to solve exact graph matching [242]. Note, however, that polynomial algorithms are available for special kinds of graphs such as trees [4], ordered graphs [119], and graphs with unique node labels [56].

Error-tolerant graph matching, in contrast, provides the means to match non-identical graphs, which is needed to establish a dissimilarity measure between handwriting graphs. Various methods for error-tolerant graph matching have been proposed in the literature including graph edit distance [29], genetic algorithms [54], graph kernels [88], spectral methods [253], and NN-based methods [227]. For a more detailed discussion of the algorithms for

exact as well as error-tolerant graph matching methods mentioned, we refer to [187].

In this thesis, we employ graph edit distance (GED) for calculating a dissimilarity measure $d(g_1, g_2) \in \mathbb{R}$ between two graphs $g_1, g_2 \in \mathcal{G}$

$$d : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R} \tag{4.4}$$

GED is an error-tolerant graph matching algorithm that was originally developed for the string domain [137, 249]. In the context of handwriting recognition, the string edit distance (SED) is traditionally used to calculate performance measures for text line recognition (see Section 5.2.4). In this thesis, we also employ SED for evaluating transcription alignment with inaccurate transcriptions (see Chapter 10). The basic idea is to calculate the dissimilarity between two patterns by applying distortions to one of the patterns until it is identical with the other and summing up the cost of each individual distortion. Because possible distortions and their cost can be defined in a very flexible way, the concept of edit distance could be transferred to trees [214] and eventually to graphs [29, 202]. An important property of GED is that it can cope with arbitrary graphs. In particular, arbitrary labels can be taken into account for both nodes and edges.

In the following, GED is defined in Section 4.3.1 and algorithmic solutions are discussed in Section 4.3.2.

4.3.1 Definition

The possible graph distortions are called *edit operations*. We consider a standard set of edit operations given by node and edge deletion, insertion, and label substitution. More formally, given two graphs $g_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $g_2 = (V_2, E_2, \alpha_2, \beta_2)$, the following edit operations are taken into account

1. Node deletion $v_1 \rightarrow \epsilon,\ v_1 \in V_1$
2. Node insertion $\epsilon \rightarrow v_2,\ v_2 \in V_2$
3. Node label substitution $a_1 \rightarrow a_2,\ a_1, a_2 \in L_V$
4. Edge deletion $e_1 \rightarrow \epsilon,\ e_1 \in E_1$
5. Edge insertion $\epsilon \rightarrow e_2,\ e_2 \in E_2$
6. Edge label substitution $b_1 \rightarrow b_2,\ b_1, b_2 \in L_E$

A sequence $\gamma_k = O_1, ..., O_k$ of k edit operations that transfer graph g_1 to graph g_2 is called *edit path*.

For each edit operation, a flexible *cost function* $c(O)$ can be defined in various ways. Typically, it assigns a real-valued, positive penalty $c(O) \geq 0$ to the corresponding graph distortion, which is dependent on the label value

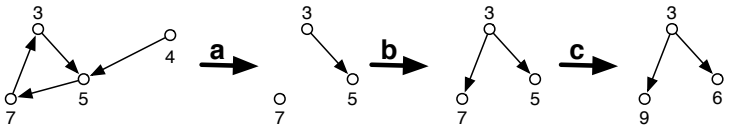


Figure 4.3: Graph Edit Distance

of the node or edge involved. The deletion of a node implies the deletion of all connected edges, for which an implicit deletion cost can be defined. Based on a given cost function, the *graph edit distance* is finally given by

$$d(g_1, g_2) = \min_{\gamma_k} \sum_{i=1}^{k} c(O_i) \tag{4.5}$$

the minimal sum of costs over all possible edit paths. Obtaining a minimum is feasible, because the set of minimum-cost edit paths is finite and limited by a maximum of $|V_1| + |E_1|$ deletions, $|V_2| + |E_2|$ insertions, and $|V_1| \cdot |V_2| + |E_1| \cdot |E_2|$ substitutions. Note that GED need not be metric. For instance, the symmetry $d(g_1, g_2) = d(g_2, g_1)$ is not guaranteed. However, the edit distance becomes metric when the cost function is positive semi-definite, symmetric, and satisfies the triangle inequality [29].

Deleting all nodes and edges of g_1 and inserting the nodes and edges of g_2 is always a valid edit path. However, this trivial edit path provides little information about the similarity of the graphs. More interesting are edit paths that retain identical nodes and edges, substitute similar nodes and edges in terms of their label values, and rarely delete or insert unmatched nodes and edges. If the cost function of the edit operations favors these kind of edit paths, GED becomes a profound dissimilarity measure. Therefore, cost functions often have high costs for insertions and deletions and relatively low costs for substitutions, especially if the label values of two nodes or edges are similar.

As an example, Figure 4.3 shows an edit path from the graph g_1 on the left-hand side to the graph g_2 on the right-hand side that applies deletion (a), insertion (b), and substitution (c). Let the cost function be $|\alpha(v)|$ for node deletion and insertion, 2 for edge deletion and insertion, and $|\alpha(v_1) - \alpha(v_2)|$ for node label substitution. Because the directed edges are unlabeled, no edge label substitution is needed. In a first step, one node and three edges are deleted with cost $c(v \to \epsilon) + 3 \cdot c(e \to \epsilon) = 10$. Then, an edge is inserted with cost $c(\epsilon \to e) = 2$. Finally, two node label substitutions are performed with cost $c(7 \to 9) + c(5 \to 6) = 3$. Obviously, this is the shortest edit path from g_1 to g_2 in terms of cost and therefore the edit distance is $d(g_1, g_2) = 15$ in this example.

For the handwriting graphs presented in Section 4.2.3, the Euclidean cost function is defined as

- $c(O) = ||(x_1, y_1) - (x_2, y_2)||$ for node label substitution
- $c(O) = C_n \geq 0$ for node deletion and insertion
- $c(O) = C_e \geq 0$ for edge deletion and insertion

which ensures GED to be metric. Since no edge labels are used, no edge label substitution cost is needed. The non-negative parameters C_n and C_e for deletions and insertions are optimized on a validation set in order to adapt the generic cost function to the data under consideration (see Chapter 8).

4.3.2 Algorithms

The calculation of the graph edit distance $d(g_1, g_2)$ is based on finding a minimum-cost edit path according to Equation 4.5. For unconstrained graphs, exact solutions are typically found by means of A^* tree search [100]. Starting with a node from the graph g_1, all possible mappings to a node of graph g_2 through edit operations are inserted as child nodes into the search tree. Then, the remaining nodes of g_1 are processed taking into account all mappings done so far until complete edit paths that transform g_1 into g_2 are found and inserted as leaf nodes. Finally, the minimal cost of a complete edit path is returned as the graph edit distance[3]. In order to avoid building up the complete search tree, the partial edit paths are sorted in each step by means of a heuristic function that estimates the expected cost of the complete edit path. Only the most promising partial path is then further expanded in each step and the cost of the first complete edit path is returned. If the cost estimation is always lower than the real cost, the result is guaranteed to be optimal. Several heuristic functions have been proposed, e.g., in [29, 241].

The exponential time complexity in the number of graph nodes restricts the exact algorithm to relatively small graphs in practice if their structure and label alphabets are unconstrained. In this thesis, we use a suboptimal algorithm for GED proposed in [165, 186], which is based on Munkres' algorithm [160] for solving the assignment problem and can be calculated in cubic time. Although the algorithm does not always find an optimal solution for GED, it is reasonably accurate, especially for small distances among similar graphs which is important for the task of classification [186].

GED can be formulated as an assignment problem as illustrated in Figure 4.4. In this example, we consider a graph g_1 with three nodes v_1, v_2, v_3 (top row) that is matched with a graph g_2 having four nodes u_1, u_2, u_3, u_4 (bottom row). For each node in g_1, an ϵ-node is inserted in the bottom row, and for each node in g_2 an ϵ-node is inserted in the top row. Assignments between the top and the bottom row correspond with node edit operations, e.g., substitution (v_1, u_3), deletion (v_3, ϵ), and insertion (ϵ, u_2).

[3] Edge edit operations are performed implicitly and contribute to the cost.

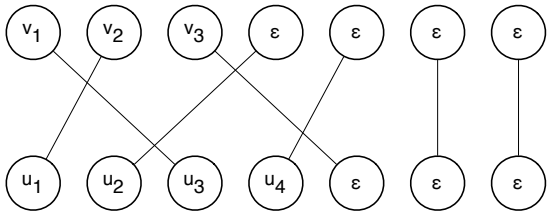


Figure 4.4: Assignment Problem

The cost of an assignment is given by the cost of the corresponding node edit operations[4]. Finding a complete assignment with minimum cost therefore corresponds to finding an optimal solution for GED if the graphs contain no edges[5]. The ϵ-nodes allow for deleting all nodes of g_1 and inserting all nodes of g_2. Clearly, this procedure can be applied to arbitrarily sized graphs.

Munkres' algorithm, also known as the Hungarian algorithm, solves the assignment problem in $O(n^3)$ where n is the sum of nodes of g_1 and g_2. It is based on a sequence of row and column operations on the $n \times n$ cost matrix that contains all possible assignment costs. For details on this algorithm, we refer to [160].

So far, only node operations are considered. By taking the implied cost of adjacent edge operations into account for each node assignment, GED can be approximated for arbitrary graphs. It is suboptimal because only local adjacent edge structures are matched instead of the global edge structure. If the graph contains no edges an optimal solution for GED is obtained.

4.4 Character Prototype Selection

The proposed structural GSF descriptor is based on local dissimilarities between text line graphs and character prototype graphs. The character prototypes are used as a visual vocabulary of structures that are frequent within the handwriting and are therefore expected to contain important information for recognition. In the following, the automatic extraction of character graphs from text line images in the training set is discussed in Section 4.4.1. Given several graphs for each character, Section 4.4.2 elaborates on different strategies for selecting few representatives as the final prototypes.

[4]The assignment cost (ϵ, ϵ) is zero.

[5]The graph describing the assignment, e.g., the graph depicted in Figure 4.4, has a special form called *bipartite graph*. Bipartite graphs can be divided into two subsets of nodes, e.g., the top and the bottom row in Figure 4.4, such that each edge links a node form one set with a node from the other set. Therefore, the GED approximation discussed here is often referred to as bipartite graph matching.

4.4.1 Character Extraction

A straight-forward approach to obtaining character template images is given by manual selection from the text line images in the training set. Although character start and end positions are often ambiguous even for humans in case of touching characters, this allows for extracting clean character instances. In the recent literature, manually selected character templates have been used, e.g., for recognizing Latin manuscripts [61], Arabic handwriting [39], and low-quality medical forms [35]. However, manual selection is a time-consuming process and it is not guaranteed that all variants of the character shapes are captured.

As a fully automatic alternative, HMM-based forced alignment is used in this thesis. Based on statistical features, which are independent of character templates, an HMM recognizer is trained and used to segment the text lines in the training set into characters. This forced alignment method is discussed in Section 5.2.3 in the context of HMM-based recognition. The resulting character images are often noisy, i.e., character parts may be missing and parts of other characters may be present. Yet, the basic character shape is typically captured in large parts and can be used for structural comparison in the proposed GSF framework. For each character, a number of template images are obtained from the training set and are represented with handwriting graphs following the procedure discussed in Section 4.2. This results in several graphs for each character. In the following, Section 4.4.2 presents several strategies for selecting few representatives as the final prototypes.

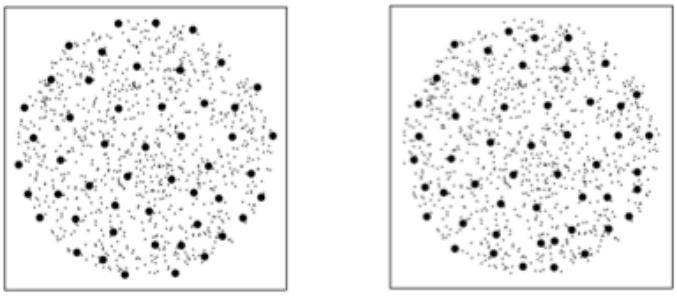
Note that in the GSF framework, the alphabet does not need to be covered completely because the character templates are only used as a visual vocabulary rather than for recognizing handwriting directly.

4.4.2 Character Selection

After HMM-based forced alignment and character graph extraction, a set G of graphs is available for each character present in the training set. By means of prototype selection, a subset $P \subseteq G$ is extracted that aims at representing the different writing styles of a character. Hereby, redundancy should be avoided while maintaining the capability to represent dominant character shapes. Although a random selection might work well in some cases, we use four other selection strategies proposed in [187] that construct the set of prototypes in a more controllable manner. They are described in the following.

Median Selection Using median graph selection, a single prototype $P = \{p_1\}$ is chosen per character. The prototype p_1 is given by the median graph

$$median(G) = \underset{g_1 \in G}{\operatorname{argmin}} \sum_{g_2 \in G} d(g_1, g_2) \tag{4.6}$$



(a) Spanning (b) K-Centers

Figure 4.5: Prototype Selection

with respect to the GED $d(g_1, g_2)$, i.e., the median graph is characterized by minimizing the sum of edit distances to all other graphs in G.

Center Selection A slightly different concept than the median graph is given by the center graph

$$center(G) = \underset{g_1 \in G}{\operatorname{argmin}} \max_{g_2 \in G} d(g_1, g_2) \tag{4.7}$$

where the maximum GED to all other graphs in G is minimized. Again, the center graph is selected as a single prototype $P = \{p_1\}$ to represent G.

Spanning Selection For spanning prototype selection, the set $P = \{p_1\}$ of prototypes is initialized with the median graph $p_1 = median(G)$. Then, additional prototypes p_i are added iteratively based on the rule

$$p_i = \underset{g \in G \setminus P}{\operatorname{argmax}} \min_{p \in P} d(g, p) \tag{4.8}$$

until a given number k of prototypes $P = \{p_1, \ldots, p_k\}$ is selected. At each step, the added prototype p_i is the graph that differs most from the previously selected prototypes.

k-Centers Selection The k-centers selection is based on a k-medians clustering of G [125]. Starting from clusters $C_1 = \{c_1\}, \ldots, C_k = \{c_k\}$ with initial centers c_i obtained from a spanning selection of k prototypes, each of the remaining graphs is added to the cluster with the nearest cluster center. Then, cluster centers are recalculated using

$$c_i = center(C_i) \tag{4.9}$$

This process is repeated until no more cluster centers are changed. It results in k prototypes $P = \{p_1, \ldots, p_k\}$ given by the final cluster centers.

An illustration of spanning and k-centers prototype selection is shown in Figures 4.5a and 4.5b for a vectorial example. Instead of graphs, 2-dimensional vectors are considered and instead of GED, the Euclidean distance is used in order to illustrate the selection strategies. While spanning prototype selection includes outliers on the border of the set, k-centers selection moves the prototypes towards the centers of dense cluster areas. Sample images of selected character prototypes for the data sets under consideration are provided in Chapter 8 in the context of GSF-based recognition.

4.5 Graph Similarity Features

In the previous sections, the input for the GSF descriptor was discussed. It consists of a text line graph g and character prototype graphs $p_1, \ldots, p_n$. The handwriting graphs are given by skeleton points labeled with their coordinates and linked with undirected, unlabeled edges (see Section 4.2). A dissimilarity measure between two graphs is established by means of graph edit distance (GED) based on a Euclidean cost function, which requires two parameters, i.e., the node insertion and deletion cost $C_n \geq 0$ and the edge insertion and deletion cost $C_e \geq 0$, respectively (see Section 4.3). The character prototype graphs may include several representatives for each character present in the training set (see Section 4.4).

In this section, the main GSF algorithm is presented. Based on dissimilarity space embedding [170, 187], the output is a sequence of feature vectors

$$x_1, \ldots, x_N, \; x_i \in \mathbb{R}^n \tag{4.10}$$

with respect to the text line image width N. At each position $0 \leq i \leq N$ within the text line, n structural similarities $x_{i,1}, \ldots, x_{i,n}$ to the character prototypes are captured. The GSF descriptor bridges the gap between structural representation and statistical recognition. It captures structural properties by means of prototype similarities and provides a sequence of vectorial features that can be used for statistical handwriting recognition in the same way as any other vectorial descriptor from Section 3.3.

In the following, the special sliding window used for GSF is discussed in Section 4.5.1 and a pseudo code of the GSF algorithm is presented in Section 4.5.2. Its computational complexity is commented in Section 4.5.3. Finally, remarks on recommended parameter values and illustrations of the resulting GSF descriptor are given in Section 4.5.4.

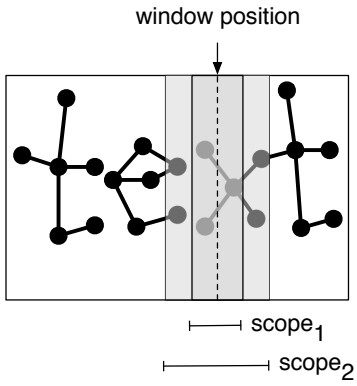


Figure 4.6: Sliding Window

4.5.1 Sliding Window

The sliding window used for GSF is different from the sliding window introduced in Section 3.3 in two aspects.

First, the sliding window is moved over text line graphs rather than text line images. This requires the text line graphs to contain information about the width of the image, accessed by the function $width(g)$, as well as information about the subgraph contained within a given start and end position of the window, accessed by the function $subgraph(g, x_{start}, x_{end})$. Both requirements are easily satisfied by the handwriting graphs under consideration because the width of the original image is known and the horizontal node position is contained in the node labels. In particular, the graph nodes can be sorted based on their horizontal x position, which allows for an efficient implementation of the sliding window. The vertical y position is only used for matching two graphs with GED. Hereby, it is assumed that both graphs have been normalized in vertical direction (see Section 3.2.2).

Secondly, the sliding window has a dynamic scope instead of a fixed window width. For each window center position, the widths of the local subgraphs are equal to the widths of the prototype characters. That is, different widths are taken into account for different prototypes in order to guarantee an optimal scope for the structural match. For example, the scope used for matching the wide prototype letter "m" is larger than the scope for matching the narrow prototype letter "i". In Figure 4.6, the sliding window used for GSF is illustrated for two different scopes. In this example, the subgraph of the narrow scope contains 3 nodes and 2 edges and the subgraph of the wide scope contains 7 nodes and 4 edges. Note that at the beginning and the end of the text line, the parts of the window that are outside the image do not contain any nodes or edges and do not need to be treated in a special way.

4.5.2 Algorithm

Algorithm 1 Graph Similarity Features

Require: text line graph g, prototype graphs $p_1, \ldots, p_n$, cost $C_n, C_e \geq 0$
Ensure: graph similarity features $x_1, \ldots, x_N$

1: $N \leftarrow width(g)$
2: **for** $j = 1 \rightarrow n$ **do**
3: $\quad w \leftarrow \frac{width(p_j)}{2}$
4: $\quad$ **for** $i = 1 \rightarrow N$ **do**
5: $\quad\quad g_i \leftarrow subgraph(g,\ i - w,\ i + w)$
6: $\quad\quad x_{i,j} \leftarrow 1 - \frac{GED(g_i,\ p_j,\ C_n,\ C_e)}{C_n \cdot (|g_i|_n\ +\ |p_j|_n)\ +\ C_e \cdot (|g_i|_e\ +\ |p_j|_e)}$
7: $\quad$ **end for**
8: **end for**
9: **return** $x_1, \ldots, x_N$

A pseudo code for GSF is given in Algorithm 1. The main loop in Line 2 is over the character prototype graphs $p_1, \ldots, p_n$. The scope $2w$ of the sliding window is set to the width of the current prototype. Afterwards, the center of the sliding window is moved over each text line image column $1 \leq i \leq N$ in the inner loop in Line 4. For each sliding window position i, the subgraph g_i within the window start $i - w$ and end $i + w$ is matched against the prototype p_j using GED in Line 6. With respect to the cost $C_n, C_e \geq 0$ for insertion and deletion of nodes and edges, the dissimilarity is normalized in order to obtain a similarity measure

$$0 \leq x_{i,j} \leq 1 \tag{4.11}$$

where 0 means minimum similarity (maximum GED) and 1 means that the graphs are isomorphic to each other (minimum GED). The maximum GED is given by deleting all nodes $|g_i|_n$ and edges $|g_i|_e$ of the text line subgraph and inserting all nodes $|p_j|_n$ and edges $|p_j|_e$ of the prototype graph. Taking into account the corresponding cost, this corresponds with the denominator in Line 6. The minimum GED is given if all nodes can be substituted with zero cost, i.e., they have the same node labels and the edge structure is not violated.

A standard normalization is finally applied to the output of Algorithm 1 by comparing the similarity of a single prototype with the similarities of the other prototypes. The final GSF descriptor is given by

$$\frac{x_{i,j}^2}{\sum_{k=1}^{n} x_{i,k}} \tag{4.12}$$

Without squaring, the normalized features would sum up to one and could be interpreted as the probability of each character to appear at a given window

position. However, keeping the information of the actual similarity value by means of squaring has shown a positive effect. Details on the experimental evaluation of GSF are given in Chapter 8.

4.5.3 Complexity

The time complexity of Algorithm 1 is given by $O(nNm^3)$ with respect to the number of prototypes n, the number of image columns N, and the expected number of graph nodes m taken into account for GED. It is a "good-natured" complexity in the sense that it is polynomial rather than exponential in m, which would be expected for matching arbitrary graphs. This stems from the fact that only an approximative GED is taken into account based on Munkres' algorithm (see Section 4.3.2). In case of graphs without edges, the resulting GED is optimal.

Straight-forward speed-ups can be achieved by reducing the number of prototypes, which are not required to cover the whole alphabet, by moving the sliding window more than one image column, and by choosing a sparse skeleton coverage (see Section 4.2.3) for reducing the expected number of graph nodes. In order to reduce the complexity of the graph matching, further approximations of the node assignment could be investigated. In particular, the Hausdorff distance [110] could be used as a rough approximation in case of graphs without edges, which can be regarded as point sets. In this thesis, we only investigate GED that can potentially be applied to arbitrary graphs without constraints on their structure and label alphabets.

4.5.4 Parameters

Parameters of GSF, which have a strong effect on the recognition accuracy, include the node distance D that determines the skeleton coverage (see Section 4.2.3) and the cost C_n, C_e for GED (see Section 4.3.1). Based on our experimental evaluation discussed in Chapter 8, we found that graphs without edges but with a dense skeleton coverage performed best. This is somewhat astonishing since edges normally contribute much to the graph representation. In case of the proposed skeleton graphs, however, a dense node coverage of the skeletons already implies node linkage for close node proximity. Using edges in this scenario is not only redundant but can also lead to unwanted constraints in case of broken characters, which are frequent for historical documents.

For all three historical manuscripts presented in Chapter 2, in particular with respect to the image resolution considered, we can recommend $D = 3.0$ and $C_n = 3.0$. This means that the nodes have a Euclidean distance of 3 on the skeleton and for GED, substitution is preferred over insertion and deletion if the Euclidean distance of two node labels is less than 6. On the downside, a large number of nodes are considered for GED in this setting,

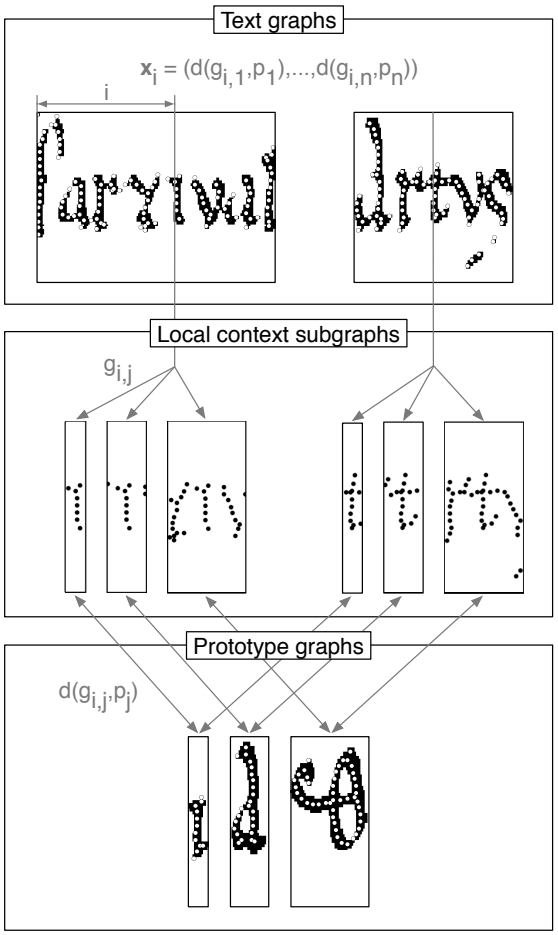


Figure 4.7: GSF Descriptor

which increases the computation time drastically (see Section 4.5.3). On the upside, an optimal solution for GED is obtained in the absence of edges.

For graphs without edges and dense skeleton coverage, Figure 4.7 illustrates the GSF descriptor for the word graphs of "Parzival" and "Artus" from the Parzival database and prototype graphs of "i", "A", and "D". The nodes are illustrated as white circles and are overlaid over the normalized binary image using $D = 5.0$. For a given position i within the word graphs, three local subgraphs $g_{i,1}, g_{i,2}, g_{i,3}$ with different context scopes are extracted and compared with the prototype graphs p_1, p_2, p_3 using GED $d(g_{i,j}, p_j)$. After normalization, the obtained dissimilarity features correspond with the GSF descriptor.

Another illustration of the GSF descriptor is shown in Figure 4.8 where the similarity features of all prototype letters of the word "Parzival" are displayed. A color is assigned to each individual letter and the peaks in

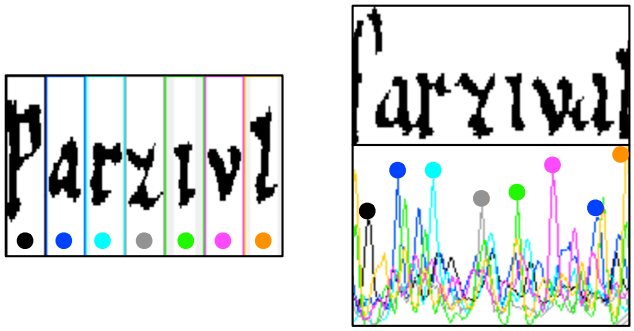


Figure 4.8: Similarity Features

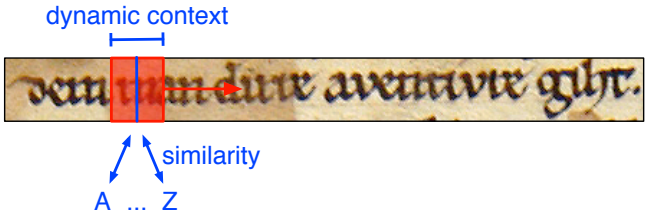


Figure 4.9: Feature Framework

the middle of the corresponding letters in the word graph are emphasized with colored circles. Note that instead of performing a recognition at this stage, the GSF descriptor only describes the word with several real-valued features that contain structural similarity information. The task of training character models and performing handwriting recognition is delegated to statistical recognition systems, e.g., HMM or NN, which can learn variants of the handwriting appearance.

4.6 Summary and Outlook

In summary, the GSF descriptor captures structural similarities to character prototypes at each position within a text image as illustrated in Figure 4.9. The scope for the structural comparison is dynamically adapted to the width of the character prototypes. The resulting sequence of feature vectors is used as input for statistical handwriting recognition systems.

In this thesis, we represent skeleton points with graph nodes labeled with the image coordinates and link them with unlabeled, undirected edges. As it turns out, a dense skeleton coverage and graphs without edges perform best for the historical manuscripts considered. The structural similarity is obtained by means of GED, which is potentially able to match arbitrary

graphs without constraints on their structure and label alphabets. GED is calculated efficiently in cubic time by means of an approximative method using Munkres' assignment algorithm [160, 165, 186]. For graphs without edges, an optimal solution for GED is obtained.

Based on dissimilarity space embedding [170, 187], the proposed approach allows for combining structural handwriting representation with statistical handwriting recognition systems. This bridge between structural and statistical pattern recognition allows to combine advantages of both fields, i.e., the power of structural representation and the rich repository of recognition algorithms available for statistical recognition.

Future work includes the investigation of alternative structural representations. So far, the skeleton graphs represent the basic structure of the handwriting, but its thickness, color spectra, intensity gradients, etc. are not taken into account. While such additional information limits the resulting recognition system to specific text appearance conditions, it is expected to increase the recognition accuracy within these conditions. For instance, it would be promising to integrate ink-specific features in the representation of Latin manuscripts written with ink on parchment, which would already encompass a large spectrum of manuscripts.

From a global perspective beyond the structural representation, more system variants are imaginable. It should be noted that the dissimilarity space embedding approach is not restricted to graphs. On the contrary, it was originally proposed for feature vectors [170]. Therefore, any of the existing statistical descriptors could be used, e.g., in combination with the Euclidean distance, to establish prototype dissimilarities and provide (dis)similarity features.

Finally, fully-fledged character classifiers could be employed to provide similarities in form of recognition probabilities. The proposed framework would offer a bridge between single character recognition and text line recognition, which has, to the knowledge of the author, not been attempted so far in this manner. In this scenario, the character classifiers need to be trained first on labeled character images. The results obtained by automatic HMM-based forced alignment are promising for obtaining "relatively clean" training samples for individual characters from text line images.

The options are manifold and we esteem this future line of research as promising, since the skeleton graphs employed in this thesis already show great potential for handwriting recognition in historical documents (see Chapter 8). The approach is particularly suited for historical manuscripts because they are typically written much more neatly than modern documents and the individual character shapes often show less variability than for modern handwriting. In such a scenario, the structural similarities with respect to a small set of characters can contribute robust features for handwriting recognition.

Chapter 5

Hidden Markov Models

Based on the sequential representation of text line images discussed in Chapters 3 and 4, a first approach to machine learning and recognition based on Hidden Markov Models (HMM) is introduced in this chapter.

The application of HMM to off-line handwriting recognition has proven to be one of the few approaches that can cope with the difficult recognition conditions mentioned in Section 1.2.3, i.e., performing segmentation and recognition at the same time. Originally proposed for the task of speech recognition [179], these powerful statistical models could also be adopted for handwriting recognition. In fact, HMM are probably the most widely used statistical models for handwriting recognition nowadays, and in the past two decades they have played an important role for the improvements achieved in the field [177]. One of the most important features of HMM-based recognition is that no segmentation of text line images into characters is needed prior to recognition. Also, the availability of efficient algorithms for learning and recognition has contributed to the success of HMM-based approaches.

In this chapter, we review the state-of-the-art of HMM-based handwriting recognition. First, the scope of the review is outlined in Section 5.1. Afterwards, HMM-based text line recognition is discussed step by step in Section 5.2. The typical topology and parameterization is introduced in Section 5.2.1 and algorithms for appearance-based learning and recognition are discussed in Section 5.2.2. Next, the integration of language models is presented in Section 5.3. Finally, an introduction to confidence modeling is provided in Section 5.4.

5.1 Scope

HMM are considered in this thesis as the most widely used generative method for segmentation-free handwriting recognition. Not included are some related techniques less frequently used, e.g., Markov random fields [34,

203], pseudo 2-dimensional HMM [21], conditional random fields [69], maximum margin HMM [57], and HMM/NN hybrid systems [65, 128].

We focus on the predominant application of HMM given by segmentation-free recognition of handwritten text lines for large vocabularies and multiple writing styles[1]. Based on character models, the presented methods are, in principle, applicable to any alphabetical language. Not included are segmentation-based approaches such as single character recognition. Single word recognition is treated as a special case of text line recognition.

The well-established core framework of HMM consists of appearance-based recognition presented in Section 5.2. The n-gram language models discussed in Section 5.3 can also be regarded as standard components nowadays. Tentative approaches towards more sophisticated language models, e.g., stochastic context-free grammars (SCFG) [261], are outside the scope of this thesis. However, we discuss some promising approaches towards confidence modeling in Section 5.4, which is an important topic of current research. In particular, the filler-based confidence modeling technique discussed in Section 5.4.1 is used in Chapter 9 for designing a novel HMM-based keyword spotting system.

5.2 Text Line Recognition

HMM-based recognition of text line images aims at finding the most probable sequence of words $\mathbf{w} = w_1, \ldots, w_N$ for a given sequence of feature vectors $\mathbf{x} = x_1, \ldots, x_T$ that is typically obtained by means of a sliding window (see Section 3.3). For finding the optimal word sequence $\mathbf{w}$, the posterior probability $P(\mathbf{w}|\mathbf{x})$ is maximized. According to Bayes' rule [59], we obtain

$$\mathbf{w} = \underset{\mathbf{w} \in \mathcal{W}}{\operatorname{argmax}} P(\mathbf{w}|\mathbf{x}) = \underset{\mathbf{w} \in \mathcal{W}}{\operatorname{argmax}} \frac{P(\mathbf{x}|\mathbf{w})P(\mathbf{w})}{P(\mathbf{x})} \tag{5.1}$$

based on the likelihood $P(\mathbf{x}|\mathbf{w})$, the prior probability $P(\mathbf{w})$, the evidence $P(\mathbf{x})$, and all possible word sequences $\mathcal{W}$[2].

Note that for handwriting recognition, the majority of the recognition systems are based on real-valued feature vectors and hence continuous HMM are employed. For continuous HMM, the likelihood $P(\mathbf{x}|\mathbf{w})$ as well as the evidence $P(\mathbf{x})$ are given by probability density function (pdf) values. In contrast, probabilities used for discrete HMM are obtained, e.g., by vector quantization [91].

[1] Although historical document recognition often takes single writer manuscripts into account, the appearance of the handwriting may change significantly within a single manuscript due to differences in ink and parchment condition (see Chapter 2).

[2] Unlike earlier publications on HMM for speech recognition [179], a more compact notation is used that hides individual HMM states and model parameters. This notation has become popular in more recent publications [174] and puts an emphasis on the language modeling component $P(\mathbf{w})$ at the word-level (see Section 5.3).

In the following, the individual components of the posterior probability are discussed in detail for text line recognition. The likelihood $P(\mathbf{x}|\mathbf{w})$ gives an answer to the question *how* the text line is written with respect to trained character models. This is the most fundamental question that can be answered by using HMM for handwriting recognition. It is discussed in this section.

The prior probability $P(\mathbf{w})$ is given by a language model and takes into account *what* is written in the text line. Language modeling has been an emerging topic in the last two decades [245] and the majority of the recognition systems nowadays include basic n-gram language models [260], which are introduced in Section 5.3.

Finally, the evidence $P(\mathbf{x})$ reflects *how well* the text line is written *in general* with respect to all possible word sequences $\mathcal{W}$. While this component is not dependent on $\mathbf{w}$ and thus can be ignored for finding the optimal word sequence, it is necessary for normalizing the recognition score in order to obtain the posterior probability. The posterior probability represents a *confidence measure* of the recognition result that can be used in several ways to improve the performance of a text line recognizer. Confidence modeling is an active area of current research [174]. Some established methods are discussed in Section 5.4.

5.2.1 Hidden Markov Models

In this section, the HMM topology and parameterization that is typically used for off-line handwriting recognition of text line images is introduced. Based on character HMM that are trained on labeled text line samples, it is shown how the likelihood $P(\mathbf{x}|\mathbf{w})$ of the feature sequence $\mathbf{x} = x_1, \ldots, x_T$ can be computed efficiently for a sequence of words $\mathbf{w} = w_1, \ldots, w_N$ in order to find the optimal word sequence

$$\mathbf{w} = \underset{\mathbf{w} \in \mathcal{W}}{\operatorname{argmax}} P(\mathbf{x}|\mathbf{w}) \tag{5.2}$$

taking into account all possible word sequences $\mathcal{W}$. Here, the result obtained is based on the appearance of the handwriting only, i.e., the correspondence with the trained character models, not taking into account language constraints on the word sequence.

The basic modeling unit is given by a character HMM. It describes two stochastic processes. The first is a first order Markov chain that produces a series of hidden, i.e., not directly observable, states $\mathbf{s} = s_1, \ldots, s_t$ from a finite set S with the Markov property

$$P(s_t|s_1, \ldots, s_{t-1}) = P(s_t|s_{t-1}) \tag{5.3}$$

That is, each state is only dependent on its direct predecessor. This process can be represented by a finite state automaton as shown in Figure 5.1 for the

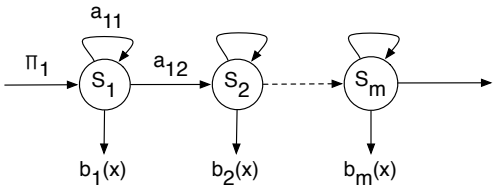


Figure 5.1: Character HMM

linear topology. Given a set of states $S = \{S_1, \ldots, S_m\}$ and starting in state S_1 with an initial state probability of $\pi_1 = P(S_1) = 1$, the model either rests in a state or changes to the next state with transition probabilities $a_{i,i} = P(S_i|S_i)$ and $a_{i,i+1} = P(S_{i+1}|S_i)$ such that $a_{i,i} + a_{i,i+1} = 1$. The linear topology shown in Figure 5.1, instead of an ergodic one where each state can be a successor to any other state, is typical for handwriting recognition. Here, the first states represent the beginning of a character, followed by the middle and end parts. Sometimes, a Bakis topology is preferred, where the skipping of one state is possible [177].

The second stochastic process that is described by the character HMM produces a series of observable feature vectors $\mathbf{x} = x_1, \ldots, x_t$ with typically continuous features $x_t \in \mathbb{R}^n$. Such an output is emitted at every time t by the current state $s_t = S_i$ and is commonly modeled by a mixture of Gaussians

$$b_i(x) = \sum_{k=1}^{G} w_{ik} \mathcal{N}(x \mid \mu_{ik}, \Sigma_{ik}) \tag{5.4}$$

where G is the number of Gaussians, $\mathcal{N}(x \mid \mu_{ik}, \Sigma_{ik})$ is a normal distribution with mean μ_{ik} and covariance matrix Σ_{ik}, and w_{ik} is the prior probability of the k-th mixture. In order to reduce the number of parameters that need to be estimated on learning samples, diagonal covariance matrices are frequently considered. Another technique involves parameter tying for so-called semicontinuous HMM, where a shared set of Gaussians is used for all states. Both techniques were used, e.g., in [252] for camera-based whiteboard reading. Here, a total of 75 characters are modeled with semicontinuous HMM that share a codebook of 1,500 Gaussians with diagonal covariance matrices.

Given a set of m states $S = \{S_1, \ldots, S_m\}$, the model parameters of a character HMM are, in summary, given by

$$\theta = (\pi, A, B) \tag{5.5}$$

where $\pi = \{\pi_i\}$ are the initial state probabilities, $A = \{a_{ij}\}$ the transition probabilities, and $B = \{b_i(x)\}$ the emission pdfs for $1 \leq i, j \leq m$.

5.2.2 Recognition Algorithms

An important reason for the widespread use of HMM for handwriting recognition is given by the availability of efficient algorithms for training and recognition. The centerpiece of these algorithms is the total or partial calculation of the observation pdf $P(\mathbf{x}|\theta)$ of the observed feature vector sequence $\mathbf{x} = x_1, \ldots, x_T$ for a set of HMM parameters θ given by

$$P(\mathbf{x}|\theta) = \sum_{\mathbf{s}\in\mathcal{S}} P(\mathbf{x}|\mathbf{s},\theta)P(\mathbf{s}|\theta) \tag{5.6}$$

with respect to all possible state sequences $\mathbf{s} = s_1, \ldots, s_T \in \mathcal{S}$, the pdf $P(\mathbf{x}|\mathbf{s},\theta)$ of a given state sequence, and its prior probability $P(\mathbf{s}|\theta)$. The state sequence specific terms are given by

$$P(\mathbf{s}|\theta) = \pi_{s_1} \prod_{t=2}^{T} a_{s_{t-1},s_t} \tag{5.7}$$

$$P(\mathbf{x}|\mathbf{s},\theta) = \prod_{t=1}^{T} b_{s_t}(x_t) \tag{5.8}$$

assuming statistical independence of the observation sequence $\mathbf{x}$ in Equation 5.8. This assumption is not really valid for handwriting recognition, since the observations are usually correlated. This is an area for potential future improvements. However, the estimation of a more complicated representation might not provide an overall better performance [123][3]. In practice, logarithmic values are typically considered in Equations 5.7 and 5.8 in order to make computations with very small values feasible.

A straight-forward calculation of $P(\mathbf{x}|\theta)$ in Equation 5.6 over all possible state sequences $\mathbf{s} \in \mathcal{S}$ would result in an exponential complexity of $O(N^T T)$ in terms of the number of states N and the observation sequence length T, which is not feasible in practice. Fortunately, the complexity can be reduced by means of dynamic programming based on the lattice representation shown in Figure 5.2 exemplarily for the linear topology[4]. Based on this lattice, the computation of $P(\mathbf{x}|\theta)$ can be done in $O(N^2 T)$ time using the Forward-Backward algorithm [12]. By this procedure, $P(\mathbf{x}|\theta)$ is

[3] An interesting comment on parameter significance mentioned in [123] (p. 269) in the context of speech recognition concerns the significance of the transition probabilities a_{ij} compared with the emission pdfs b_i. Due to the almost unlimited dynamic range of the pdfs, the b_i values tend to dominate the transition probabilities, such that the same system performance can be achieved in practice by neglecting the transition probabilities entirely, i.e., by assuming $a_{i,i} = a_{i,i+1} = 0.5$ for linear topologies. For the estimation of the emission pdfs, however, the transition probabilities still play an important role.

[4] The term *lattice* is used for the planar graph representation of the dynamic programming approach. The graph is ordered along the x-axis by time and is sometimes also called *trellis*.

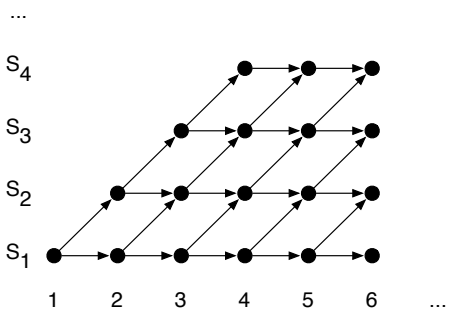


Figure 5.2: Recognition Lattice

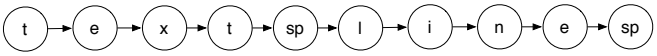


Figure 5.3: Training Text Line HMM

accumulated iteratively in the lattice over time, taking only a maximum of N parent nodes from time $t-1$ (forward) or $t+1$ (backward) into account for each node.

For training the character HMM, a text line HMM is created first as a concatenation of character models, including the special space character "sp", according to the correct transcription of a given training sample as shown in Figure 5.3 for the transcription "text line". Then, the conjoint text line model parameters θ are optimized with respect to $P(\mathbf{x}|\theta)$ from Equation 5.6, typically using the Baum-Welch algorithm [179]. This variant of the Expectation Maximization (EM) algorithm [55] iteratively adapts the model parameters in order to increase $P(\mathbf{x}|\theta)$ until the improvement becomes marginal. Note that for continuous HMM the Baum-Welch algorithm needs a few adaptations compared to discrete HMM [122]. A great advantage of this training approach is that the individual character HMM are trained conjointly and thus, no segmentation of the text line into character images is needed.

For HMM-based text line recognition, the trained character HMM are concatenated to word HMM of a given lexicon $\mathcal{L}$ of size $|\mathcal{L}| = L$. The text line can then be modeled as a large HMM containing a loop over all words that are separated from each other by the space character "sp" as illustrated in Figure 5.4. The optimal sequence of states $\mathbf{s} = s_1, \ldots, s_T$ can then be found with respect to $P(\mathbf{x}, \mathbf{s}|\theta)$ by means of the Viterbi algorithm [179]. Based on dynamic programming using the lattice representation illustrated in Figure 5.2, the score

$$\phi_t(S_j) = \max_{1 \leq i \leq N} (\phi_{t-1}(S_i) a_{ij}) b_j(x_t) \tag{5.9}$$

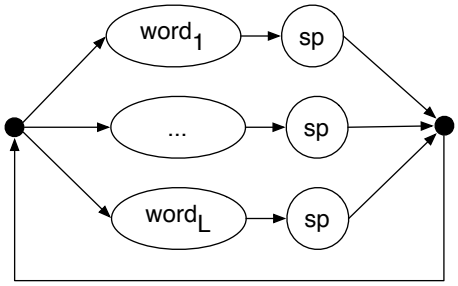


Figure 5.4: Recognition Text Line HMM

is assigned to each state S_j at time t taking into account Equations 5.7 and 5.8. Based on the initialization

$$\phi_1(S_j) = \pi_j b_j(x_1) \tag{5.10}$$

for all states S_j, the maximum observation pdf

$$P(\mathbf{x}, \mathbf{s}|\theta) = P(\mathbf{x}|\mathbf{s}, \theta)P(\mathbf{s}|\theta) = \max_{1 \leq j \leq N} \phi_T(S_j) \tag{5.11}$$

is returned alongside with the optimal sequence of states $\mathbf{s}$. From the optimal sequence of states, the optimal sequence of words $\mathbf{w}$ is finally retrieved based on the word start and end states and provides a solution to Equation 5.2. The computational time needed for Viterbi recognition of the text line is $O(L^2T)$ with respect to lexicon size L and the length T of the observation sequence. In order to speed up the process for large lexicons, pruning is often performed, e.g., by means of beam-search [129].

Variants of the text line recognition approach include the special case of single word recognition, where no loop over the words is needed, and lexicon-free recognition, where the basic modeling units are characters rather than words and an optimal sequence of characters is returned. The latter has been successfully used, e.g., in [207] for postal address reading in combination with a post-processing step to extract valid words from the output.

5.2.3 Forced Alignment

The great advantage of HMM-based text line recognition is given by the fact that no segmentation of the text line into words or characters is needed. The segmentation is performed implicitly during recognition, while finding the most likely sequence of states, and is returned as a byproduct of the recognition.

A special application of HMM-based recognition is given by *forced alignment*. In this scenario, the correct text line transcription is already known and recognition is performed to obtain the most likely character and word

segmentation. The same HMM, which is used for training (see Figure 5.3), can be used for Viterbi recognition of the training samples in order to obtain the most likely character and word segmentation. In this thesis, forced alignment is used to extract words for ground truth creation (see Chapter 7) and to extract characters for graph-based handwriting representation (see Chapter 4).

Word segmentation using forced alignment is highly accurate in the presence of clear word spacing, while character segmentation typically provides rather noisy character images. This is not surprising, since character segmentation is very difficult in case of touching characters and is often ambiguous for humans as well (see Section 1.2.3).

5.2.4 Recognition Performance

The performance of a text line recognition system is measured by the word accuracy. Hereby, the computed optimal word sequence $\mathbf{w}$ is aligned with the ground truth of test samples by means of string edit distance [249]. The word accuracy A is then given by

$$A = \frac{N - S - D - I}{N}$$

where N is the number of words in the ground truth, S the number of word substitutions, D the number of deletions, and I the number of insertions. Note that for a high number of word insertions, the word accuracy can become negative.

Important HMM parameters, which greatly affect the word accuracy and thus need to be carefully fine-tuned on a validation set in practice, are the number of Gaussian mixtures used for the emission pdf and the number of states for each character. The latter typically depends on the estimated mean character width [259].

5.3 Language Models

Based on the text line recognition approach discussed in Section 5.2, the optimal sequence of words $\mathbf{w} = w_1, \ldots, w_N$ is found with respect to the maximum likelihood $P(\mathbf{x}|\mathbf{w})$ of the feature vector sequence $\mathbf{x}$, i.e., based on the correspondence of the trained character HMM with the image. The implicit assumption of this approach is that all word sequences have the same a priori probability $P(\mathbf{w})$. Of course, this assumption does often not hold true, e.g., for natural language, where words do not occur arbitrarily in a text. Language models aim at capturing the constraints that are imposed on word sequences in a given language and can be used to improve the maximum posterior probability estimation from Equation 5.1 with

$$\mathbf{w} = \underset{\mathbf{w} \in \mathcal{W}}{\operatorname{argmax}} P(\mathbf{x}|\mathbf{w}) P(\mathbf{w}) \tag{5.12}$$

taking into account non-trivial prior probabilities $P(\mathbf{w})$ in addition to the HMM likelihood $P(\mathbf{x}|\mathbf{w})$. Including language models for HMM-based recognition can be regarded as a standard approach nowadays [177].

5.3.1 n-Gram Models

The most frequently used approach to language modeling for handwriting recognition is based on n-grams. Here, the underlying stochastic process that produces the sequence of words $\mathbf{w} = w_1, \ldots, w_t$ is a Markov chain of order $m = n - 1$ with the property

$$P(w_t|w_1, \ldots, w_{t-1}) = P(w_t|w_{t-m}, \ldots, w_{t-1}) \tag{5.13}$$

That is, each word is only dependent on its m predecessors that are also referred to as the history of w_t. The corresponding probabilities of the unigrams $P(w_t)$, bigrams $P(w_t|w_{t-1})$, trigrams $P(w_t|w_{t-2}, w_{t-1})$ and so on have to be estimated on a training corpus of texts written in the language under consideration. Examples of large publicly available corpora for modern languages include, e.g., the *Brown Corpus of Standard American English* [81] and the *Lancaster-Oslo/Bergen Corpus* (LOB) [120] for British English[5].

In practice, bigrams are the predominant type of language model for HMM-based handwriting recognition [245]. The reason is twofold. First, the additional gain in terms of word accuracy reported for n-grams of higher order, e.g., trigrams, is typically rather low [260]. Secondly, larger text corpora are needed for a reliable estimation of high order n-grams. For bigrams, the word sequence probability is given by

$$P(\mathbf{w}) = P(w_1) \prod_{i=2}^{t} P(w_i|w_{i-1}) \tag{5.14}$$

with the unigram probability $P(w_1)$ and the bigram probabilities $P(w_i|w_{i-1})$. In principle, the n-gram probabilities can simply be estimated by their relative occurrence in the training corpus. However, zero probabilities are an unrealistic estimate in case of n-grams that do not occur in the training corpus at all. Such an unseen n-gram would result in a zero probability of the complete word sequence. Therefore, several smoothing techniques have been proposed to deal with unseen n-grams, e.g., based on backing-off and interpolation [124, 126].

The bigram language model can be integrated into HMM-based recognition by modifying the score used for Viterbi recognition in Equation 5.9. With respect to logarithmic values and bigram probability $w_{ij} = P(w_j|w_i)$, the modified score is given by

$$\phi_t(S_j) = \max_{1 \leq i \leq N} (\phi_{t-1}(S_i) + \log a_{ij} + \alpha \log w_{ij} + \beta) + \log b_j(x_t) \tag{5.15}$$

[5] For historical languages, on the other hand, large text corpora with consistent orthography are often not available (see Section 1.4).

whenever the state change from S_i to S_j is a word change. Hereby, the parameters α and β are related to the integration of the language model. The factor $\alpha \geq 0$ is called *grammar scale factor* and weights the impact of the bigram word model against the likelihood of the feature vector sequence. The factor $\beta \in \mathbb{R}$ is called *word insertion penalty* and balances the number of word insertions during recognition. Both factors have a great influence on the overall word accuracy and thus need to be fine-tuned on a validation set in practice.

5.3.2 Language Model Performance

In order to evaluate the quality of a language model independently of the recognition, a frequently used criterion is given by the perplexity

$$\mathcal{P}(\mathbf{w}) = P(\mathbf{w})^{-\frac{1}{T}} \tag{5.16}$$

of an unseen word sequence $\mathbf{w} = w_1, \ldots, w_T$ that was not used for creating the language model. The perplexity reflects the cross-entropy between the language model and the word distribution in the unseen data [118]. In the worst case, each word occurs with the same probability $P(w_t) = \frac{1}{L}$ independently of its context. In this case, for lexicon size L, the resulting perplexity is L. If the language model can better predict the word sequence, a lower perplexity $1 \leq \mathcal{P}(\mathbf{w}) \leq L$ is obtained. Intuitively, the perplexity can be regarded as the lexicon reduction that is achieved by the language model.

5.4 Confidence Models

In Section 5.3, it was shown how the optimal word sequence $\mathbf{w}$ can be found for a given feature vector sequence $\mathbf{x}$ with respect to $P(\mathbf{x}|\mathbf{w})P(\mathbf{w})$, taking into account the appearance likelihood $P(\mathbf{x}|\mathbf{w})$ as well as the language model probability $P(\mathbf{w})$. In practice, the word accuracy that can be achieved with this word sequence model is regarded nowadays as still being far from perfect (see Section 1.3).

In order to improve the reliability of handwriting recognition systems for real-world applications, a common approach is to endow the recognition result $\mathbf{w}$ with a confidence $C(\mathbf{w})$, preferably with values between 0 and 1. By comparing the confidence score

$$C(\mathbf{w}) \geq T \tag{5.17}$$

with a threshold T, unreliable results with $C(\mathbf{w}) < T$ can be rejected in order to process them manually afterwards, e.g., in postal address reading [28, 64]. There are many other application areas that are dependent on a confidence score, e.g., writer identification [208], writer verification [209], combination

of multiple classifier systems [18, 115], interactive transcription [230, 236], and keyword spotting (see Chapter 9).

Several confidence measures were proposed in the handwriting recognition literature. They can be broadly classified into the three categories *heuristic*, *posterior-based*, and *likelihood ratio-based*. The first includes, e.g., likelihood stability under variation of language model parameters [18] and character length stability for different handwriting recognition systems [115]. In this chapter, we discuss the two more general approaches in greater detail, i.e., posterior-based and likelihood ratio-based.

From a Bayesian point of view, the posterior probability is a natural measure of confidence that can be used to find an optimum tradeoff between false acceptance and false rejection of recognition results with respect to the user's preference [46]. For handwriting recognition, the posterior-based confidence is given by

$$C(\mathbf{w}) = P(\mathbf{w}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{w})P(\mathbf{w})}{P(\mathbf{x})} \tag{5.18}$$

where the evidence $P(\mathbf{x})$ is needed in addition to the terms discussed so far. The computation of this confidence measure is discussed in Section 5.4.1. By comparing the posterior of the optimal solution with the posterior of a competing alternative, the frequently considered likelihood ratio-based confidence measure can be obtained. This measure is able to avoid the costly computation of $P(\mathbf{x})$. It is presented in Section 5.4.2. Finally, the Receiver Operating Characteristic (ROC) curve is introduced in Section 5.4.3 as a widely used performance measure for confidence-based recognition. With respect to this performance measure, the optimal confidence threshold from Equation 5.17 can be determined based on the user's preference [234].

5.4.1 Posterior-Based Confidence

For the posterior probability $P(\mathbf{w}|\mathbf{x})$ of the optimal sequence of words $\mathbf{w}$, the evidence $P(\mathbf{x})$, sometimes also referred to as *score normalization*, is given by

$$P(\mathbf{x}) = \sum_{\mathbf{w} \in \mathcal{W}} P(\mathbf{x}|\mathbf{w})P(\mathbf{w}) \tag{5.19}$$

It is based on all possible word sequences $\mathcal{W}$ and can be obtained efficiently by means of the Forward-Backward algorithm in $O(L^2T)$ time with respect to the size L of the lexicon and the size T of the observation sequence $\mathbf{x}$ (see Section 5.2.2)[6]. In practice, however, the computational complexity is still challenging in real-world applications with very large lexicons. Hence, approximate solutions are considered that can be found, e.g., by means of

[6] In contrast to HMM training, only the forward part of the algorithm is needed for calculating the evidence.

a beam search that takes only a limited number of word sequences with a high likelihood into account.

For text line recognition, a direct implementation of Equation 5.19 based on word graphs was recently presented in [230] for an interactive transcription system. Originally proposed for speech recognition in [251], each word sequence $\mathbf{w} \in \mathcal{W}$ is represented in a graph similar to the state-based recognition lattice discussed in Section 5.2.2. The word graph can be obtained as a byproduct of the Viterbi decoding by recording possible word endings at each time step. The nodes of the graph represent discrete points in time and the arcs are labeled with a word, its start and end position, and its likelihood. Using this word graph, the posterior probability of the text line or the posterior probability of a single word within the text line is then calculated by means of the Forward-Backward algorithm. The computational complexity of the procedure can be reduced by considering only the N-best words at each time step for creating the word graph.

For the special case of single word recognition, the observation sequence $\mathbf{x}$ of a word image can be presented to all words $w \in \mathcal{L}$ of the lexicon $\mathcal{L}$ and the evidence is simplified to

$$P(\mathbf{x}) = \sum_{w \in \mathcal{L}} P(\mathbf{x}|w)P(w) \tag{5.20}$$

taking into account the unigram probability $P(w)$[7]. In the literature, the resulting posterior $P(w|\mathbf{x})$ has been used, e.g., in [127] for word rejection and in [174] in the general context of confidence modeling for on-line handwriting recognition.

If no lexicon is available or the computational cost of computing lexicon-based posteriors is too high, an approximation of $P(\mathbf{x})$ by means of so-called *filler* models can be considered. Also known as *garbage* or *world* models, they represent general text content disregarding lexicon and language model constraints. The evidence is then approximated by

$$P(\mathbf{x}) \sim P(\mathbf{x}|\mathbf{f})P(\mathbf{f}) \tag{5.21}$$

with respect to the optimal sequence of filler models $\mathbf{f}$. Typical filler models are given by character HMM, sometimes endowed with character bigrams for non-trivial priors $P(\mathbf{f})$ [28, 61, 232]. With this approach, the computational complexity of calculating the evidence is drastically reduced to $O(A^2T)$ with respect to the alphabet size A. The posterior probability of single words within a text line can be obtained by taking into account the local filler likelihood between a word's starting and ending position. The filler-based score normalization can either be integrated into the recognition process [61, 232] or be added as a post-processing module [28]. As a variant of character

[7] For single word recognition, a uniform distribution of the words over the lexicon is often assumed. In this case, the recognition is based on the text appearance only.

HMM, a single GMM, referred to as *universal vocabulary*, has recently been proposed in [190] for score normalization in the context of keyword spotting.

5.4.2 Likelihood Ratio-Based Confidence

Instead of using the posterior probability $P(\mathbf{w}|\mathbf{x})$ directly as a confidence measure (see Section 5.4.1), there are a number of approaches known from the literature that obtain the confidence from the closely related posterior ratio between the best word sequence $\mathbf{w}_{1st}$ and the second best word sequence $\mathbf{w}_{2nd}$, e.g., in [28, 127, 156, 174]. The corresponding confidence measure is then given by

$$C(\mathbf{w}_{1st}) = \frac{P(\mathbf{w}_{1st}|\mathbf{x})}{P(\mathbf{w}_{2nd}|\mathbf{x})} = \frac{P(\mathbf{x}|\mathbf{w}_{1st})P(\mathbf{w}_{1st})}{P(\mathbf{x}|\mathbf{w}_{2nd})P(\mathbf{w}_{2nd})} \tag{5.22}$$

with respect to the likelihood ratio. An advantage of the likelihood ratio-based confidence measure is that the costly computation of the evidence $P(\mathbf{x})$ is not needed. In [156], the likelihood ratio-based confidence is investigated in the context of on-line text line recognition. Based on a word graph, obtained by Viterbi decoding with a beam search, the likelihood ratio to the second best word hypothesis is used, among other approaches, as a confidence measure for individual words within the text line. For the special case of single word recognition, the same confidence modeling approach is pursued, e.g., in [28, 127, 174].

Another approach to likelihood ratio-based confidences is given by statistical hypothesis testing. According to Neyman-Pearson's Lemma [166], the likelihood ratio $\frac{f(x|\theta_0)}{f(x|\theta_1)}$ with respect to a probability density function $f(x|\theta)$ with parameter θ provides an optimal test for the null hypothesis $H_0 : \theta = \theta_0$ and alternative hypothesis $H_1 : \theta = \theta_1$. For HMM-based handwriting recognition, this test can be stated as

$$C(H_0) = \frac{P(\mathbf{x}|\theta_0)}{P(\mathbf{x}|\theta_1)} \tag{5.23}$$

with respect to the observation likelihood from Equation 5.6 based on the HMM parameters θ_0 and θ_1 of the null and alternative hypotheses, respectively. In practice, an approximation of Equation 5.23 is used based on the likelihood of the most probable sequence of HMM states that can be obtained by Viterbi decoding. Another approximation is given by the fact that the true probability density function is unknown and has to be estimated from the training data assuming a Gaussian mixture model. Hence, the test is not optimal, but has proven to be very successful, in particular in the field of speech recognition [192].

When the optimal word sequence $\mathbf{w}_{1st}$ is considered as the null hypothesis and the second best word sequence $\mathbf{w}_{2nd}$ as the alternative hypothesis,

Equation 5.23 is, in fact, approximated by Equation 5.22. In the handwriting recognition literature, there are a number of reports that use different alternative models, often referred to as *anti-models* or *cohort models*, to obtain a confidence measure. In [208], different writer-specific HMM are trained for the task of writer identification and the likelihood ratio between the best and the second best writer model is used as a confidence measure. An additional writer-independent anti-model was used in [209] for confidence modeling in the context of writer verification.

Also, the filler models discussed in Section 5.4.1 have been used frequently as general anti-models for text, e.g., in [28, 127, 156, 174]. In fact, the same confidence measure

$$C(\mathbf{w}) = \frac{P(\mathbf{x}|\mathbf{w})P(\mathbf{w})}{P(\mathbf{x}|\mathbf{f})P(\mathbf{f})} \tag{5.24}$$

results for the likelihood ratio as for the posterior approximation taking into account Equations 5.18 and 5.21.

In general, it is difficult to predict which confidence model will perform best for a given task, considering different constraints on available training data and computation time. Instead, the best model is typically chosen on a validation set with respect to the system's performance. This issue will be discussed in the next section.

5.4.3 Confidence Model Performance

The performance of a confidence-based recognition system is measured by its ability to identify and reject unreliable recognition results. Given a confidence value, the result is accepted only if the value exceeds a certain threshold and is rejected otherwise. For evaluation, the numbers of correctly accepted (CA), falsely accepted (FA), correctly rejected (CR), and falsely rejected (FR) samples are used to calculate the false acceptance rate (FAR) and false rejection rate (FRR) as

$$FAR = \frac{FA}{FA + CR} \tag{5.25}$$

$$FRR = \frac{FR}{FR + CA} \tag{5.26}$$

These two quantities define the Receiver Operating Characteristic (ROC) curve [234]. Based on different thresholds, a trade-off between FAR and FRR can be achieved that is suitable for a given application. For example, in a postal address reading application, it might be necessary to achieve a very small false acceptance rate in order to prevent wrong deliveries, at the cost of a higher false rejection rate that leads to an increased amount of manual post-processing.

Chapter 6

Neural Networks

Besides the generative approach to handwriting recognition using hidden Markov models (see Chapter 5), we consider a discriminative approach based on neural networks (NN) in this thesis.

In the handwriting recognition literature, NN are frequently used in the context of single character recognition [47, 116, 135, 136]. However, when it comes to sequential problems such as text line recognition HMM are clearly predominant. A major challenge for NN-based recognition is given by the fact that class labels are needed for each vector in the feature sequence (see Chapters 3 and 4) for a straight-forward network training. Again, the basic problem of Sayre's paradox is encountered which states that segmentation is needed for recognition and recognition is needed for segmentation (see Section 1.2.3).

Most NN-related approaches for sequential problems such as speech recognition [185], on-line handwriting recognition [17], and off-line handwriting recognition [65] are, in fact, hybrid systems between HMM and NN. Two well-known strategies include feature extraction using NN [17] and modeling the HMM emission pdf with NN [185, 65]. In case of emission modeling, an initial HMM system is required to perform a forced alignment (see Section 5.2.3) in order to provide class labels for each vector in the feature sequence. The class labels are then used to train a discriminative NN which provides posterior probabilities $P(c|x)$ of the character c for the feature vector x. After an appropriate scaling, an emission value is obtained and several alignment iterations are repeated until convergence. The actual text line recognition is finally performed by the HMM.

Very recently, a purely NN-based solution for handwriting recognition has been proposed in [96] using a special form of recurrent neural networks that process the feature vector sequence bidirectional and employ long short-term memory cells instead of standard network units (BLSTM). As a main difference to the hybrid systems, no HMM component is needed, neither for training nor for recognition. This is achieved by an algorithmic framework

called connectionist temporal classification (CTC) which is able to deal with Sayre's paradox. Based on CTC, the NN-based recognizer can be applied in exactly the same manner as HMM for handwriting recognition. That is, text line images do not need to be pre-segmented into characters, neither for training nor for recognition. In contrast to HMM, discriminative recognition is performed with respect to character posteriors $P(c|x)$ instead of the generative pdf $P(x|c)$. Statistical language models can be integrated in the same way as for HMM.

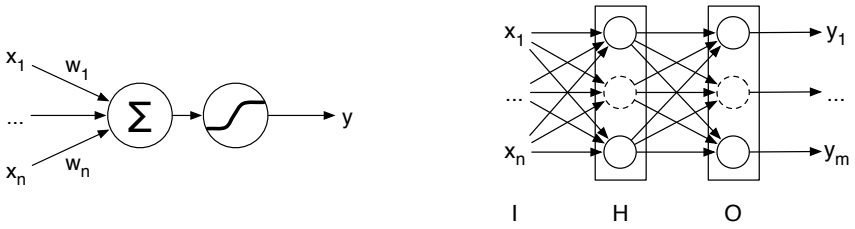
In this chapter, we analyze the CTC layer from an HMM oriented point of view and find that it bears a close resemblance to the text model and algorithmic framework underlying HMM. In fact, CTC is based on the same lattice structure used for HMM training and recognition (see Figure 5.2). Same as for HMM, the lattice is processed with the Forward-Backward algorithm [12] for training (sum of paths in the lattice) and with a Viterbi-like forward pass for recognition (best path in the lattice). Similar links have been observed for example in [25] in the context of HMM-NN hybrid systems. We believe that such links contribute to a better understanding of the underlying recognition-segmentation challenge for sequential data.

Based on this CTC interpretation, we have developed a novel algorithm for NN-based recognition, which is an approximative solution of the optimal Viterbi-like forward pass. Using a special ϵ-compression of the feature vector sequence, which is specific to the BLSTM output layer, and a token passing approach with rigorous pruning we can achieve a significant speedup over the optimal solution while maintaining the recognition accuracy. In addition, the algorithm returns a word recognition lattice rather than only the best sequence of words, which makes the output more versatile, e.g., for confidence modeling [230] and classifier combination [18].

The remainder of this chapter is organized as follows. First, the network architecture of the BLSTM networks is described in Section 6.1. Next, the CTC framework for training and recognition is discussed in Section 6.2. Finally, the proposed token passing algorithm for text line recognition is presented in Section 6.3. Experimental results are provided in Chapter 8 in the context of automatic transcription.

6.1 BLSTM Networks

In this section, we discuss the basic structure and function of BLSTM networks. Starting with multilayer perceptrons, the properties of recurrence, bidirectional processing, and memory cells are described. The review strictly focuses on the models and algorithms underlying the BLSTM networks proposed in [96]. For a general introduction of NN, we refer to [116] which also includes the application scenario of NN-based single character recognition.



(a) Perceptron (b) Layers

Figure 6.1: Multilayer Perceptron

6.1.1 Multilayer Perceptrons

Artificial neural networks were originally devised as a computational model of biological neurons (nerve cells) [157, 194, 197]. A single artificial neuron, also known as perceptron [194], is shown in Figure 6.1a. It receives several real-valued inputs $x_1, \ldots, x_n$, weights them differently, builds the sum $a = \sum_1^n w_i x_i$, and returns an output $y = \theta(a)$ based on some activation function θ. This abstraction corresponds with a single biological neuron which receives impulses from connected neurons depending on the strength of the connections and emits impulses depending on the activation potential reached.

In pattern recognition, NN have been successfully used as parametric models for function approximation, most notably for approximating the class posterior probability in order to perform discriminative pattern classification. For pattern recognition with BLSTM networks, the first relevant network architecture is the multilayer perceptron [197]. It has been proven that a multilayer perceptron is, under some conditions, a universal function approximator [104]. That is, it is principally able to approximate any function with arbitrary precision.

A standard network architecture with three layers is illustrated in Figure 6.1b. The input layer includes n feature values representing a pattern and propagates them to each perceptron in the hidden layer. Based on some activation function θ_H, their output is propagated to the output layer which contains a perceptron for each pattern class. Based on a second activation function θ_O, the final output $y_1, \ldots, y_m$ is considered for classification. Because of the forward direction of the data flow such architectures are also called feedforward neural networks. A typical activation function for the hidden layer is the logistic sigmoid

$$\theta_H(a) = \frac{1}{1 + \exp(-a)} \tag{6.1}$$

which maps large negative network inputs a to an activation close to 0 and large positive inputs to an activation close to 1. In the output layer, a

softmax function

$$\theta_O(a_i) = \frac{\exp(a_i)}{\sum_{i=1}^{m} \exp(a_i)} \tag{6.2}$$

is often applied such that the activations can be interpreted as the posterior probability $y_i = P(C_i|x)$ of class C_i for the input features $x = (x_1, \ldots, x_n)$ with $\sum_{i=1}^{m} y_i = 1$.

The goal of network training is to find network weights w such that the y_i correspond with the class posterior probabilities. Assuming equal weight priors the maximum likelihood objective for the learning samples $(x_1, C_1), \ldots, (x_N, C_N)$ is to maximize $\prod_{i=1}^{N} P(C_i|x_i, w)$ or, equivalently, minimizing the objective function

$$O = -\sum_{i=1}^{N} \ln P(C_i|x_i, w) = -\sum_{i=1}^{N} \ln y_i \tag{6.3}$$

This can be achieved by means of gradient descent and backpropagation [197]. For a training sample (x_i, C_i), the partial derivative of the objective function with respect to the network input a_j of an output perceptron is

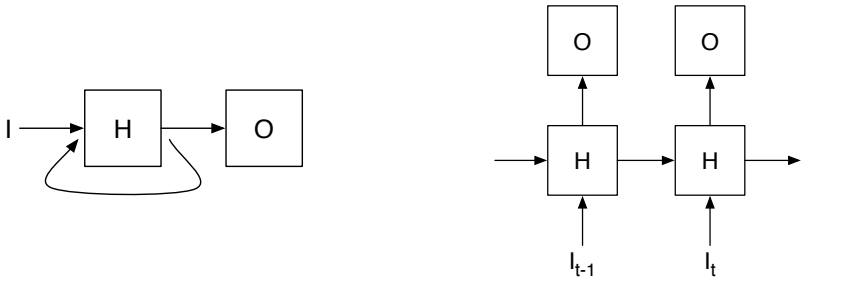
$$\frac{\partial O}{\partial a_j} = y_j - \delta_{i,j} \tag{6.4}$$

where $\delta_{i,j}$ is 1 for $i = j$ and 0 otherwise. This gradient indicates the direction in which the network input a_j needs to be changed in order to improve the objective function. For $i = j$ the output should be close to 1 and for $i \neq j$ the output should be close to 0. The gradient is then back-propagated over the sum of weights and the differentiable activation function θ_H in the hidden layer in order to calculate all weight gradients in the network.

Based on weight gradients, training is performed as follows. First the weights are randomly initialized. Then, the weight gradients $\frac{\partial O}{\partial w(1)}$ are calculated for the first training epoch. In batch mode, the gradients are summed up over all samples with respect to Equation 6.3. In online mode, weight gradients are considered for each sample individually. The weights are iteratively updated over several training epochs according to

$$\Delta w(i) = m\Delta w(i-1) - \eta \frac{\partial O}{\partial w(i)} \tag{6.5}$$

with respect to the weight change of the previous epoch, the momentum m, and the learning rate η with $0 \leq m, \eta \leq 1$ that control the magnitude of the weight changes. A fine-tuning of m and η can help to avoid local minima during training which is a well-known problem for gradient descent algorithms. Training is repeated until some convergence criterion is satisfied. In order to avoid overfitting to the training samples, the classification error on an independent validation set is typically taken into account. Training is stopped if the validation error cannot be further reduced over several epochs.



(a) Architecture (b) Backpropagation Through Time

Figure 6.2: Recurrent Neural Networks

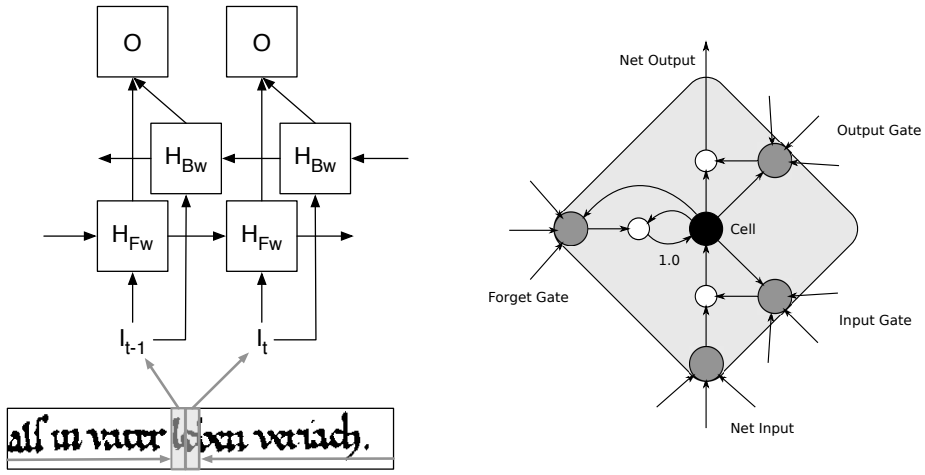
6.1.2 Recurrent Neural Networks

In contrast to feedforward networks, recurrent neural networks contain loops that return perceptron activations back into the network. In Figure 6.2a this is illustrated for a single hidden layer which is connected to itself, i.e., all output activations of the hidden units are returned back as a network input. This architecture is well-suited for sequential problems where the input is given by a sequence $x_1, \ldots, x_T$ of feature vectors. It allows to access contextual information since each feature vector has an influence on all subsequent vectors.

For training BLSTM networks, backpropagation through time [250] is used as illustrated in Figure 6.2b. In order to determine the weight gradients for the feature vector input I_t at time t, an additional input from the previous time step is considered for backpropagation. Starting at time $t = T$ and decrementing t by one at each step, the weight gradients are summed up over the complete sequence and are finally used for updating the weights.

Two elements complete the architecture of BLSTM networks. First, bidirectional processing of the feature vector sequence is achieved by using two distinct hidden layers [212], one for forward processing and one for backward processing as illustrated in Figure 6.3a. For handwriting recognition, a text line is read from both directions and the output layer receives input from the forward layer H_{FW} as well as the backward layer H_{BW} at each position within the text line. Training can be realized again by means of backpropagation through time for each layer separately.

The final element are the long short-term memory cells (LSTM) [102] which are used instead of single perceptron units. They address the vanishing gradient problem, i.e., the exponential increase or decrease of values depending on the weights assigned to loop connections. LSTM cells consist of several perceptrons that control the cell value which is stored in a central node as illustrated in Figure 6.3b. The activation of the input gate unit determines whether the net input is added to the cell value, the output gate



(a) Bidirectional Processing (b) Long Short-Term Memory Cell

Figure 6.3: Bidirectional LSTM Neural Networks

determines whether the cell value is propagated into the network, and the forget gate determines whether the cell value is kept in the next time step. All units are standard perceptrons that can be trained as usual. However, the overall complexity of the architecture is considerable. For details on the activation functions and backpropagation equations we refer to [95].

So far, the implicit assumption for training BLSTM networks was that a class label and hence an objective function is available for each input vector. In hybrid systems this can be obtained by means of HMM-based forced alignment. The connectionist temporal classification framework (CTC) [96] provides an elegant alternative with respect to a global objective function as discussed in the next section.

6.2 Text Line Recognition

In contrast to HMM (see Section 5.2), the discriminative NN-based approach determines the optimal sequence of words $\mathbf{w} = w_1, \ldots, w_N$ within the feature vector sequence $\mathbf{x} = x_1, \ldots, x_T$ based on posterior probabilities instead of likelihoods. The optimal sequence of words with respect to $\mathbf{x}$ and a statistical language model L is

$$\mathbf{w} = \underset{\mathbf{w} \in \mathcal{W}}{\operatorname{argmax}} P(\mathbf{w}|\mathbf{x}, L) = \underset{\mathbf{w} \in \mathcal{W}}{\operatorname{argmax}} \frac{P(\mathbf{w}|\mathbf{x})P(\mathbf{w}|L)}{P(\mathbf{w})} \tag{6.6}$$

applying rules of conditional probabilities and assuming an independence of $\mathbf{x}$ and L. This assumption is not true in general but is justifiable if the language model is estimated on external corpora. Further assuming that

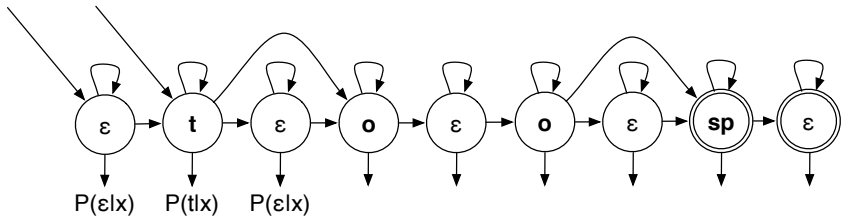


Figure 6.4: CTC Text Model

the unconditioned prior $P(\mathbf{w})$ is equal for all word sequences, it is proposed in [96] to consider

$$\mathbf{w} = \underset{\mathbf{w} \in \mathcal{W}}{\operatorname{argmax}} P(\mathbf{w}|\mathbf{x}) P(\mathbf{w}|L) \tag{6.7}$$

for finding the optimal sequence of words. The language model term $P(\mathbf{w}|L)$ corresponds with the n-gram term discussed in Section 5.3.1. The text appearance term $P(\mathbf{w}|\mathbf{x})$ is independent of the language model and is obtained by means of BLSTM networks. In the following, the training procedure is discussed in Section 6.2.1 and text line recognition in Section 6.2.2.

6.2.1 Training

The text model underlying the connectionist temporal classification (CTC) framework is an alternating sequence of "no character" denoted with ϵ and text characters. We interpret it as a finite state automaton shown in Figure 6.4 exemplarily for the word "too" followed by a space character "sp". Starting either in the first ϵ-state or the state "t", the automaton either stays in the same state, changes to the next state, or skips an ϵ-state. An exception is given for two identical characters where the ϵ-state cannot be skipped in order to distinguish a change of character. The last two states are end states. The automaton describes valid character sequences $\mathbf{s} = s_1, \ldots, s_T$ for a given sequence length T. All character sequences are valid that arrive at an end state after T transitions. In the example, the only valid character sequence $\mathbf{s} = s_1, \ldots, s_5$ for $T = 5$ would be t, o, ϵ, o, sp.

The introduction of an explicit "no character" state between two text characters is a new concept which differs from traditional HMM text models. Apart from that, the automaton resembles the text model for HMM-based recognition (see Figure 5.1) when considering a single state per character, a linear topology, and additional transitions to skip states. In contrast to the HMM text model, the emission pdf $P(x|s)$ of the state s is replaced with the probability $P(c|x)$ of the character c and the transition probabilities are removed from the model.

The output layer of the BLSTM networks contains an output unit for each character of the alphabet and an additional unit for the special ϵ char-

acter. The output activation y_c^t of the character c at time t is interpreted as the character probability $P(c|x_t)$. Due to the softmax activation function in the output layer the character probabilities indeed sum up to 1 over the complete alphabet A which includes the ϵ character. The probability of a character sequence $\mathbf{s} = s_1, \ldots, s_T$ is then given as

$$P(\mathbf{s}|\mathbf{x}) = \prod_{t=1}^{T} P(s_t|x_t) = \prod_{t=1}^{T} y_{s_t}^t \tag{6.8}$$

with respect to the $|A|^T$ possible character sequences for alphabet size $|A|$. For a labeled training sample $(\mathbf{x}, \mathbf{w})$, the probability of the word sequence from Equation 6.7 can now be expressed as

$$P(\mathbf{w}|\mathbf{x}) = \sum_{\mathbf{s} \in \mathcal{S}(\mathbf{w})} P(\mathbf{s}|\mathbf{x}) = \sum_{\mathbf{s} \in \mathcal{S}(\mathbf{w})} \prod_{t=1}^{T} y_{s_t}^t \tag{6.9}$$

with respect to all valid state sequences $\mathcal{S}(\mathbf{w})$ of the automaton that corresponds with the word sequence $\mathbf{w}$. It can be calculated by means of a forward pass of the Forward-Backward algorithm over the recognition lattice that corresponds with the automaton. The lattice has exactly the same structure as for HMM-based recognition (see Section 5.2.2). An example is shown in Figure 6.5a for the automaton of "too". The shaded area corresponds with all admissible paths through the lattice.

For training, the global objective function for a training sample $(\mathbf{x}, \mathbf{w})$ is given as usual as

$$O = -\ln P(\mathbf{w}|\mathbf{x}) \tag{6.10}$$

For training a specific network output y_c^t at time t the probability is split up at time t over all characters of the alphabet

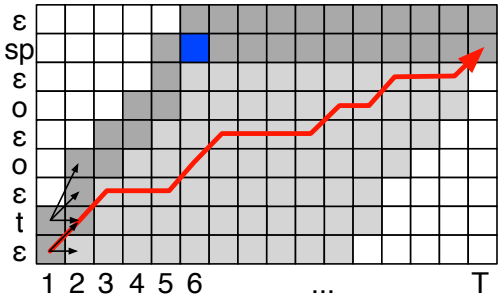
$$P(\mathbf{w}|\mathbf{x}) = \sum_{c \in A} \sum_{\mathbf{s} \in \mathcal{S}_{c,t}(\mathbf{w})} P(\mathbf{s}|\mathbf{x}) \tag{6.11}$$

where $\mathcal{S}_{c,t}(\mathbf{w})$ includes all valid paths that go through character c at time t. For characters that do not appear in the word sequence this set is empty. For example, the darker shaded area in Figure 6.5a corresponds with the area of all paths in $\mathcal{S}_{sp,6}(too)$. The sum $\sum_{\mathbf{s} \in \mathcal{S}_{c,t}(\mathbf{w})} P(\mathbf{s}|\mathbf{x})$ can be efficiently calculated by means of the Forward-Backward algorithm with respect to logarithmic values.

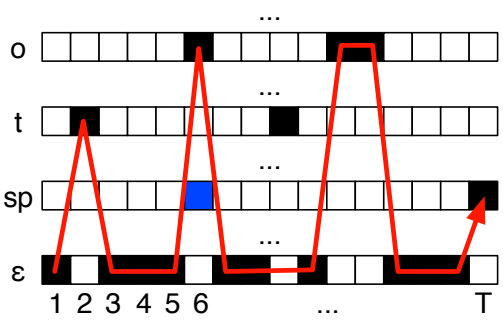
As shown in [96], the split objective function can be differentiated for individual output units which results in the following gradient

$$\frac{\partial O}{\partial a_c^t} = y_c^t - \frac{\sum_{\mathbf{s} \in \mathcal{S}_{c,t}(\mathbf{w})} P(\mathbf{s}|\mathbf{x})}{\sum_{\mathbf{s} \in \mathcal{S}(\mathbf{w})} P(\mathbf{s}|\mathbf{x})} \tag{6.12}$$

which can be backpropagated through time. For example, the goal value for y_{sp}^6 corresponds with the probability mass in the darker shaded area in Figure 6.5a divided by the total probability mass.



(a) Recognition Lattice



(b) Network Output

Figure 6.5: Text Recognition

6.2.2 Recognition

Based on a trained network, text line recognition is achieved by a Viterbi-like forward pass through a large recognition lattice that includes all words of the lexicon. In principle, the same algorithm is performed as discussed in Section 5.2.2 for HMM. With respect to logarithmic values, Equation 5.9 is modified to

$$\phi_t(S_j) = \max_{1 \leq i \leq N} (\phi_{t-1}(S_i)) + \log y^t_{S_j} \tag{6.13}$$

removing the state transition term and replacing the emission pdf with the character probability $y^t_{S_j}$. The initialization in Equation 5.10 is changed to

$$\phi_1(S_j) = \log y^1_{S_j} \tag{6.14}$$

for all start states S_j of a word. Bigram language models are integrated by modifying Equation 5.15 to

$$\phi_t(S_j) = \max_{1 \leq i \leq N} (\phi_{t-1}(S_i) + \alpha \log w_{ij} + \beta) + \log y^t_{S_j} \tag{6.15}$$

whenever the state change from S_i to S_j is a word change. Note that the grammar scale factor α and the word insertion penalty β are only added for the sake of completeness. As in [96] we use $\alpha = 1$ and $\beta = 0$ in this thesis. This makes sense since the dynamic range of $P(\mathbf{w}|\mathbf{x})$ and $P(\mathbf{w}|L)$ is the same in Equation 6.7. Using different language model parameters could be an area of future improvements.

6.3 Token Passing Algorithm

In [96], a token passing approach to text line recognition is presented. Token passing [256] is a memory-efficient implementation of dynamic programming that loads only one column of the recognition lattice in the memory, calculates the transitions to the next column in form of token values, and keeps only the best token at each state according to the dynamic programming rule. Then, the column values are replaced with the new tokens and the process is iterated over $1 \leq t \leq T$. Like this it is avoided to load the complete matrix in the memory which can become problematic for large lexicons.

Considering a complete column, the recognition algorithm used in [96] provides an optimal solution for finding the best path through the lattice. However, this leads to computational problems for large lexicons underlying natural language which typically include tens of thousands of words. Since each word end is connected with each word start in the recognition lattice the computational complexity of the recognition is $O(N^2T)$ with respect to the number of words in the lexicon N and the length of the feature vector sequence T. As mentioned in [96], this worst case complexity is seldom encountered since the word end tokens can be sorted. The expected complexity is $O(N \log(N)T)$ which is still problematic as demonstrated in Chapter 8 in the context of automatic transcription. Another limitation of the algorithm is that it only provides the single best sequence of words. A more versatile output such as n-best sentences or even word lattices are needed, e.g., for confidence modeling [230] and classifier combination [18].

In this section, we present a fast approximative solution for BLSTM-based recognition with bigram language models which can overcome these limitations. The algorithm includes several components. First, we have devised an ϵ-compression specifically for BLSTM networks. Due to the close relation between HMM-based and NN-based recognition, we have furthermore integrated two best practices for large vocabulary decoding with HMM, i.e., prefix organization of the search space and token pruning [10, 99]. The output of the algorithm is a word lattice as desired.

In the remainder of this section, the data structures of the algorithm are described in Section 6.3.1 and a pseudo code is provided in Section 6.3.2. An experimental evaluation of the algorithm is presented in Chapter 8 in the context of automatic transcription.

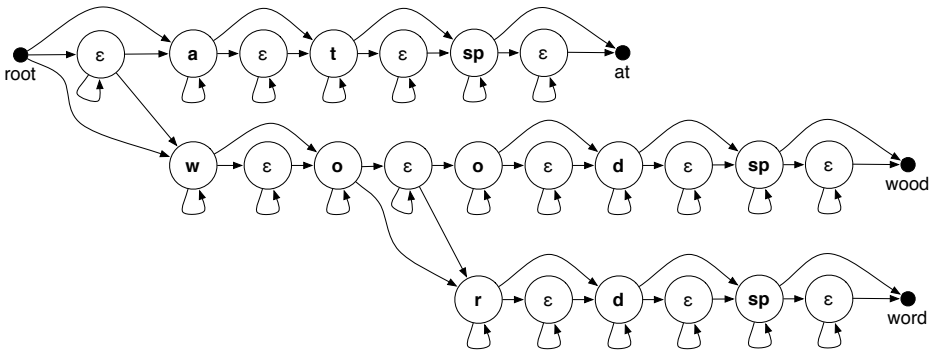


Figure 6.6: Text Line Prefix Graph

6.3.1 Data Structures

The proposed algorithm employs a compressed observation sequence, a prefix graph, and a word lattice as the basic data structures.

For handwriting recognition, the BLSTM network output is rather sparse in our experience. Most of the time, only the ϵ-node is active with a probability close to one. It is interrupted by character peaks as illustrated in Figure 6.5b for decoding the word "too". We use this property to perform an ϵ-compression where consecutive ϵ-activations with a probability $y_\epsilon^t > \theta$ are compressed to a single activation

$$\hat{y}_\epsilon^{a,b} = \prod_{t=a}^{b} y_\epsilon^t \tag{6.16}$$

which reduces the overall number of activations that need to be processed. For $\theta = 99.9\%$ we can achieve compression factors between 6 and 10 for the data sets considered in this thesis (see Chapter 8). In the data structure, the length of the compressed activations is stored in order to reconstruct correct word boundaries after recognition.

The lattice column is organized as a text line prefix graph as illustrated in Figure 6.6. By sharing word prefixes, the lattice column can be stored more compactly and the search space of dynamic programming is reduced. As a special condition of the CTC text model, no transition is allowed between two subsequent characters if they are equal such as the two "o" in "wood". The graph has two special types of nodes. First the root node which is used to insert new tokens and secondly, word leaf nodes which represent word ends. Note that the CTC text model is more compact than the typical HMM text model. Instead of several states per character only one state is needed.

The third data structure is given by a word recognition lattice illustrated

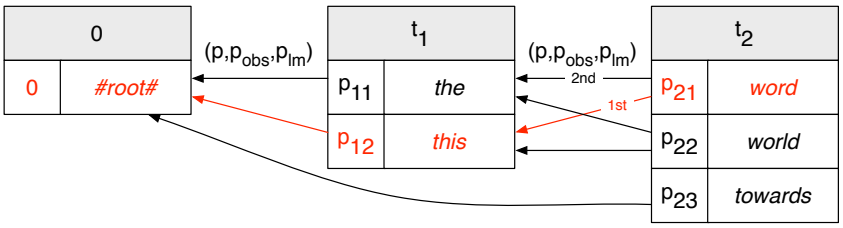


Figure 6.7: Word Recognition Lattice

in Figure 6.7. The lattice is generated during recognition and stores possible word endings. The words are linked with edges to reconstruct complete word sequences. Each edge is labeled with the observation probability p_{obs} of the word that was accumulated over the activations y_s^t within the prefix graph, the bigram probability p_{lm} with respect to the preceding word, and the overall probability p that has been accumulated since the word start. This probability is used to sort the words such that the word with the highest probability $p_{t,1}$ appears at the beginning of the word list. The edges are sorted as well such that the optimal sentence can be instantly traced back as soon as $t = T$ is reached.

6.3.2 Algorithm

A pseudo code of the proposed token passing approximation is given in Algorithm 2. Input is the prefix graph g according to the lexicon, the network outputs *obs* for each character and time step, and the bigram language model lm. Furthermore, the grammar scale factor α, the word insertion penalty β, and two pruning parameters can be provided. We consider a rigorous histogram pruning to n_τ best tokens at each time step and a maximum of n_ω word ends that are recorded in the lattice.

The initialization in Lines $1 - 6$ adds a root entry to the lattice at time $t = 0$ and creates a token list *tokens* that contains a single root token. It has a log-probability $p = 0$, an entry time into the prefix graph $t = 0$, and is linked by reference to the root node of the prefix graph.

The main loop in Line 7 iterates over the length T of the compressed observation sequence. At each time step, an empty list *newTokens* is prepared that collects all tokens that are passed in this step.

The token passing within the prefix graph is performed in Lines $9 - 14$. Each token in the current *tokens* list is passed to the successor nodes in the prefix graph. In Line 11, its probability value is updated with the observation log-probability of the successor node at the current time[1]. Passing a

[1] An exception for this update are leaf nodes in the prefix graph for which the token value remains unchanged.

token means to link it to the new graph node by reference and add it with an updated probability value to the $newTokens$ list. When all tokens have been processed, the $newTokens$ list is sorted and only the n_τ most probable tokens are kept.

In Lines $16 - 26$ leaf tokens are collected that represent word endings. In Line 19, the observation probability that was accumulated within the prefix graph is obtained by subtracting the probability at graph entry time. Then, edges are created to all words that were recorded at graph entry time. The language model log-probability is weighted with α and β, stored in the edge, and added to the total probability. Finally the word is added to the $newWords$ list. Note that α and β are only added to the algorithm for the sake of completeness. In this thesis, we use the defaults $\alpha = 1$ and $\beta = 0$ (see Section 6.2.2).

The $newWords$ are added to the lattice in Lines $27 - 31$. Only the n_ω best words are stored. If new words were added to the lattice a new root token is inserted into the prefix graph with the best probability over all new words, the current time as root entry time, and a link to the root node of the prefix graph. The root token is inserted by adding it to the $newTokens$ list.

Finally, the $newTokens$ become the $tokens$ in Line 32 and the procedure is repeated until time T is reached. In Line 34 a final cleanup of the lattice is performed by means of graph search starting from the line end in order to remove unreachable words. Then the lattice is returned as the final result.

For a fixed number of n_ω words stored at the line end, the time complexity of the algorithm is $O(M \log(M) T)$ where $M = n_\nu$ is the number of tokens. This results from the loop over all tokens in Line 9 and the sorting in Line 15. In Chapter 8 we show that the optimal accuracy of text line recognition can be maintained with only relatively few tokens even for lexicons that include up to 20000 words. This achieves a considerable speedup and yields a more versatile result in form of a word lattice when compared to the baseline algorithm. We believe that we can work with such a rigorous pruning because of the compact search space with respect to compressed observation sequences and a single state per character in the CTC text model.

Algorithm 2 Text Line Recognition

Require: text line prefix graph g, network outputs obs, bigrams lm
Require: bigram weights α,β, n-bests n_τ, n_ω
Ensure: word recognition lattice l

$t \leftarrow 0$
$p \leftarrow 0$
$l.words(t) \leftarrow \{\}$
add $(\#root\#)$ to $l.words(t)$
$tokens \leftarrow \{\}$
add $(p, t, g.root)$ to $tokens$
for $t = 1 \rightarrow T$ **do**
  $newTokens \leftarrow \{\}$
  for all τ in $tokens$ **do**
    for all η in $\tau.node.next$ **do**
      $p \leftarrow \tau.p + \log obs(\eta, t)$
      add $(p, \tau.start, \eta)$ to $newTokens$
    end for
  end for
  sort $newTokens$ based on $\tau.p$ and keep the n_τ best
  $newWords \leftarrow \{\}$
  for all leafs τ in $newTokens$ **do**
    $\omega \leftarrow \tau.node.word$
    $p_{obs} \leftarrow \tau.p - l.best(\tau.start)$
    for all $\hat{\omega}$ in $l.words(\tau.start)$ **do**
      $p_{lm} \leftarrow \alpha \log lm(\hat{\omega}, \omega) + \beta$
      $p \leftarrow \tau.p + p_{lm}$
      add $(\hat{\omega}, p, p_{obs}, p_{lm})$ to $\omega.edges$
    end for
    add ω to $newWords$
  end for
  if $newWords$ not empty **then**
    sort $newWords$ based on $\omega.best$ and keep the n_ω best
    $l.words(t) \leftarrow newWords$
    add $(l.best(t), t, g.root)$ to $newTokens$
  end if
  $tokens \leftarrow newTokens$
end for
finalize l
return l

Part II

Historical Manuscript Recognition

Chapter 7

Ground Truth Creation

In this chapter, we discuss the creation of training samples for handwriting recognition systems such as the HMM and NN investigated in Chapter 8 for automatic transcription. The goal is to annotate handwriting images with machine-readable text, also called the *ground truth* of the machine learning task. The ground truth is usually created with human supervision to ensure that clean, error-free training samples are obtained. Besides its primary use to train recognizers, ground truth is also needed for evaluating the performance of a recognizer.

Annotating scanned manuscripts with machine-readable transcriptions at page level is, unfortunately, not sufficient as ground truth for handwriting recognition. Instead, the page images have to be segmented into individual text lines or words that need to be annotated individually with their transcription. In contrast to printed documents, a segmentation into individual characters is usually not feasible for handwritten documents (see Section 1.2.3). State-of-the-art handwriting recognition systems such as the HMM and NN considered in this thesis do not rely on character segmentation prior to training and recognition. Therefore, ground truth annotations at character level are not necessarily needed.

For modern handwriting, ground truth creation can be realized efficiently by means of special forms that are filled in by contemporary writers. For instance, the writers can be asked to copy given texts into individual boxes for text lines, words, and characters. After scanning the forms, the handwriting within the boxes can be extracted in a straight-forward manner and the corresponding texts can be assigned fully automatically. In principle, this allows for an efficient generation of large amounts of training samples. Human supervision is only needed to check the final results.

For historical handwriting, on the other hand, ground truth creation is rendered more difficult by the fact that training samples have to be extracted from actual manuscripts. In this scenario, human supervision is needed for all steps of the document analysis task, i.e., layout analysis, image segmen-

tation, and text assignment. Even if a transcription is available at page level for scanned manuscripts this supervision is a time-consuming task and a major obstacle for mass digitization of historical manuscripts. Methods are needed that allow for an efficient ground truth creation with a minimum of human interaction.

In this chapter, a ground truth creation procedure is presented that was developed for the creation of the IAM-HistDB (see Chapter 2). Several hundred page images of historical manuscripts are considered for which a machine-readable text edition was already available at page level. A sequence of several automatic and manual processing steps is proposed that aims at minimizing human interaction for the extraction of annotated text line and word images. A particular aim of the procedure is that laypersons without special knowledge in computer science or linguistics are able to perform all required manual interactions.

The chapter is structured as follows. First, related work is discussed in Section 7.1. Then, the IAM-HistDB is described in Section 7.2 including unpublished parts that are not discussed in Chapter 2. Afterwards, Section 7.3 presents the ground truth creation step by step. Section 7.4 comments on the resulting ground truth annotations and the storage and time expenses for the IAM-HistDB. Finally, Section 7.5 summarizes the chapter and provides an outlook to future research.

7.1 Related Work

In this section, we discuss related ground truth data sets for training and testing handwriting recognition systems. For the purpose of research, the public availability of ground truth, for instance in form of free downloads from the Internet, is of crucial importance. It supports the development of novel recognition systems and allows for a comparison of different recognizers on the same data. Section 7.1.1 lists a number of data sets that are readily available for modern handwritings. Afterwards, Section 7.1.2 discusses some of the firsts data sets that have become available for historical handwritings. The small number of such data sets has motivated the creation of the IAM-HistDB in the context of this thesis.

7.1.1 Modern Handwriting

In case of modern handwriting, ground truth data is readily available for a number of data sets.

For on-line data, which is acquired with special writing devices to capture time information of the writing process, the first data sets included UNIPEN [98] and IRONOFF[243] containing labeled isolated digits, letters, and words in Western scripts. Later on, the data sets focused on whole sentences such as the IAM-OnDB [142] containing a large number of labeled

English sentences. Also, data sets for non-Western scripts became available, for instance Japanese [163] and Indian [20].

For off-line data, which is acquired from handwriting images, one of the first available data sets was CEDAR [109] containing labeled images of isolated, address related words and numbers, for instance city names, state names, and ZIP codes in Western scripts. More recent data sets were focused on whole sentences such as the IAMDB (see Section 2.4) containing English sentences. Also, non-Western scripts became available, for instance Arabic [169].

7.1.2 Historical Handwriting

The availability of the data sets mentioned in Section 7.1.1 had a huge impact on the development of recognition technologies for modern handwriting and allowed a comparison of different systems on established ground truth data. In case of historical documents, however, only few data sets have become readily available yet. The reason for this is twofold. First, the application domain is rather new, and secondly, the ground truth creation is more difficult and time-consuming than for modern documents since actual manuscripts have to be considered rather than special acquisition forms. Special page layouts and background noise in case of degraded paper and parchment render the task of text image extraction more difficult. Available transcriptions are often not aligned with the text lines of the document images. If no transcription is available at all, another problem is given by the fact that only experts can perform the time-consuming transcription of special old languages, while for modern documents laypersons are able to perform this task.

One of the first readily available data sets for handwriting recognition in historical documents was the George Washington database (see Section 2.3) containing 20 pages of George Washington's letters. It was made available by Rath et al. [183] and has been used by several research groups for automatic transcription and keyword spotting (see Chapters 8 and 9).

With the development of interactive annotation tools, more data sets are currently being created, which will hopefully provide a larger scope of historical scripts and languages in the future. Examples for interactive annotation systems include the DEBORA system used for Renaissance documents [24], the DMOS system used for old civil status registers and military forms [50], and the STATE [93] and CATTI [191] systems used for old Spanish manuscripts.

Closely related to the ground truth creation method presented in this thesis is the GIDOC[1] system [215], which has been used to annotate several hundred page images of old Spanish manuscripts, for instance the pub-

[1] Gimp-based Interactive transcription of old text DOCuments, available at `http://prhlt.iti.upv.es/page/projects/multimodal/idoc/`.

Table 7.1: Medieval German manuscripts.

Manuscript	Century	Folia	Pages
Cod. 857	13th	323	646
Cod. AA 91	15th	180	360

licly available GERMANA database [171]. GIDOC provides semi-automatic methods for layout analysis, text line detection, and transcription. In particular, handwriting recognition systems are employed to support manual transcription by proposing automatic transcription results. In terms of human effort, creating machine-readable transcriptions is the bottleneck of ground truth creation. An average of 30 minutes per manuscript page is reported for a manual transcription of the GERMANA database by experts in paleography [171].

In contrast, we focus on manuscripts for which a machine-readable text edition is already available. In this scenario, semi-automatic ground truth creation is based on text alignment rather than manual transcription. Taking into account layout analysis, text line detection, and transcription alignment, we report an average interaction time of 5.5 minutes per manuscript column for the proposed ground truth creation of the IAM-HistDB. Laypersons were able to perform all manual interactions.

7.2 IAM-HistDB

The IAM Historical Handwriting Database (IAM-HistDB) primarily contains medieval manuscripts of the epic poem *Parzival* by Wolfram von Eschenbach, one of the most significant epics of the European Middle Ages. The corresponding old German manuscripts are discussed in Section 7.2.1. In order to incorporate other scripts and languages, the IAM-HistDB was also extended to Latin and English manuscripts discussed in Section 7.2.2.

7.2.1 German Manuscripts

There exist multiple manuscripts of the poem *Parzival*, written with ink on parchment or paper, that differ in writing style and dialect of the old German language. We consider two manuscripts[2] listed in Table 7.1. Together, the manuscripts contain about 500 sheets (folia) with a front page and a back page, each. The Cod. 857 is kept in the Abbey Library of Saint Gall,

[2]Originally, a third manuscript was taken into account as well [73]. However, because we could not obtain the permission to make the corresponding data publicly available online, we abandoned the work on this manuscript.

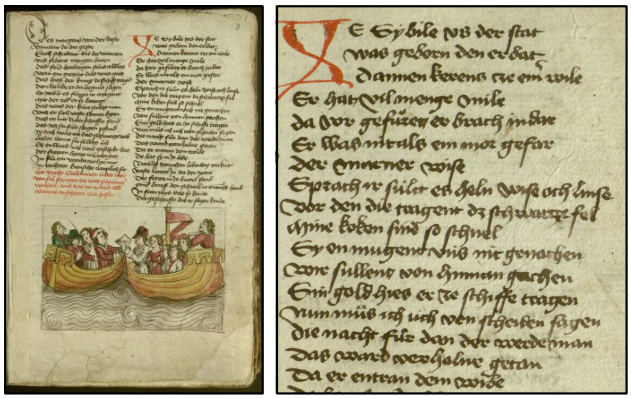


Figure 7.1: Cod. AA 91

Switzerland, and the Cod. AA 91 in the Burgerbibliothek Bern, Switzerland. Besides the *Parzival* poem, the Cod. 857 also contains the famous *Nibelungenlied*.

Exemplary images of the Cod. 857 are shown in Figure 2.2 of Chapter 2 for the already published PARDB that includes 47 pages of the manuscript. The Cod. AA 91 has not been used for handwriting recognition so far and has not been made publicly available yet. An exemplary page is shown in Figure 7.1. The physical dimension of the manuscripts is about 31×21 cm. They are arranged in two columns of pairwise rhyming lines that correspond in most cases with the verses of the poem.

The IAM-HistDB includes about 500 manuscript pages for which a transcription is available. The transcriptions were acquired by the German Language Institute of the University of Bern (see Section 2.2). On the institute's *Parzival* project website[3], more detailed information about the manuscripts and their transcriptions can be found.

7.2.2 Latin and English Manuscripts

The same ground truth creation procedure was used to annotate 60 pages of the Cod. Sang. 562 held by Abbey Library of Saint Gall, Switzerland, as well as 20 pages of George Washington Papers held by the Library of Congress. The corresponding data sets, i.e., the SGDB and the GWDB, are detailed in Sections 2.1 and 2.3.

The ground truth creation method has been adapted several times based on user feedback in order to reduce the human effort and to improve the quality of the ground truth. The SGDB is the only database published so far that incorporates all features presented in this chapter. For instance, the

[3] `http://www.parzival.unibe.ch/`

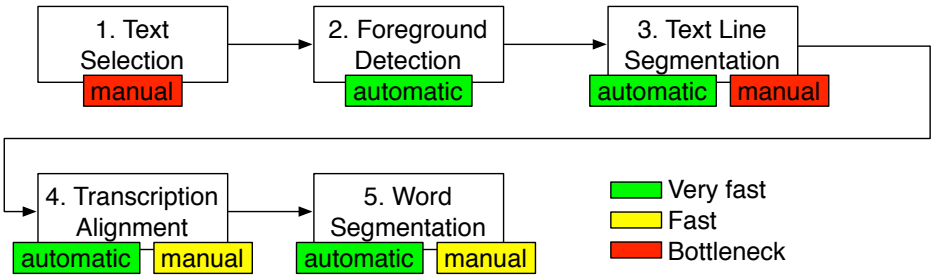


Figure 7.2: Workflow

text line locations in the original manuscript images were, unfortunately, not recorded for the PARDB and GWDB. While these locations are not strictly necessary for training and testing handwriting recognition systems, they are valuable for research on automatic layout analysis and text line extraction.

7.3 Procedure

The proposed semi-automatic procedure for creating the ground truth of the IAM-HistDB consists of five consecutive processing steps. An overview of the workflow is shown in Figure 7.2. For each step, it is indicated whether automatic processing, manual interaction, or both are required. The bottleneck in terms of processing time are the manual interactions in the first and the third step. In the following, the individual processing steps are detailed in Sections 7.3.1– 7.3.5.

7.3.1 Text Selection

As the first step, contiguous text areas are manually selected with bounding polygons on each document page. In most cases, two columns are selected per page. Titles and paragraphs are included in the columns, i.e., they are not selected separately. Special cases are paragraphs covering two columns and captions of drawings. The polygon selection avoids ornaments, drawings, decorated initial letters, page numbers, and annotations on the margin of the page that were added later to the manuscript. For the purpose of layout analysis, these elements could be selected in a similar way and added to the ground truth in the future.

For polygon selection, the *Paths* tool of the GIMP[4] software is used and the selections are saved as Scalable Vector Graphics (SVG). While the selection is not very tight around the text areas for the purpose of fast

[4] `http://www.gimp.org/`

Figure 7.3: Text Selection

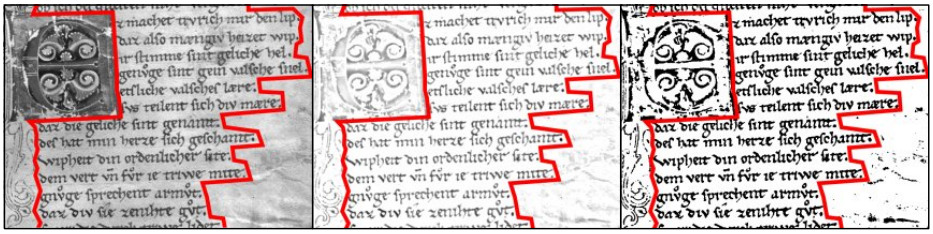


Figure 7.4: Foreground Detection

processing, it avoids stains, holes and other artifacts on the parchment that would impede the subsequent foreground detection. In Figure 7.3, the text selection using GIMP is illustrated.

7.3.2 Foreground Detection

After the manual selection of text areas, the text foreground is detected automatically. The text foreground is needed for text line segmentation (see Section 7.3.3) as well as for handwriting recognition based on binary features (see for instance Sections 3.3.2 and 3.3.3).

First, grayscale document images are obtained by luminance. Then, a Difference of Gaussians (DoG) filter is applied to locally enhance edges and hence the text foreground. Hereby, the image is first blurred with two Gaussian kernels having different standard deviation radii σ_1 and σ_2. Then, one of the blurred images is subtracted from the other. If the two radii σ_1 and σ_2 are chosen carefully, most of the parchment background noise, e.g., stains and ink bleed-through, can be removed and the text foreground can be accentuated. The text foreground of the document image is then obtained by binarization with a global threshold T and the selected text

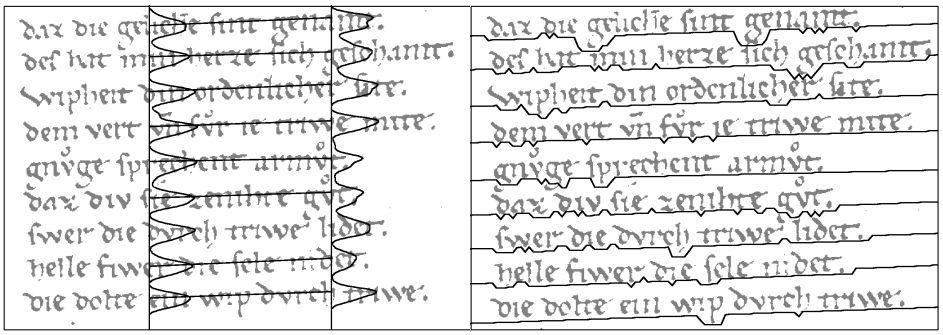


Figure 7.5: Text Line Segmentation

areas are cut out from the binarized image.

For each manuscript of the IAM-HistDB, the radii σ_1 and σ_2 as well as the binarization threshold T are determined manually and are used for all manuscript pages. Hereby, a good tradeoff has to be found between reducing pepper noise and maintaining text detail. Remaining background noise typically includes holes and stitches within the text areas. Figure 7.4 shows a binarization example for $\sigma_1 = 40.0$, $\sigma_2 = 2.0$, and $T = 215$. The grayscale image is displayed on the left, the result of the local edge enhancement using DoG in the middle, and the global thresholding result on the right.

For the SGDB and the GWDB, text foreground detection was less challenging. For these data sets, Sauvola's binarization method [204] is used without prior edge enhancement. Each pixel is assigned to the text foreground if its intensity exceeds a threshold that is dependent on the local pixel intensity distribution. We have used a local window of 20×20 pixels and applied a median filter after binarization in order to remove some remaining noise.

7.3.3 Text Line Segmentation

Once binary images of the text areas are obtained, the text is segmented automatically into text lines based on an approach that was originally proposed for online handwritten documents in [143]. The resulting piecewise linear boundary between two adjecent text lines is then subject to manual correction in a graphical user interface.

Automatic Segmentation The text line segmentation is performed in two steps. In the first step, the starting point and the skew, i.e., the horizontal inclination, of each text line is calculated based on histogram analysis. For the starting points, a histogram resulting from the horizontal projection of the black pixels is calculated for the left part of the text, taking only a few

initial letters of each text line into account. The starting points of the text lines are then given by the maxima of the histogram. For skew estimation, two additional histograms are calculated in the middle of the text area as illustrated in Figure 7.5 (left). For each text line, the skew is then given by the slope of the connection between two maxima.

In the second step, the starting points and the skew of the text lines are used to calculate a piecewise linear separating path by means of a search procedure. First, the start of the separating path is chosen in the middle between two text line starting points. Then, new points are added to the path in the direction of the text line skew at regular distance intervals. For avoiding to hit any of the two adjacent text lines, dynamic programming is used to optimize the position of the new points. Hereby, the cost function of the path position is influenced by the deviation from the direction of the skew, the distance to the foreground pixels, and a penalty for crossing a text line. In Figure 7.5 (right), an example of the resulting segmentation is shown. Typical errors occur for large descenders, for instance for the letter *g*. On the first line in Figure 7.5, this leads to a suboptimal split of two touching lines at the line end. On the fifth line, two *g*s are cut at the beginning of the line. For more details on the text line segmentation, we refer to [143].

Manual Correction The text line segmentation is implemented by a Java application, which enables manual correction of the proposed segmentation path in a graphical user interface. In Figure 7.6, two images of a selected text column are displayed in the left and middle column, respectively. The first image is a JPG compressed partial image from the original document image and the second one is the binarization result. Both images have the same dimensions, given by the bounding box around the polygon chosen in the text selection stage. The text line segmentation result is overlaid on the images and can be corrected directly for the binarized image or, alternatively, for the color image.

After correction, the individual text lines are cut out from the binarized image for further processing. Typical problems that need manual correction include touching lines, letters with large descenders or ascenders, and special marks above a letter as illustrated in Figure 7.5 (right). In the graphical user interface, two consecutive text lines share a common polygon boundary for the purpose of fast processing. For highly overlapping text lines, two different boundaries might be necessary.

7.3.4 Transcription Alignment

To train a handwriting recognition system, the available transcription needs to be exactly aligned with the segmented text line images. Hereby, line breaks have to be adjusted and the transcription must be stored in plain

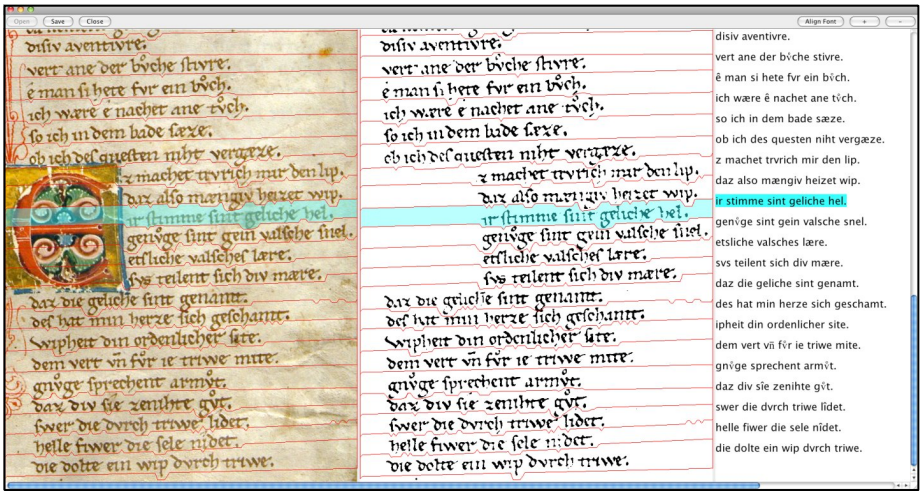


Figure 7.6: Transcription Alignment

text, such that each letter in the image corresponds to one single character in the transcription. In case of multiple interpretations of characters or words, the most probable interpretation according to the experts in linguistics is chosen for the IAM-HistDB.

Automatic Alignment The HTML transcription of the IAM-HistDB manuscripts (see Section 2.2) is parsed and stored in Unicode plain text. Special medieval letters that are not part of the Latin alphabet are already given by a Unicode character in the HTML transcription. All elements that are not included in the selected text area, e.g., decorated initial letters and annotations on the margin of the page, are removed automatically as well as abbreviations that are written out in the transcription but appear as a special character in the text image. All information necessary for this automatic removal is given in the richly annotated HTML transcription. Line breaks are initially inserted after each verse of the poem. They correspond in most cases with the line breaks in the document image.

Manual Correction The parsed plain text transcription is integrated into the Java GUI application for manual correction of the alignment as illustrated in Figure 7.6. In the right column, the transcription is displayed in a text editor. The text lines are distributed equally over the height of the text image by adapting font size and line spacing for visual alignment. Also, the text line transcription is synchronized with the text line images to facilitate line break correction. Note that both the correction of text line

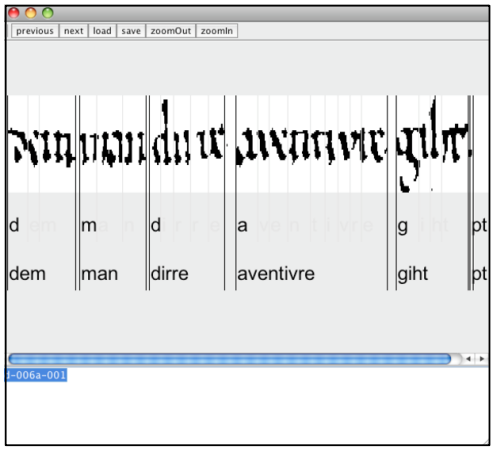


Figure 7.7: Word Segmentation

segmentation and transcription alignment is performed at the same time using the depicted GUI application.

For the SGDB and GWDB, more human interaction was necessary to obtain clean transcription alignment results. For the GWDB, line breaks and word breaks at the line end had to be adapted in the text editor. For the SGDB, the electronic text edition (see Section 2.1) deviates from the correct transcription in several aspects. For instance the capitalization, punctuation, and word abbreviation had to be adapted manually in the text editor. Automatic procedures for such an alignment are discussed in Chapter 10. They could be integrated into the ground truth creation in the future.

7.3.5 Word Segmentation

The last step for ground truth creation of the IAM-HistDB is the segmentation of text lines into words. Following the approach proposed in [258], an HMM recognition system is used to automatically determine word boundaries that are then corrected manually if necessary.

Automatic Segmentation For HMM-based word segmentation, forced alignment is performed according to Section 5.2.3. We consider the text line normalization from Section 3.2.2, the geometric features from Section 3.3.2, and the HMM architecture from Section 5.2.1. The number of states per character HMM are adopted from experiments on text line recognition, which are detailed in Chapter 8. Using a single Gaussian $G = 1$ for the emission model of the HMM has proven to perform well for word segmentation. As a result, the word start and end positions are returned.

Table 7.2: Ground truth format.

Ground Truth Item	Description	Format
Transcription	Unicode character sequence	TXT
Text line locations	Bounding polygon	SVG
Word locations	Word start and end positions	TXT

Manual Correction The results from the HMM-based forced alignment are displayed in a Java application, which enables manual correction in a graphical user interface. Only a few corrections are necessary, because the accuracy of the forced alignment typically is near 100%. In Figure 7.7, the Java application is illustrated for an example text line. The user can move the word boundaries to correct the alignment.

7.4 Results

The ground truth creation procedure presented in Section 7.3 returns processed text images alongside with ground truth annotations. In the following, Section 7.4.1 comments on the format of the ground truth and Section 7.4.2 gives an overview of the storage and time expenses of the procedure.

7.4.1 Ground Truth Format

Table 7.2 summarizes the resulting ground truth annotations. Transcriptions are stored as a sequence of Unicode characters for each text line. In particular, special medieval letters are encoded with a single Unicode character as well. The text line transcriptions provide the central ground truth for line-based handwriting recognition systems (see Chapters 8, 9, and 10). For the SGDB, non-abbreviated word labels from the electronic text edition are added additionally to provide the ground truth for automatic transcription alignment systems (see Chapter 10).

The text line locations are stored as Scalable Vector Graphics (SVG) in form of closed polygon paths. The polygons are obtained by combining the line separation paths from Section 7.3.3 with the text areas from Section 7.3.1. The SVG provide the ground truth for automatic layout analysis and text line segmentation systems. Also, the SVG allow to extract unnormalized text line images from the original pages for line-based handwriting recognition systems.

Finally, the word locations are given as start and end positions within normalized text lines. They provide the ground truth for automatic word segmentation systems. Also, the locations allow to extract word images from

the normalized text lines for word-based handwriting recognition systems. Unfortunately, the normalization cannot be changed in this scenario for different handwriting recognition systems. As a future extension, we therefore envisage to reverse the normalization and provide SVG for word locations in the same way as for text lines.

7.4.2 Storage and Time Expense

The main storage expense of the IAM-HistDB is given by the original document images, i.e., 17 MB per page for the Cod. 857 and 88 MB per double page for the Cod. AA 91. For the ease of online distribution, we provide compressed high-quality JPGs for the SGDB and PARDB (see Chapter 2). The processed text images include normalized text lines and words. These binary images as well as the ground truth annotations have a size of only few KB per page, text line, or word.

The main time expense for ground truth creation lies in the manual selection of text areas, the correction of text line segmentation, and the correction of the transcription alignment. For the IAM-HistDB, text area selection took about $T_1 = 2$ minutes per column on average, and the correction of the segmentation together with the correction of the transcription alignment about $T_2 = 3.5$ minutes per column if the user was familiar with the task. Whenever the transcription needed manual adaptation, for instance for line break correction in case of the *Nibelungenlied* in the Cod. 857, T_2 rapidly increased to about 12 minutes per column.

7.5 Summary and Outlook

In this chapter, a semi-automatic ground truth creation procedure for handwriting recognition in historical documents is presented. It takes into account noisy background that is typical for historical documents and produces a ground truth that can be used for the development and assessment of various pattern recognition methods, e.g., layout analysis, text line segmentation, automatic transcription, keyword spotting, and transcription alignment. The ground truth creation is demonstrated for the IAM-HistDB consisting of several hundred old German, Latin, and English manuscript pages for which a machine-readable text edition was already available at page level.

A small set of algorithmic tools is used for the automatic part of the ground truth creation process that includes DoG-supported binarization, text line segmentation using histogram analysis and dynamic programming, parser-based transcription alignment, and HMM-based word segmentation.

Manual interactions involve the selection of text areas on the original image, the correction of line and word segmentation, and the correction of transcription alignment. Based on simple graphical interfaces, laypersons

are able to perform these manual interactions efficiently. The average time needed for manual interactions for the ground truth creation of the IAM-HistDB was about 5.5 minutes per text column.

Future work includes the integration of automatic transcription alignment methods that are needed to cope with inaccurate text editions (see Chapter 10). If no text edition is available at all, automatic transcription methods could be integrated in an interactive way to reduce the effort of manual transcription. A particular demand that was raised by the users of our ground truth creation procedure was that the automatic methods should learn from the corrections made by the human users in order to avoid the same mistake after few manual corrections. For instance, the automatic line segmentation frequently commit errors in the presence of large descenders. The dynamic programming used for line segmentation could be enhanced by integrating corrections made by the human users in the objective function of the line separation path.

Chapter 8

Automatic Transcription

The ultimate goal of handwriting recognition in historical documents is to transcribe the handwriting in scanned manuscripts fully automatically into computer-readable text. In this chapter, experimental evaluation results are presented for automatic transcription of the IAM-HistDB (see Chapter 2) with two state-of-the-art recognition approaches. First, a generative approach with hidden Markov models (HMM) detailed in Chapter 5 and secondly, a discriminative approach with a special form of neural networks (NN) described in Chapter 6.

The experimental evaluations serve three purposes. First, they constitute a comprehensive assessment of automatic handwriting recognition for different historical scripts and languages. With respect to digital library projects, such feasibility studies are clearly needed and they are still very rare in the research literature. Secondly, the two recognition approaches are compared with each other. While HMM are well-established in the field, the NN used in this thesis were introduced a few years ago and have only rarely been applied to handwriting recognition so far. Finally, the reported results serve as a baseline for the newly introduced IAM-HistDB. They can be used for comparison with other handwriting recognition techniques in the future.

Besides a standard configuration of the recognition systems we also investigate several promising extensions. Two feature extensions include the application of linear and non-linear feature selection methods as well as the novel structural descriptor based on graph similarity (see Chapter 4). An improvement of NN-based text line recognition is achieved with a newly introduced token passing algorithm (see Chapter 6) which is a fast approximation of the optimal recognition algorithm. It is evaluated with respect to its approximation error and computational speedup.

The remainder of this chapter is structured as follows. First, the evaluation results of the standard system configuration are presented in Section 8.1. Then, the feature extensions are discussed in Section 8.2 and the NN extension in Section 8.3. A summary and outlook is provided in Section 8.4.

8.1 Text Recognition

In this section, HMM-based and NN-based handwriting recognition is evaluated for single word and text line recognition. Single word recognition (SWR) performs a classification of pre-segmented word images with respect to a lexicon of words. Text line recognition (TLR) returns a sequence of recognized words for complete text line images.

Two important assumptions underly the general assessment scenario. First, we consider an error-free extraction of word and text line images from scanned documents in order to focus on handwriting recognition. That is, the reported results should be interpreted as an upper bound with respect to automatic text extraction. In this scenario, TLR is the more difficult task because the segmentation of text line images into words has to be performed during recognition while for SWR word images are available.

Secondly, we focus on the transcription of single manuscripts without external sources of information. This is a typical scenario in case of special historical scripts and languages for which external training samples may not be available, neither for text appearance modeling nor for language modeling (see Section 1.4). It is assumed that some learning samples have been extracted manually or semi-automatically beforehand from the manuscript (see Chapter 7). They are used to train the recognition system and to validate various system parameters. The recognition performance is then measured on an independent set of test images representing the rest of the manuscript that has to be transcribed automatically.

For language modeling, we assume equal word probabilities in case of SWR. This is a useful condition to assess the recognition performance with respect to the text appearance only. For TLR we additionally take unigram and bigram language models (see Section 5.3.1) into account that are estimated on the training set. The lexicon of words that actually appear in the manuscript is not known in general. Without access to external lexicons, any word that does not appear in the training set is out-of-vocabulary (OOV) and cannot be recognized in our assessment scenario. Because it is difficult to compare word accuracies for different OOV conditions, we consider a closed vocabulary scenario which is frequently used in the literature. That is, the lexicon used for recognition includes all words from the whole manuscript. The results reported in this section should therefore be interpreted as the expected performance for known words. Note that only the word label is assumed to be known. Character appearance models as well as language models are, of course, estimated on the training set alone.

In summary, we present single manuscript recognition results with respect to error-free text image extraction and a known lexicon of words. This evaluation provides an insight into the general capabilities of current handwriting recognition systems and establishes baseline results for the newly introduced IAM-HistDB.

Table 8.1: Database statistics. Number of text lines in the training, validation, and test set.

Database	Training	Validation	Test
SGDB	468	235	707
PARDB	2237	912	1328
GWDB	325	168	163

Table 8.2: Language model statistics. Number of characters in the alphabet, words in the lexicon, and the bigram perplexity P on the test set.

Database	Alphabet	Lexicon	P
SGDB	49	5762	1653.0
PARDB	93	4934	399.4
GWDB	82	1471	183.9

The remainder of this section is organized as follows. First, the characteristics of the data sets are described in Section 8.1.1. Next, the configuration of the HMM and NN recognizers are discussed in Sections 8.1.2 and 8.1.3. The experimental results are then presented in Section 8.1.4 and a discussion is finally provided in Section 8.1.5.

8.1.1 Data Sets

For experimental evaluation, we consider the three historical data collections of the IAM-HistDB, that is, the Saint Gall database (SGDB), the Parzival database (PARDB), and the George Washington database (GWDB). Each database is split into three disjunct sets for training, validation, and testing as listed in Table 8.1. For the PARDB, the text lines of the three sets are equally distributed over all manuscript pages. For the SGDB, the first 30 pages are used for training, the next 10 pages for validation, and the remaining 20 pages for testing. Similarly, the first 10 pages of the GWDB are used for training, the next 5 pages for validation, and the remaining 5 pages for testing. Further database statistics are detailed in Chapter 2.

For language modeling, word unigrams and bigrams are estimated on the training and validation set. In case of word breaks at the line end, both parts are treated as individual words. Punctuation marks are added to the preceding word. In order to cope with unseen words, Kneser-Ney smoothing is applied [126] and the vocabulary is closed over the whole database. Table 8.2 lists the language model statistics. The test set perplexity can be

interpreted as a lexicon reduction that is achieved by the language model (see Section 5.3.2). For the bigram language model, unigram probabilities are considered for the first word of each line. The language models were created using the SRILM toolkit[1].

8.1.2 HMM Setup

For HMM-based recognition, we employ character HMM with a linear topology (see Section 5.2.1). For the PARDB and the GWDB, the number of states $\alpha \cdot w_c$ is a fraction of the mean character width w_c which is estimated for each character separately based on a few samples. For the SGDB, a fixed number of states S are used. The emission model of the character HMM is given by a mixture of G Gaussians with diagonal covariance matrices. All model parameters, i.e., α, S, and G, are optimized on the validation set with respect to the recognition accuracy.

For text representation, we use the nine geometric features described in Section 3.3.2. The HMM are trained with the Baum-Welch algorithm and text recognition is performed with the Viterbi algorithm (see Section 5.2.2) using the HTK toolkit[2]. Statistical language models are integrated with respect to a grammar scale factor GSF and a word insertion penalty WIP (see Section 5.3.1), which are optimized on the validation set.

Note that both HMM and NN are segmentation-free recognizers in the sense that no segmentation of text line images into words or characters is needed prior to training and recognition. Single word recognition can be considered as a special case of text line recognition.

8.1.3 NN Setup

For NN-based recognition, the networks are trained with online gradient descent, a learning rate of 10^{-4}, and a momentum of 0.9 as proposed in [96] (see Chapter 6). 10 neural networks with a fixed number of 100 LSTM cells are randomly initialized for each database and are trained until the character accuracy on the validation set has not improved over 10 training epochs. Results are reported for the best network out of ten, which is selected with respect to its recognition accuracy on the validation set.

Handwritten text is represented with the same nine geometric features used for HMM. They are normalized to zero mean and unit variance. Text recognition is performed with the optimal token passing algorithm used in [96] (see Chapter 6). No language model parameters are employed, i.e., $GSF = 1$ and $WIP = 0$.

[1] `http://www.speech.sri.com/projects/srilm/`
[2] `http://htk.eng.cam.ac.uk/`

Table 8.3: Word recognition errors in percentage. Single word recognition SWR with uniform word probability and text line recognition TLR with uniform word probability, word unigrams, and word bigrams.

Database	System	SWR	$TLR_{uniform}$	$TLR_{unigrams}$	$TLR_{bigrams}$
SGDB	NN	**4.0**	7.3	6.3	**6.2**
	HMM	6.4	13.3	10.8	10.6
PARDB	NN	**8.9**	7.7	7.0	**6.7**
	HMM	14.1	21.6	17.6	15.5
GWDB	NN	**19.7**	22.0	19.3	**18.1**
	HMM	30.5	31.2	27.5	24.1

8.1.4 Results

The test performance of the recognition systems is shown in Table 8.3. The word error rate is indicated for single word recognition (SWR) as well as text line recognition (TLR) with three different language models, i.e., uniform word probability, word unigrams, and word bigrams. For both SWR and TLR the best results are marked in bold. The SWR results and bigram TLR results are visualized in Figure 8.1.

The overall best performance is achieved for single word recognition of the Carolingian minuscules from the Saint Gall database where only 4% of the words are wrongly recognized by the neural networks. The second best result is achieved for the Gothic script from the Parzival database for which an error rate of 6.7% is reported for text line recognition with neural networks and bigram language models. The same setting achieves the best word error of 18.1% for the longhand script of the George Washington database.

In general, text line recognition performs worse than single word recognition when uniform word probabilities are assumed. This is due to the fact that text line recognition has to find word boundaries within the text line images in addition to word recognition, which imposes an additional difficulty. An exception is given for the SGDB, where the best network trained on text line images performs better than the best network trained on single word images.

When using statistical language models for text line recognition, the performance can be increased significantly and in some cases, single word recognition can even be outperformed. Improvements of unigrams and bigrams over uniform word probabilities are statistically significant in all cases (t-test, $\alpha = 0.05$). Improvements of bigrams over unigrams are significant in three cases (PARDB-HMM, GWDB-HMM, and PARDB-NN).

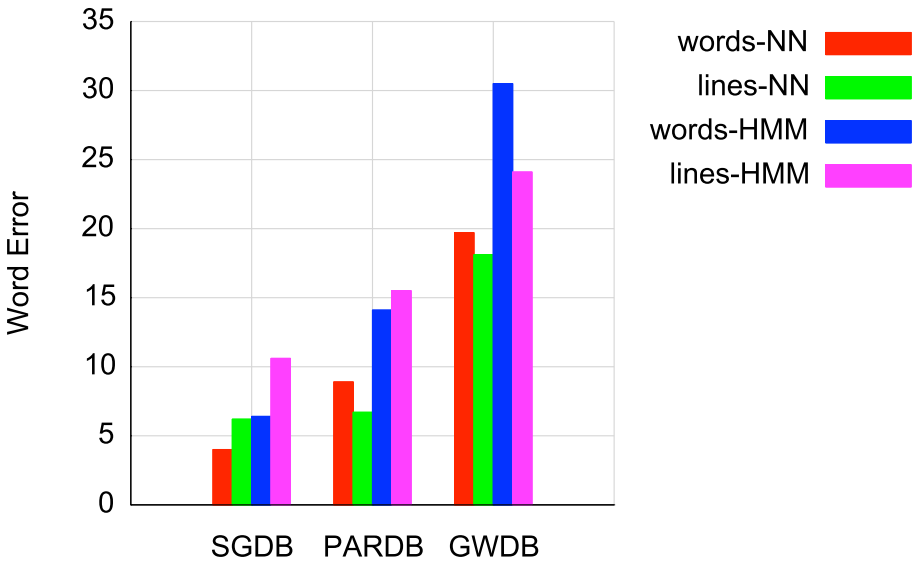


Figure 8.1: Text Recognition Results

The improvements of NN-based recognition over HMM-based recognition are statistically significant in all 12 cases.

The optimization of the most important system parameters on the validation set is documented in Appendix A. Figure A.1 shows the influence of the number of Gaussian mixtures for HMM-based single word recognition as well as the gap between the average neural network and the selected best network. Figure A.2 documents the optimization of the grammar scale factor for HMM-based text line recognition. The optimization of a word insertion penalty in addition to the grammar scale factor improves the results. Finally, Figure A.3 shows the decrease of the character error for neural network training over several epochs. The risk of overfitting becomes evident for all data sets. The error vanishes almost completely on the training set for the SGDB and the PARDB while the validation error cannot be further reduced at some point.

8.1.5 Discussion

Based on a certain amount of training samples from single manuscripts, the neat and clean handwriting style encountered in the IAM-HistDB leads to relatively high recognition accuracies despite difficult image conditions and relatively small training sets. Especially the 4% error rate for the Carolin-

Table 8.4: Related GWDB results for word recognition without OOV.

System	Training Pages	Word Error
Lavrenko et al. [134]	10	39.4
Lavrenko et al. [134]	19	34.9
Adamek et al. [3]	19	17.4
Howe et al. [106]	19	16.0
NN	10	18.1
HMM	10	24.1

gian script of the Saint Gall database and the 6.7% for the Gothic script of the Parzival database are promising for content-based indexing of historical manuscripts in digital libraries.

However, it is important to note such accuracies are not to be expected from a completely automatic document analysis system. First, additional errors may occur if automatic layout analysis and text line extraction are erroneous. Secondly, the assessment was performed with a closed vocabulary. In a real-world scenario any unknown word will lead to additional out-of-vocabulary errors.

The word error rates on the IAM-HistDB are far below the benchmark results for unconstrained modern handwriting recognition. For the IAMDB the best error rate reported in the literature is currently about 26% [65]. The NN-based recognition system used in this thesis is on par with this benchmark according to [65].

The relatively high word error rates for the George Washington database can be explained by the small size of the training set which consists of only 10 pages. Other word error rates reported in the literature are listed in Table 8.4. A direct comparison of the results is not always feasible since we use a smaller number of training samples than most other reports. Yet the general difficulty of the data set becomes evident.

The text line experiments can be regarded as a successful demonstration how well both HMM and NN can cope with word segmentation in addition to recognition. Especially for noisy historical document images, it is desirable to process complete text lines since word segmentation prior to recognition is prone to errors. In the context of text line recognition it is also demonstrated that statistical language models can significantly improve single manuscript recognition even if no external text corpora are used.

A direct comparison between HMM and NN clearly demonstrates the superior performance of NN for automatic transcription of the IAM-HistDB.

8.2 Feature Extensions

In this section, we present two extensions for the sequential data representation. First, we provide experimental evaluation results for linear and non-linear feature selection methods in Section 8.2.1. Secondly, we report results for the proposed graph similarity features in Section 8.2.2.

8.2.1 Feature Selection

In general, it cannot be predicted prior to recognition which data representation performs best for a given handwritten document. However, feature selection methods have the potential to improve a chosen descriptor in general. For handwriting recognition, the sequential nature constrains the set of applicable methods to unsupervised approaches since no class labels are available for individual feature vectors. In the following, we present experimental results for the three methods discussed in Section 3.4, namely principal component analysis (PCA), independent component analysis (ICA), and kernel PCA (KPCA).

Setup Feature selection is applied to the nine well-known geometric features from Section 3.3.2. Besides the historical PARDB, we also consider the IAMDB for evaluation since the feature set has been mainly used for modern documents in the past. HMM-based single word recognition is performed with about 12000 words from each database. Half of the words, i.e. each second word, is used for training and a quarter of the words for validation and testing, respectively.

The number of HMM states was adopted from the text recognition experiments in Section 8.1. Up to 30 Gaussian mixtures are validated for each feature selection method. For feature selection, the optimal dimensionality of the new feature vectors is also found with respect to the validation accuracy, ranging from 1 to 9 for PCA as well as for ICA, and up to 19 for KPCA. The RBF kernel parameter γ is validated over a logarithmic range from 10^{-2} up to over 10^{3}.

While for PCA and ICA the whole training set is used to find the linear feature space transform, the training of the KPCA is constrained to the feature vector sequences of each 100th word due to computational reasons, similarly to [229]. Still, several thousand feature vectors are considered in order to find the principal components of the feature space.

Results The results of the experimental evaluation on the test set are listed in Table 8.5. For both datasets the word accuracy is reported for the reference system using the nine geometric features without feature selection as well as for the three feature selection methods PCA, ICA, and KPCA. The optimal KPCA parameter values, optimized with respect to the validation

Table 8.5: Feature selection results. Improvements marked with a star (*) are statistically significant over the value in the preceding column.

Dataset	Ref.	PCA	ICA	KPCA	Dim.	γ	G
IAM	73.91	75.77*	77.56*	79.65*	13	10^2	28
Parzival	88.69	88.76	89.88*	91.38*	7	10^3	22

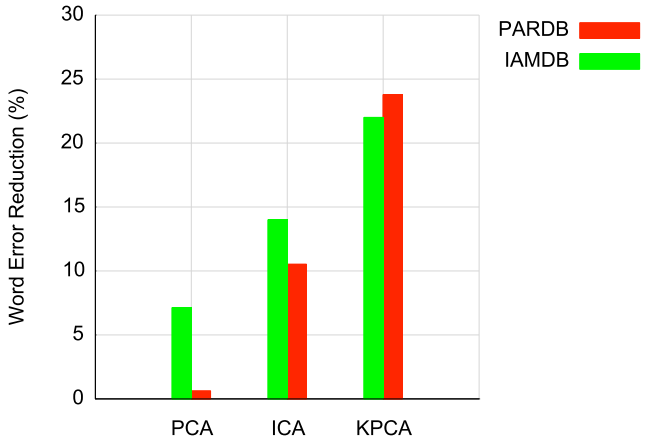


Figure 8.2: Feature Selection

accuracy, are indicated in the three right-most columns, i.e., the dimension of the feature vectors, the γ parameter of the RBF kernel, and the number of Gaussian mixtures (G) of the HMM recognizer.

The improvements of the word accuracy are all statistically significant (t-test with $\alpha = 0.05$) except for PCA when compared to the Parzival reference system. In particular, ICA achieved statistically better results than PCA and KPCA achieved statistically better results than ICA. Figure 8.2 illustrates the word error rate reduction of the feature selection methods when compared to the reference system. For both datasets, KPCA reduced the word error rate by over 20 percent.

Regarding the KPCA parameters, a dimension increase from 9 to 13 dimensions was optimal or the IAM dataset. The γ parameter had a high impact on the word accuracy and had to be validated carefully over a rather large logarithmic scale. The optimal number of Gaussian mixtures was about twice as much for KPCA when compared to the reference system, PCA, and ICA.

Summary The experimental results show that feature selection has great potential to improve off-line handwriting recognition. In accordance with previous work, the word accuracy of an HMM recognizer could be improved with statistical significance using PCA and ICA. With the independent ICA features, significantly better results were achieved than with the linearly uncorrelated PCA features.

As a main contribution, we show that KPCA is able to further improve the word accuracy significantly. By using KPCA with an RBF kernel, the linear PCA transform in the feature space corresponds to an efficient nonlinear transform of the original feature vectors, thus overcoming the limitation of linearity of the standard feature selection methods.

8.2.2 Graph Similarity Features

The graph similarity features (GSF) presented in Chapter 4 integrate structural handwriting information in real-valued features by means of vector space embedding. Using a sliding window with a dynamic width, a structural match with several character prototypes is performed at each position in the text line. This procedure is expected to better preserve the two-dimensional nature of handwriting in the data representation than classical statistical features. In the following, we present an experimental evaluation that compares the proposed GSF descriptor with two statistical descriptors. Besides the nine geometric features from Section 3.3.2, which employ a window of one pixel width, we also consider the zoning features from Section 3.3.3 as a further reference descriptor with a wider window size.

Setup For experimental evaluation, HMM-based single word recognition is performed with about 12000 words from the PARDB. Half of the words are used for training and a quarter of the words for validation and testing.

Besides 79 manually extracted character prototypes, automatic selection strategies (see Section 4.4.2) with respect to HMM-based forced alignment are taken into account as well. For the median and center selection, one prototype per character class in the training set is chosen, resulting in 76 prototypes altogether. For the spanning and k-centers selection, up to five prototypes are chosen per character and the optimal number is determined with respect to the word accuracy obtained on the validation set. For each number of prototypes k, only those characters were taken into account that occur at least k times in the training set. This results in up to 280 prototypes for $k = 5$, which corresponds to the dimension $n = 280$ of the graph similarity features.

For HMM-based forced alignment, only one Gaussian mixture component is used which performs well in our experience [115]. The number of Gaussian mixtures for single word recognition is optimized over a range of

Table 8.6: GSF reference systems.

Descriptor	Acc.	Parameters
Marti & Bunke	88.69	G=7
Vinciarelli	90.49	G=5
Graph Similarity	94.00	D=3.0,C=3.0,G=29

$G \in \{1, 5, 10, 15, 20, 25, 30\}$ on the validation set. The optimal number of HMM states is adopted from the text recognition experiments in Section 8.1.

Graph parameters that are optimized with respect to the validation accuracy include the connection point distance $D \in \{3.0, 5.0, 7.0, 9.0\}$ and the insertion and deletion cost $C \in \{0.4D, 0.6D, \ldots, 1.4D\}$ for graph similarity feature extraction. Graphs are validated with and without edges.

Results For manual prototype selection, the word recognition accuracy on the test set and the optimal parameter values are given in Table 8.6 for the proposed graph similarity features and both reference descriptors. The reference systems are significantly outperformed on the PARDB.

Astonishingly, the use of edges in the graph was not beneficial with respect to the accuracy on the validation set. For dense skeleton coverage with graph nodes, edges are somewhat redundant. In addition, edges can have a negative effect for broken characters in noisy images of historical documents because they penalize broken connections in the handwriting skeleton with edge deletion costs.

The parameters D and C of the graph similarity features had a significant impact on the validation accuracy. The optimal choice was given by the most dense connection point distance of $D = 3.0$ and the same node insertion and deletion cost $C = 3.0$. We interpret the high optimal number of Gaussian mixtures $G = 29$ found for graph similarity features as an indicator for a good overall feature quality, because the HMM could be closely adapted to the training set without suffering from overfitting.

The feature normalization discussed in Section 4.5.2 has proven to be very important. A test accuracy of only 84.19 is reported for the plain graph edit distance features. For the unnormalized similarity features, a test accuracy of 90.60 is reported. Both results are significantly below the result achieved with the normalized graph similarity features.

For automatic prototype selection with the same system parameters like manual selection, the word accuracy on the test set is given in Table 8.7. The manual selection is outperformed by all automatic selection strategies. In case of the k-centers selection, the improvement is statistically significant (t-test, $\alpha = 0.05$).

In Figure 8.3, the selected prototypes are shown for the different selection

Table 8.7: GSF prototype selection. Word accuracy on the test set with optimal number of prototypes per character k and feature dimension n.

Selection	Accuracy	Parameters
Manual	94.00	n=79
Median	94.07	n=76
Center	94.31	n=76
Spanning	94.14	k=3, n=195
k-Centers	94.51	k=4, n=244

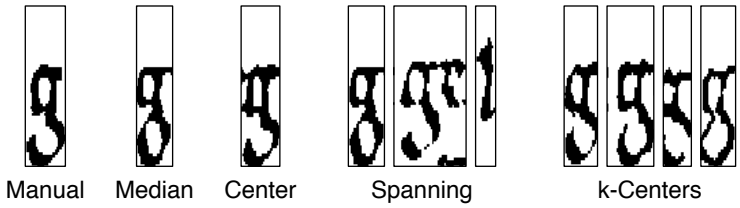


Figure 8.3: Selected Prototypes

strategies, exemplarily for the character "g". Similar results are obtained for the other characters. The spanning selection results are ordered by iteration and the k-centers results by cluster size in descending order. For the median and center selection, the result is astonishingly close to the human choice and it makes sense that nearly the same recognition accuracy is achieved. Including more character prototypes by spanning selection results in the selection of outliers after the first iteration, stemming from wrong character segmentations. The visual inspection confirms the k-centers selection as the most promising prototype selection strategy, since all selected cluster centers are relatively clean character images with different appearances.

Summary The proposed graph similarity features have shown a very promising performance for HMM-based single word recognition. A word error rate of 5.5% is reported for a subset of the PARDB which is the highest result we have achieved on this database so far. The GSF descriptor has significantly outperformed two statistical reference descriptors. Using HMM-based forced alignment, automatic prototype selection was able to outperform manual selection.

The optimal parametrization of the GSF features on the PARDB was to employ a dense skeleton coverage, to use no edges in the handwriting graph, and to select a rather large amount of prototype characters with k-Centers.

8.3 NN Extension

In this section, we present an experimental evaluation of the algorithmic extension for NN-based text line recognition presented Section 6.3. A fast approximation of the optimal token passing algorithm used in [96] is proposed, which aims at reducing the computational effort for text line recognition with bigram language models and provides a more versatile recognition output than the baseline algorithm in form of word lattices. The algorithm is introduced in this thesis and has not been published elsewhere yet.

The experimental setup is described in Section 8.3.1, results are reported in Section 8.3.2, and a discussion is provided in Section 8.3.3.

8.3.1 Setup

We consider the experimental setup for text line recognition with word bigrams from Section 8.1. In addition to the three historical databases, i.e., the SGDB, PARDB, and GWDB, we also include the modern IAMDB (see Section 2.4) in order to investigate larger language models.

For the IAMDB, 6161 text lines are used for training, 920 for validation, and 929 for testing. The language model is estimated on the Lancaster-Oslo/Bergen corpus (LOB) [120], which contains a variety of printed British English texts. About a million words are included in the corpus. Because the IAMDB was created based on these texts, we have removed the text lines of the test set from the corpus. The language model uses Kneser-Ney smoothing, is limited to a vocabulary of the 20000 most frequent words of the corpus, and has a perplexity of 427.1 on the test set. The out-of-vocabulary rate is 8.2%, i.e., the maximum accuracy for recognition is 91.8%.

The setup of the NN recognizer corresponds with Section 8.1.3. Ten networks are randomly initialized and the best net is selected based on its accuracy on the validation set. The proposed token passing algorithm and the optimal baseline algorithm are applied to the same network output. For pruning, we have used a maximum number of $n_\tau = 5$ up to 30000 tokens. The other parameters of the algorithm are set to $n_\omega = 10$ word ends recorded in the lattice, grammar scale factor $\alpha = 1$, and word insertion penalty $\beta = 0$.

The threshold for ϵ-compression is set to $\theta = 99.9\%$. The resulting compression factor is listed in Table 8.8 for all data sets. It ranges between 5.7 for the PARDB and 9.7 for the IAMDB. The resulting number of network outputs per character is similar for all data sets.

For measuring the CPU time, we have tested our Java prototype of the algorithm against a C++ implementation of the baseline algorithm [96] kindly provided by Dr. Alex Graves. For the historical databases, we have also recorded the CPU time for HMM-based 1-best Viterbi recognition with HTK[3]. Experiments are performed on AMD Opteron 2354 2.2GHz nodes.

[3] `http://htk.eng.cam.ac.uk/`

Table 8.8: Epsilon compression for $\theta = 99.9\%$. Compression factor C is listed alongside with the number of feature vectors per character F_u, F_c for uncompressed and compressed sequences, respectively.

Database	C	F_u	F_c
SGDB	7.4	30.9	4.2
PARDB	5.7	22.0	3.9
GWDB	6.4	31.8	4.9
IAMDB	9.7	42.7	4.4

Table 8.9: Token passing speedup for 10000 tokens. Number of lexicon words, CPU time per character in milliseconds T_{ref} (optimal algorithm), T_{new} (proposed approximation), and speedup factor.

Database	Lexicon	T_{ref}	T_{new}	Speedup
SGDB	5762	815.3	86.3	9.4
PARDB	4934	554.9	96.3	5.8
GWDB	1471	152.6	103.7	1.5
IAMDB	20000	24083.5	116.8	206.2

8.3.2 Results

The behavior of the algorithm with respect to its accuracy and CPU time are shown in Figures 8.4 and 8.5 at the end of this chapter. In Figure 8.5, an $x \log(x)$ reference is plotted with $x = \frac{n_\tau}{300}$. The maximum accuracy is rapidly approximated as the number of tokens increases. The NN baseline algorithm was faster than HMM with HTK. For the IAMDB, NN-based recognition with the LOB language model achieves an error rate of 25.4% which confirms the findings in [65] for this NN architecture and is among the best results achieved on the IAMDB so far.

In Table 8.9, the speedup of the approximation with respect to the baseline algorithm is listed for $n_\tau = 10000$ tokens. For this number of tokens, the accuracy of the proposed algorithm equals the accuracy of the baseline for three out of four data sets, i.e., it is not statistically significantly worse (t-test, $\alpha = 0.05$). For Saint Gall database, 20000 tokens would be needed resulting in a speedup factor of 8.5. For the IAMDB, 5000 tokens would already be sufficient, leading to a speedup factor of 667.2. This is the highest speedup factor we can report without significantly decreasing the accuracy.

The CPU time of the baseline algorithm increases for larger lexicons as expected since its computational complexity is $O(N^2T)$ with respect to the

lexicon size N and the observation length T. In contrast, the computational complexity of the proposed algorithm is $O(M \log(M)T)$ with respect to the maximum number of tokens M. It is surprising that the amount of tokens needed for achieving an optimal accuracy is rather constant for the four databases despite their differences in lexicon size, i.e., recognition is independent of the lexicon size in the considered scenario. Hence, the speedup is drastically increased for large lexicons ranging form 1.5 for the GWDB (lexicon size 1471) to 206.2 for the IAMDB (lexicon size 20000).

8.3.3 Discussion

With the proposed approximation of the optimal text line recognition algorithm we were able to reduce the CPU time drastically without loosing accuracy. The proposed algorithm has shown a surprising independence of the lexicon size in the considered scenario. Speedup factors range between 1.5 (GWDB, 10000 tokens, lexicon size 1471) and 667.2 (IAMDB, 5000 tokens, lexicon size 20000).

Besides the goal of an optimal accuracy, it would also be interesting to focus on computational speed. Indeed, Figure 8.5 indicates that reasonable accuracies can be achieved with only very few tokens. This has great potential for processing large manuscript collections in the context of digital libraries for historical documents.

The success of rigorous token pruning is based on the compact search space we have experienced for off-line handwriting recognition with BLSTM networks in combination with CTC. First, the observation sequence can be compressed since the "no character" ϵ activation is close to 1 most of the time. After compression, only about 4 observations per character have to be processed for the databases considered. Secondly, the CTC text model requires only one state per character which leads to very compact prefix graphs for token passing.

Apart from the speedup capabilities the proposed algorithm also provides a more versatile recognition output in form of word lattices. So far, we have only considered the best path through the lattices. Future work includes an assessment of the general lattice quality, for instance in the context of classifier combination.

8.4 Summary and Outlook

In this chapter, we have investigated the performance of automatic transcription on the IAM-HistDB. The neat and clean handwriting style in the historical documents leads to high recognition accuracies compared with unconstrained modern handwriting, despite difficult image conditions and relatively small training sets. Especially the 4% error rate for the Carolingian script of the Saint Gall database and the 6.7% for the Gothic script of

the Parzival database are promising for content-based indexing of historical manuscripts in digital libraries. With only 10 training pages, we report an error rate of 18.1% for the George Washington database.

For real-world applications, additional errors have to be expected if automatic layout analysis and text line extraction are erroneous. Also, additional out-of-vocabulary errors have to be taken into account if the available lexicon does not cover all words that occur in the manuscript.

It was demonstrated that both HMM and NN can cope with complete text line recognition. Especially for noisy historical document images, it is desirable to process complete text lines since word segmentation prior to recognition is prone to errors. Using statistical language models compiled from the training set, i.e., without external resources, text line recognition results were on par with single word recognition and uniform language model. In fact, the best results reported for the Parzival database and the George Washington database were achieved with text line recognition.

The best results were obtained with NN-based recognition. The discriminative approach constantly and significantly outperformed HMM-based recognition when using a set of nine geometric features for handwriting representation.

The geometric features could be significantly enhanced with feature selection. When using non-linear kernel PCA, an error reduction of over 20% could be achieved for HMM-based recognition on subsets of the Parzival database as well as the IAMDB for modern handwriting.

With the proposed graph similarity features, the word error could be reduced about 50% for HMM-based recognition on a subset of the Parzival database, when compared with the geometric features. The error rate of 5.5% is the best result we have achieved on this data set so far. The optimal parametrization of the graph similarity features was to employ a dense skeleton coverage, to use no edges in the handwriting graph, and to select a rather large amount of prototype characters with k-centers.

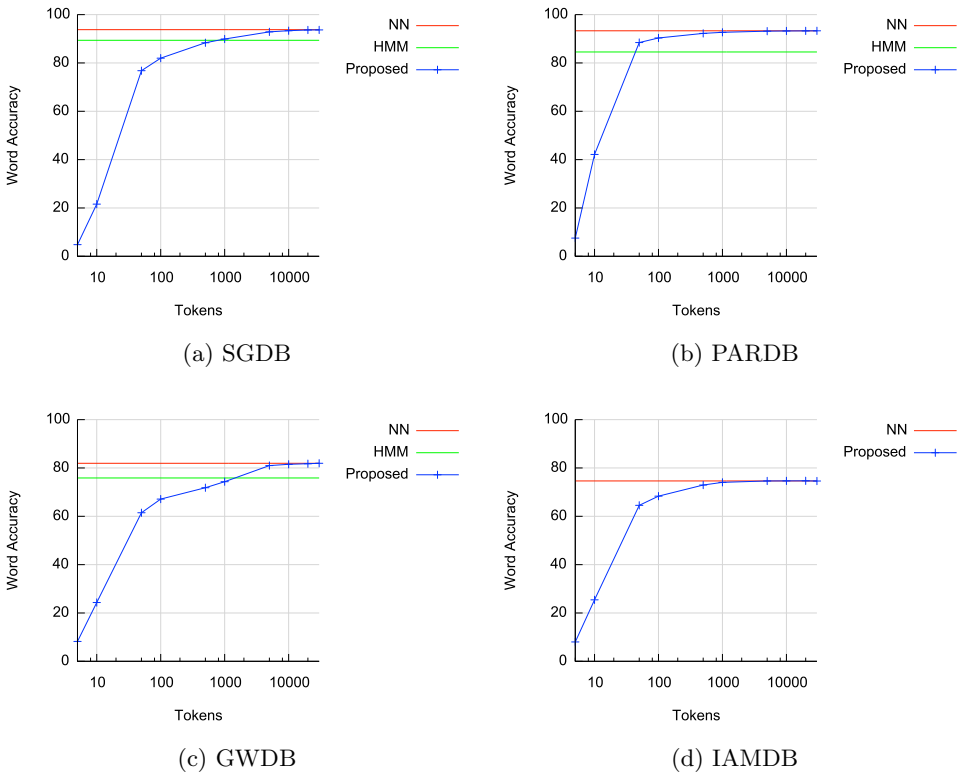
For NN-based text line recognition with bigram language models, we have tested a token passing approximation which is newly introduced in this thesis. When compared with the optimal baseline algorithm, the CPU time could be drastically reduced without loosing accuracy. The proposed algorithm has shown a surprising independence of the lexicon size in the considered scenario. For a 20000 word lexicon, the proposed algorithm was over 600 times faster than the baseline on the IAMDB. Only 5000 tokens were sufficient to achieve an optimal accuracy in the compact search space of BLSTM networks with CTC.

There is a series of open issues that are not addressed in this thesis. First, a combined evaluation of text extraction and handwriting recognition is needed in order to assess the accuracy of a complete document analysis system. Secondly, external language corpora have not been investigated so far. A good starting point for the Saint Gall database would be, for instance,

the electronic Patrologia Latina text edition (see Section 2.1). Finally, the use of training samples and recognition systems for a large collection of similar manuscripts would be an ideal assessment. This is conditioned by the availability of a sufficient amount of annotated manuscripts.

Two lines of research are especially promising for future work. First, the investigation of graph similarity features is still at the beginning stage. We have only tested one graph model, one dissimilarity function, and one data set. The framework is very general and more studies are clearly needed to assess the proposed prototype-based approach to handwriting features.

Secondly, the token passing approximation has shown an impressive performance and the need for further studies is evident. It seems that the compactness of the search space for BLSTM and CTC allows for high-speed text recognition. In many scenarios, it would be of great interest to use suboptimal but fast algorithms to access the textual content of large document collections, most notably for indexing large historical archives and for semi-supervised learning.



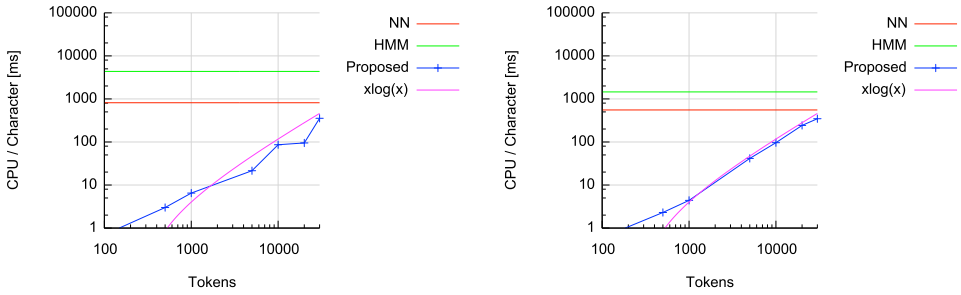
(a) SGDB

(b) PARDB

(c) GWDB

(d) IAMDB

Figure 8.4: Token Passing Accuracy



(a) SGDB

(b) PARDB

(c) GWDB

(d) IAMDB

Figure 8.5: Token Passing Speed

Chapter 9

Keyword Spotting

The automatic transcription of historical documents discussed in Chapter 8 is conditioned on the availability of two sources of information. First, training samples are needed to derive appearance models for characters and words. Secondly, electronic text corpora are needed to derive word lexicons and language models. However, both sources of information are difficult to access for historical documents as mentioned in Section 1.4. The extraction of training samples typically includes time-consuming and expensive manual interaction and large text corpora with a consistent orthography are often unavailable for historical documents.

If a complete transcription is not feasible, an important intermediate goal is the identification of individual search terms within manuscript images. Also known as keyword spotting, this would allow for content-based indexing of historical archives [184]. An index based on a limited number of keywords could establish a first access to millions of scanned manuscript images. Without the computational overhead of a large lexicon and language model, keyword spotting algorithms are also expected to perform faster than automatic transcription, which is desirable for large archives. Originally proposed by Manmatha et al. in [149] for indexing historical documents, keyword spotting has also been investigated in the context of modern documents. One of the major applications is given by automatic mail sorting, where a fast triage of handwritten correspondence is desired based on keywords like "urgent" [190].

Two different approaches to handwritten word spotting can be distinguished in the literature. On the one hand, *template-based* methods directly match keyword template images with manuscript images. Whenever a close match is achieved, manuscript images are returned as a result. An advantage of this approach is that template images can be obtained with relatively little human effort, even if the underlying language and its alphabet are unknown. However, such systems are limited by the fact that for each keyword at least one template image is needed and unknown out-of-vocabulary words cannot

be spotted at all. Also, template images typically have low generalization capabilities to new handwriting styles.

On the other hand, *learning-based* methods employ statistical learning to train keyword models. At word level, models can directly be trained from word template images. Such word models are expected to show a better generalization capability to new handwriting styles than raw template images. Also, the computational effort is reduced to a single match against one keyword model instead of a match against all template images of a keyword. However, word models still need a considerable amount of templates for training and are not able to spot out-of-vocabulary keywords. When the learning-based approach is applied at character level, a word spotting system obtains, in principle, the capability to spot arbitrary keywords by concatenating the character models appropriately. Although this approach is well-known for the recognition of speech [193] and poorly printed documents [42], only few reports can be found in the handwriting recognition literature.

In this thesis, we present two novel keyword spotting systems that perform machine learning at character level, one using HMM and the other using NN. Compared with previous word spotting systems, the proposed sub-word model approach has several advantages. First, the systems are segmentation-free at the training stage as well as at the recognition stage, i.e., they are not dependent on a segmentation of text lines into words which can be prone to errors. An advantage over learning-based systems at word level is given by the fact that only a small number of character classes needs to be trained for which a rather large number of training samples is available in a given handwritten text. Furthermore, the proposed systems are able to spot arbitrary keywords that are not required to be known at the training stage. Compared with transcription-based systems, the proposed word spotting approach has the advantage that no lexicon is needed. First, such a lexicon might not be available for historical languages and secondly, lexicon-based recognition imposes a high computational effort for large vocabularies underlying natural language.

A known disadvantage of the proposed approach is, however, that it requires transcribed text line images for training which implies considerable human effort for historical documents. Furthermore, template-based image matching might be the only option available if neither language nor alphabet are known for a historical document. Finally, lexicon-based transcription is expected to outperform character-based spotting in terms of accuracy if reliable lexicons and language models are available.

In an experimental evaluation, we have tested the proposed systems on the George Washington database and the Parzival database (see Chapter 2). Since the keyword spotting systems are not specific to historical documents, we also evaluate them on the IAM database for the sake of comparison with modern documents. The systems are compared with a well-established

template matching method based on Dynamic Time Warping (DTW) [183]. It is demonstrated that the proposed sub-word model approach outperforms the reference system on all data sets.

The remainder of this chapter is organized as follows. First, related work is discussed in Section 9.1. Next, the proposed keyword spotting systems are introduced in Section 9.2. HMM-based spotting is presented in Section 9.2.1 and NN-based spotting in Section 9.2.2. The DTW reference system is discussed in Section 9.2.3. Afterwards, the conducted experiments are elaborated in Section 9.3. Finally, a summary and an outlook to future work is provided in Section 9.4.

9.1 Related Work

Keyword spotting has been applied to speech [161, 193] and poorly printed documents [42, 132] before. For handwritten text, it was proposed a few years later in [149]. In the following, several template-based and learning-based methods for handwritten keyword spotting are discussed.

9.1.1 Template-Based Spotting

In [149], a template-based approach was pursued that used the Scott and Longuet-Higgins distance (SLH) [213] to compare keyword template images and unknown word images. In the following, other features based on global image characteristics have been proposed, e.g., gradient, structural and convexity features (GSC) [257], and features based on moments of binary images [19].

A different template-based approach is given by using local features in conjunction with elastic matching. Notable work in this domain includes the corner features proposed in [195], sliding window features in combination with dynamic time warping (DTW) [182], and gradient angle features in combination with a cohesive elastic distance [139]. DTW, in particular, is well-established in the field and has been used in combination with different features in recent work, e.g., based on word profiles [183], closed contours [3], and local gradients [189, 231]. In [231], an extension of DTW is presented that allows to match keyword templates with complete text lines rather than segmented word images.

9.1.2 Learning-Based Spotting

A learning-based approach at word level was recently presented in [190]. Based on local gradient features, posterior probabilities of keyword HMM are used for keyword spotting in conjunction with universal vocabularies for score normalization. A similar approach was presented in [45] for non-

Figure 9.1: Keyword Spotting

symmetric half plane HMM. Instead of using posterior probabilities, a scoring method based on Fisher kernels has been proposed in [172].

For spotting arbitrary keywords, the learning-based approach has been investigated at character level as well. Character template images were used in [39, 61] to train generalized HMM for keyword spotting. While promising results were reported for historical Latin manuscripts [61] and Arabic scripts [39], an automatic acquisition of such character templates from handwritten text images is difficult in general. The same limitation is faced in [35], where segmented character images and Gabor features were used in a template-based method for keyword spotting.

In an earlier work [64], character HMM were used to spot street names for the constrained task of address reading. In a segmentation-free approach, character models are trained on complete text line images and are then connected to model keywords as well as general non-keyword text. Recently, this line of research was followed in [232] for unconstrained handwritten word spotting.

In contrast to [64, 232], the methods proposed in this thesis are not dependent on a lexicon. Instead of a comparison with other lexicon words, character-based confidence models are used to spot keywords. For HMM-based spotting, a log-odds score with respect to a general filler model is employed as a confidence value. This technique is well-known from other application domains of HMM, e.g., speech recognition [193] and bioinformatics [11], and has been used similarly in [190] at word level. For NN-based spotting, the character posteriors provided by the networks are used directly as confidence.

9.2 Systems

The keyword spotting task considered in this thesis is the identification of keywords in text line images. The aim is to return all text lines that contain a given search term. An example is shown in Figure 9.1 for retrieving text line images of the George Washington database (top) and Parzival database (bottom) that contain the keywords “Orders” and “vater”, respectively. Whenever one or several text lines include a search term, the corresponding manuscript can be indexed with this keyword in digital li-

braries. In addition, the retrieved location of the keyword, i.e., its start and end position within the line, can be used for browsing the manuscript.

There are two preconditions for the proposed keyword spotting systems. First, a set of training samples need to be available for machine learning. They consist of text line images together with their correct transcription. Secondly, the manuscript pages need to be segmented into text lines for spotting. No segmentation of text line images into words is required, neither for training nor for spotting. In this thesis, we consider perfect text line extraction in order to focus on handwriting recognition. In real-world applications, text line extraction errors would impose additional spotting errors.

Two learning-based spotting systems are proposed that operate at character level, one using HMM and the other using NN. For learning character models, the same procedure is applied as for automatic transcription. That is, text line images are normalized according to Section 3.2.2 and features are extracted with a sliding window according to Section 3.3. Afterwards, character HMM are trained with the Baum-Welch algorithm according to Section 5.2.2. The character models approximate the feature distribution with several Gaussian mixtures for each state of a linear topology. The BLSTM NN are trained with gradient descent according to Chapter 6, such that they are able to provide character posterior probabilities $p(c_i|x_j)$ for each character c_i of the alphabet and each feature vector x_j of an arbitrary text line. The proposed keyword spotting systems employ these character models to derive a score $s(\mathbf{x}, w) \in \mathbb{R}$ for matching the text line image $\mathbf{x}$, given as a sequence of feature vectors, with a keyword w, given as a string. The score is compared with a threshold T

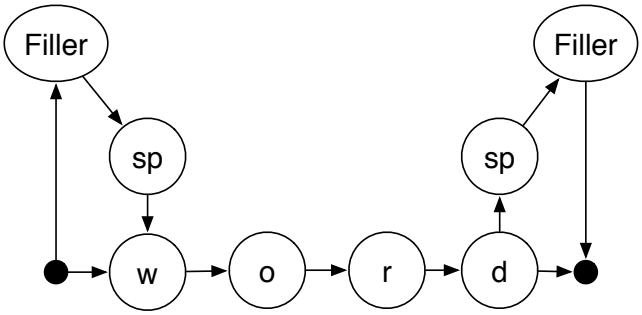
$$s(\mathbf{x}, w) \geq T \tag{9.1}$$

and all text line images, whose score exceeds the threshold, are returned as the spotting result. Basically, the text line images are ranked with respect to their similarity to the keyword, i.e., their matching score, and the flexible threshold determines the similarity needed for spotting. The keyword is given by an arbitrary sequence of known characters, similar to search terms entered in Internet search engines.

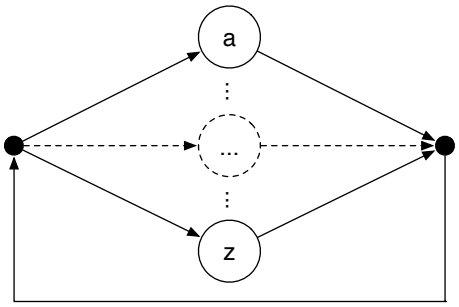
In the following, the proposed HMM-based spotting system is presented in Section 9.2.1 and the NN-based system in Section 9.2.2. Afterwards, the reference system using DTW is discussed in Section 9.2.3.

9.2.1 HMM-Based Spotting

The proposed score $s(\mathbf{x}, w)$ of the text line image $\mathbf{x}$ for the keyword w is based on the posterior probability $p(w|\mathbf{x}_{a,b})$ where a and b are the most likely start and end position of the keyword and $\mathbf{x}_{a,b}$ is the corresponding



(a) Keyword HMM



(b) Filler HMM

Figure 9.2: HMM-Based Spotting

part of the text line image. Applying Bayes' rule to logarithmic values, we obtain

$$\log p(w|\mathbf{x}_{a,b}) = \log p(\mathbf{x}_{a,b}|w) + \log p(w) - \log p(\mathbf{x}_{a,b}) \tag{9.2}$$

The prior $p(w)$ can be integrated into a keyword specific threshold that is optimized at training stage. For arbitrary keywords that are not known at the training stage, we assume equal priors. In both cases, we only take the terms

$$\log p(\mathbf{x}_{a,b}|w) - \log p(\mathbf{x}_{a,b})$$

into account for calculating the text line score. This log-likelihood difference is obtained by two text line models.

The first model used for the term $\log p(\mathbf{x}_{a,b}|w)$ is the *keyword text line model* K shown in Figure 9.2a for the keyword "word". It is constrained to contain the exact sequence of characters of the keyword at the beginning, in the middle, or at the end of the text line, separated by the space character "sp". The rest of the text line is an arbitrary sequence of characters that is modeled with the *filler text line model* F shown in Figure 9.2b. From Viterbi recognition (see Section 5.2.2) with keyword model K we obtain the most likely start position a and end position b of the keyword, alongside

with the log-likelihood $\log p(\mathbf{x}_{a,b}|w) = \log p(\mathbf{x}_{a,b}|K)$.

The second model used for the term $\log p(\mathbf{x}_{a,b})$ is the unconstrained filler model F. The obtained log-likelihood $\log p(\mathbf{x}_{a,b}) = \log p(\mathbf{x}_{a,b}|F)$ indicates the general conformance of the text image to the trained character models. A known writing style that was used for training has a higher likelihood than an unknown writing style. By subtracting the term $\log p(\mathbf{x}_{a,b})$ from the keyword log-likelihood $\log p(\mathbf{x}_{a,b}|w)$ the score is normalized with respect to the writing style and allows a better generalization to unseen test images. Also known as log-odds scoring [11], this technique has been used similarly in [190] with a single GMM filler.

The filler-based score normalization is discussed in the general context of confidence modeling in Section 5.4. Note that the likelihood $p(\mathbf{x}_{a,b}|w)$ is not a valid confidence measure and, indeed, leads to a near-random spotting performance in our experience. The challenge for HMM-based keyword spotting is to find robust confidence models. The posterior is a natural measure of confidence and is approximated with a filler model in the proposed system[1].

For the final text line score we can ignore the start and end position of the keyword, because the log-likelihood difference between the keyword model K and the filler model F is zero outside the keyword position, not taking into account the small deviations at the keyword borders. That is, the log-likelihood difference $\log p(\mathbf{x}_{a,b}|w) - \log p(\mathbf{x}_{a,b})$ is given by the log-likelihood difference

$$\log p(\mathbf{x}|K) - \log p(\mathbf{x}|F)$$

of the text line models K and F over the complete text line image $\mathbf{x}$. In a final step, the log-likelihood score is normalized with respect to the keyword length $L_K = b - a$ and compared to a threshold T for word spotting.

$$s(\mathbf{x}, w) = \frac{\log p(\mathbf{x}|K) - \log p(\mathbf{x}|F)}{L_K} \geq T \tag{9.3}$$

In effect, we obtain an estimated posterior probability per state, which is comparable across different keywords.

Note that the maximum text line score $s(\mathbf{x}, w)$ is zero. This maximum score is achieved if the exact keyword character sequence is, in fact, the most likely character sequence in the filler model. The optimal value of T can be determined in the training phase with respect to the user needs as a trade-off between system precision and recall. Precision and recall are discussed in Section 9.3.2 and are calculated for all possible thresholds for system evaluation.

[1] In fact, the filler model can also be interpreted as an alternative hypothesis, i.e., "no keyword is present in the text line". The very same log-odds score reflects a hypothesis test in this context as pointed out in Section 5.4.2.

In case of multiple occurrences of a keyword within a text line, the most likely position is taken into account in this thesis for scoring the text line image and it can be returned if needed. In order to return all keyword positions, a token passing implementation of the Viterbi algorithm could be considered. Whenever a token arrives at the sink, i.e., the non-emitting state at the bottom-right of Figure 9.2a, a possible word end is found. By storing the word start in the token, several possible keyword positions within the text line are be obtained, which can be sorted by their score $s(\mathbf{x}, w)$ to retrieve N-best keyword positions[2].

Computational Complexity In the proposed lexicon-free approach to word spotting, the time complexity to score a text line image is $O(A^2N)$ for Viterbi recognition with the keyword and filler text line models where A indicates the number of characters in the alphabet and N the length of the text line. This is a great advantage in terms of computational speed compared to transcription-based approaches using a lexicon. Here, the time complexity is $O(L^2N)$ for a lexicon of size L. For transcription-based systems that deal with texts written in natural language, typical values for L are several ten thousands, whereas the number A of characters in an alphabet is usually below one hundred.

9.2.2 NN-Based Spotting

For HMM-based keyword spotting, the challenge to derive a reliable score $s(\mathbf{x}, w)$ was to approximate the posterior probability of the keyword in a computationally feasible manner with respect to the Gaussian mixture densities of the character HMM. In case of the discriminative NN character models, we obtain the posterior probability of the keyword w directly as

$$\log p(w|\mathbf{x}_{a,b}) = \sum_{i=a}^{b} \log p(c_{l_i}|x_i) \tag{9.4}$$

at its most likely position $\mathbf{x}_{a,b}$ with respect to the optimal sequence of characters $c_{l_a}, \ldots, c_{l_b}$ and logarithmic values. This posterior is independent of a lexicon in this case and can be calculated directly form the character posteriors $p(c_i|x_j)$ given for all characters $c_1, \ldots, c_A$ of the alphabet, including the ϵ character, and for all feature vectors of the text line $\mathbf{x} = x_1, \ldots, x_N$. That is, the posterior can be obtained directly form the character models $p(c_i|x_j)$ of the neural network (see Chapter 6).

The most likely position $\mathbf{x}_{a,b}$, sequence of characters $c_{l_a}, \ldots, c_{l_b}$, and posterior probability $p(w|\mathbf{x}_{a,b})$ is obtained with the token passing algorithm for NN as shown in Figure 9.3 for the keyword "wood". The token passing graph

[2] In this scenario, post-processing would be needed to ensure that the keywords are non-overlapping.

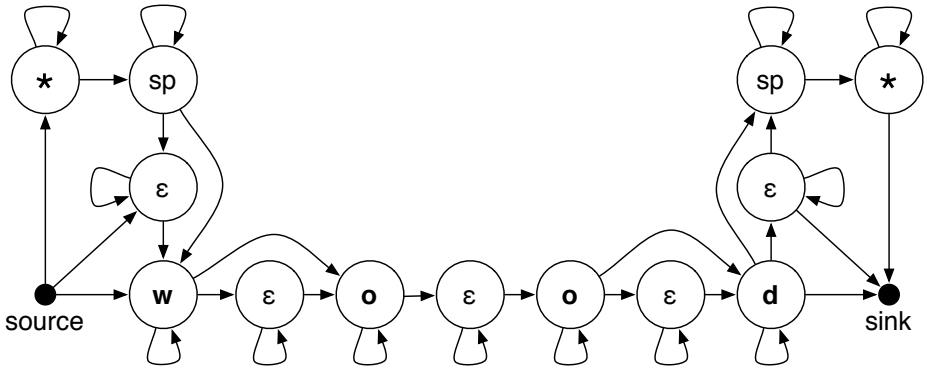


Figure 9.3: NN-Based Spotting

for NN is constructed very similar to the HMM graph in Figure 9.2a. Disregarding the "no character" ϵ-nodes, the same setup is given. The keyword is requested to be at the beginning, at the end, or in the middle of the text line separated by the space character "sp". Outside the keyword, the text line is modeled with a generic filler given by the "$*$" node. Unlike HMM, posteriors are taken into account and we set $p(*|x_j) = 1$ for all feature vectors x_j. Hence, the text line outside the keyword does not contribute to the resulting posterior and we can directly obtain $p(w|\mathbf{x}_{a,b})$ from Equation 9.4. That is, no score normalization with respect to a filler model is needed.

While the internal HMM states are hidden in Figure 9.2a for better visibility, we display the complete token passing graph for NN in Figure 9.3 in order to show the special treatment of the ϵ character. Between each character peak in the output of the neural network, an arbitrary number of ϵ characters are allowed. This "no character" output node of the NN is active with a probability near 1 for the large part of the text line and is only rarely interrupted by a character peak. In case of two consecutive peaks of two different characters, e.g., $w \rightarrow o$ in the figure, a direct transition is allowed. In order to distinguish consecutive peaks of the same character from character transitions, at least one ϵ peak is requested between two identical characters, e.g., $o \rightarrow \epsilon \rightarrow o$ in the figure.

Token passing starts with inserting a single token with a logarithmic posterior of 0 at the source, which is propagated to the nodes $*$, ϵ, and w. The logarithmic character probabilities $\log p(*|x_1)$, $\log p(\epsilon|x_1)$, and $\log p(w|x_1)$ are added to the value of the three tokens. Afterwards, an iteration over all positions $1 \leq j \leq N$ in the text line is performed. At each iteration, the tokens are propagated along the directed graph edges, are updated with $p(.|x_j)$, and the token with the highest log-probability is kept at each graph node. The token arriving at the sink after $j = N$ iterations carries the de-

sired posterior $\log p(w|\mathbf{x}_{a,b})$. If needed, the optimal location of the keyword can be retrieved if the entrance into the node w is recorded in the token. Also, a ranked list of several keyword positions can be retrieved this way by taking into account tokens that arrive earlier at the sink for $j < N$. Non-overlapping keywords would require additional post-processing. In this thesis, we only take the most probable position of the keyword into account to score text line images.

Similar to HMM, the final score of the text line is given by normalizing the posterior probability

$$s(\mathbf{x}, w) = \frac{p(w|\mathbf{x}_{a,b})}{L_K} \geq T \tag{9.5}$$

with the length of the keyword $L_K = |w|$ given by its number of characters. In effect, we obtain a probability per character, which is comparable across different keywords. That is, the same threshold can be applied to different keywords.

Computational Complexity NN-based keyword spotting is even faster than HMM-based spotting, because no loop over all characters is needed. The time complexity is given by $O(|w|N)$ with respect to the number of characters $|w|$ of the keyword and the width N of the text line. The memory needed is $O(|w|)$[3].

Even without special speed-up measures such as ϵ-compression and node beam search (see Chapter 6), NN-based keyword spotting is the fastest handwriting recognition algorithm presented in the whole thesis. It allows us to extract content-based information from handwriting images in milliseconds per text line on our computers if the network output is considered available. In contrast, we spend minutes per text line for automatic transcription with large lexicons using the same network output and moderately pruned token passing.

9.2.3 Reference System

We compare the proposed word spotting systems with a widely used reference system that is based on Dynamic Time Warping (DTW). DTW-based keyword spotting was proposed in [161] for speech recognition and is also well-established in the field of handwritten word spotting [3, 183, 189, 231].

Since DTW can be applied to the same feature vector sequence used for HMM and NN, a fair comparison between classical template matching and the proposed learning-based approach is possible when using this reference system. While the proposed systems use the feature sequence for training

[3]Token passing reduces the memory requirements to a single column of the dynamic programming matrix illustrated in Figure 5.2.

character models, DTW directly matches the feature sequence of keyword template images and unknown text images. If the resulting distance score is lower than a certain threshold, a positive match is returned.

In order to apply DTW at text line level, a segmentation of the text line into words is needed. Especially for historical documents, an explicit segmentation is a challenging problem [150]. The impact of errors in word image extraction can be reduced by adding an uncertainty margin around candidate words [130]. Another solution is to use continuous dynamic programming to perform a subsequence matching within a complete text line as recently proposed in [231]. In this thesis, we consider a perfect, manually corrected word segmentation that is readily available for the data sets under consideration. Hence, the results reported for DTW in Section 9.3.3 can be seen as an upper bound of the DTW performance with respect to word segmentation.

DTW Distance For calculating the DTW distance between two feature vector sequences $\mathbf{x} = x_1, \ldots, x_N$ and $\mathbf{y} = y_1, \ldots, y_M$, an optimal alignment of the two sequences is determined that takes into account different sequence lengths. By this process, the feature vectors are aligned along a common, warped time axis. The cost of an alignment is given by the sum of distances $d(x, y)$ of each aligned vector pair. The distance measure $d(x, y)$ employed is the squared Euclidean distance

$$d(x, y) = \sum_{i=1}^{n} (\hat{x}_i - \hat{y}_i)^2 \tag{9.6}$$

with respect to the number of features n and normalized features $\hat{x}_i$. For normalization, a linear scaling is applied such that $\hat{x}_i \in [0, 1]$. The DTW distance $DTW(\mathbf{x}, \mathbf{y})$ of the word images $\mathbf{x}$ and $\mathbf{y}$ is then given by the minimum alignment cost that is found by means of dynamic programming [183]. For speeding up the distance calculation, a Sakoe-Chiba band [199] is used. The resulting distance is finally normalized with respect to the length of the optimal alignment, also called warping path. For more details on the DTW distance algorithm, we refer to [183].

Text Line Scoring For word spotting, the score $s(\mathbf{x}, w)$ of the text line image $\mathbf{x}$ for the keyword w is based on the DTW distance between segmented word images. Given the images $\mathbf{x}_1, \ldots, \mathbf{x}_k$ of all words contained in the text line and the available keyword template images $\mathbf{t}_1, \ldots, \mathbf{t}_l$, the text line score is given by the minimum DTW distance

$$s(\mathbf{x}, w) = \min\{DTW(\mathbf{x}_i, \mathbf{t}_j)\} \leq T \tag{9.7}$$

for $1 \leq i \leq k$ and $1 \leq j \leq l$. That is, a text line is returned as a positive match, if the distance of one of its word images to any keyword template image is smaller than a certain threshold T.

Table 9.1: Number of keywords and number of samples per character and keyword in the training set.

Database	Keywords	Character Samples	Keyword Samples
GWDB	105	220.0	2.7
PARDB	882	797.2	7.6
IAMDB	1217	4054.9	7.4

Note that unlike the proposed systems, the DTW-based reference system requires at least one template image for spotting a keyword. For a large amount of template images, the computation time poses another problem. While all training data is incorporated in the character models in the proposed systems and can be used for a single match against a given feature vector sequence, DTW requires matching the sequence with all available keyword templates.

9.3 Experiments

The keyword spotting systems are tested on the GWDB and PARDB. For comparison with modern documents, we also evaluate them on the IAMDB. The results achieved with the proposed HMM-based and NN-based systems are compared with the DTW reference system.

That data sets are described in Chapter 2 and the keyword spotting systems in Section 9.2. We consider the training, validation, and test sets from Chapter 8, the text line normalization from Section 3.2.2, the geometric features from Section 3.3.2, the HMM architecture from Section 5.2.1, and the BLSTM NN architecture from Chapter 6.

In the following, the experimental setup is detailed in Section 9.3.1, the evaluation measure based on recall and precision is discussed in Section 9.3.2, and the results are presented in Section 9.3.3. Finally, a discussion of the results is provided in Section 9.3.4 and a summary and outlook are given in Section 9.4.

9.3.1 Setup

In this section, the experimental setup is discussed with respect to the data sets, keyword spotting systems, and previous publications. Essential keyword statistics are listed in Table 9.1. Besides the number of keywords that are spotted, the number of training samples is shown with respect to character and keyword templates. The numbers illustrate the advantage of the proposed sub-word model approach. Instead of matching few keyword

templates with test images, a large number of training samples is available for each character in the same training set. Both the HMM-based and NN-based keyword spotting system can access the character samples in a segmentation-free manner within complete text line images. That is, no text line segmentation into words and characters is required prior to training.

GWDB The 20 pages are split into 10 pages for training, 5 for validation, and 5 for testing. We perform a 4-fold cross validation and report mean results. On average, 328 lines are used for training, 164 for validation, and 164 for testing. For each cross validation, all keywords are spotted that appear at least once in the training and test set. This ensures that the DTW reference system has at least one word template for spotting and the test set has at least one relevant text line containing the keyword. On average, 105 keywords are spotted.

PARDB 2237 lines are used for training, 912 for validation, and 1328 for testing. From the 3220 words that appear in the training set, all 882 words that appear at least once in the test set are used as keywords.

IAMDB 6161 lines are used for training, 920 for validation, and 929 for testing. From the 4000 most frequent words in the database, we first select all 3421 non stop words[4]. From the non stop words, all 1217 words that appear at least once in the training and test set are used as keywords.

HMM The number of HMM states are set for each character individually with respect to the estimated mean width. Except for the space character "sp", they are adopted from experiments on text line recognition, which are detailed in Chapter 8. Since the space character is important for separating the keyword from the rest of the text line, the number of states, i.e., the minimum width of the word spacing, are optimized explicitly over $S \in \{5, 10, 15, 20\}$ with respect to a keyword spotting task on the validation set[5]. The number of Gaussian mixtures $G \in \{1, 2, \ldots, 20\}$ are optimized with respect to the same task. The optimal values (S, G) were $(15, 7)$ for the GWDB, $(10, 15)$ for the PARDB, and $(10, 12)$ for the IAMDB.

NN 10 randomly initialized networks are trained for the GWDB for each cross-validation, 10 for the PARDB, and 50 for the IAMDB. The best performing network is chosen on the validation set with respect to the same keyword spotting task used for HMM optimization.

[4]Frequent words like "the" and "a" are usually not interesting as individual search terms and are therefore removed from the keyword list. We used the stop word list from the SMART project [201], which can be found at `http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop`.

[5]The 10 most frequent words of the validation set are used as keywords.

DTW For the DTW reference system, no training and optimization is required. Instead, the keyword template images from the training set are used directly for image matching.

Previous Publications The experimental setup corresponds with the one used in [76]. For NN, the set of keywords is slightly different from the one used in [86]. In addition to the average precision (AP) reported in [86], we also consider the mean average precision (MAP) over all keyword queries for evaluating the NN-based system. These performance measures are introduced in the next section.

9.3.2 Evaluation

For measuring the performance, we consider two possible scenarios. In the first scenario, a *local threshold* is used for each keyword separately. For a given threshold T, all text lines whose score exceeds the threshold in Equations 9.3 and 9.5 are returned as positive matches[6]. With respect to the ground truth, the number of *true positives (TP)*, *false positives (FP)*, and *false negatives (FN)* are evaluated for all possible thresholds. From these values, the *recall* R and *precision* P of the word spotting system are obtained. They are given by

$$R = \frac{TP}{TP + FN} \tag{9.8}$$

$$P = \frac{TP}{TP + FP} \tag{9.9}$$

Recall reflects the fraction of retrieved keywords and precision reflects the correctness of the returned text lines. Usually, a trade-off between recall and precision has to be made by choosing an appropriate threshold. For retrieving most keywords, small thresholds can be used and for obtaining precise results, large thresholds can be used. In this thesis, we consider the *recall-precision curve* at recall $0.0, 0.1, \ldots, 1.0$ using the popular `trec_eval`[7] software. Examples are shown in Figure 9.4. The *average precision* (AP), an approximation of the area under curve, is used as a single value for measuring the system's performance. For local thresholds, we report the *mean average precision* (MAP) of `trec_eval` over all keyword queries as it is usual in the literature.

In the second scenario, a *global threshold* is used that is independent of the keyword. For a single query and all possible global thresholds, the recall-precision curve as well as the AP are calculated in the same way as for local thresholds. Note that using a global threshold for all keywords is

[6] For the DTW reference system, the score in Equation 9.7 has to be below the threshold.
[7] http://trec.nist.gov/trec_eval/

Table 9.2: Keyword spotting results. MAP is the *trec_eval* mean average precision over all keyword queries in case of local thresholds. AP is the average precision in case of a global threshold.

Database	System	MAP	AP
GWDB	NN	**91.5**	**81.4**
	HMM	79.3	62.1
	Reference	54.1	44.0
PARDB	NN	**94.5**	**95.0**
	HMM	88.1	85.5
	Reference	36.9	39.2
IAMDB	NN	**91.1**	**82.5**
	HMM	68.9	47.8
	Reference	12.7	4.1

sound because the text line scores of the proposed systems and the reference system are normalized with respect to the keyword length.

In a real-world situation, the proposed word spotting system can be used in both scenarios. For a vocabulary of common keywords, local thresholds can be optimized at training stage. For arbitrary out-of-vocabulary keywords, a global threshold has to be applied.

9.3.3 Results

The performance of the proposed HMM-based and NN-based keyword spotting systems as well as the DTW-based reference system are listed in Table 9.2. Besides the standard evaluation with respect to the mean average precision (MAP) over all keyword queries, we also provide the average precision (AP) when a global threshold is used for all keywords. The results are visualized in Figure 9.5. All MAP improvements of NN over HMM and HMM over DTW are statistically significant over all keyword queries (t-test, $\alpha = 0.05$). The recall-precision curves of the global threshold scenario are shown in Figure 9.4. The NN-based keyword spotting achieves the best performance for all databases, i.e., a MAP of 91.5% for the GWDB, 94.5% for the PARDB, and 91.1% for the IAMDB.

For the GWDB, the smallest difference between the systems can be observed. Here, the small amount of training data available imposes a problem for the proposed learning-based approach. Still, a better performance is achieved by HMM-based and NN-based spotting than by the template-based reference system.

For the PARDB, the best performance is achieved with the proposed

systems compared with the two other data sets. The relatively large amount of training data allows keyword retrieval with a high precision. Interestingly, the DTW-based reference system achieves worse results on the PARDB than on the GWDB despite the fact that a larger number of template images are available per keyword (see Table 9.1).

On the modern IAMDB, the DTW-based reference fails to cope with the large number of different handwriting styles. The HMM-based and NN-based systems clearly show a generalization capability. This generalization capability is also important for historical documents if several different manuscripts are processed by the same keyword spotting system.

When using global thresholds, a lower performance is achieved for the proposed systems in general. Following the reasoning from Section 9.2.1, this can be explained by the fact that the word prior $\log p(W)$ is no longer considered for scoring. Nevertheless, remarkable results are achieved in this out-of-vocabulary scenario.

9.3.4 Discussion

Overall, the reported results are promising for real-world applications of the proposed learning-based systems both in terms of average precision and computational complexity. In the literature, it was often argued against word spotting systems that rely on trained character models based on the argument that they share the same limitations as transcription-based handwriting systems [190]. The three major arguments are, first, that they have a slow computational speed, secondly, that they need a large amount of training data, and thirdly, that they cannot be easily adapted to new languages and alphabets.

Regarding the first argument, it is shown in Sections 9.2.1 and 9.2.2 that the proposed systems do not suffer from the high computational requirements imposed by a lexicon. Instead, character models are used to efficiently score text line images. The independence of a lexicon is particularly interesting for historical documents since the acquisition of such lexicons from electronic text editions might not be feasible for special historical languages.

The second argument is proven wrong in the experimental scenario considered for the GWDB where it is shown that even in case of a small amount of available training data, the proposed systems are able to outperform the template-based DTW reference system.

The third argument holds true to some degree and shows a limitation of the proposed approach. If the language under consideration and, in particular, its alphabet are unknown, no character models can be trained. Instead, the only option available might be template matching.

In the remainder of this section, we compare the achieved performance on the GWDB with other approaches reported in the literature.

Table 9.3: Related GWDB results. The size of the training set is indicated in terms of words and pages. MAP is the mean average precision over all keyword queries.

System	Train	MAP
Howe et al. [105]	18 p.	79.5
Rath et al. [180]	18 p.	54.0
Rath et al. [183]	1 w.	41.0
Rothfeder et al. [195]	1 w.	36.2
NN [86]	10 p.	**91.5**
HMM [76]	10 p.	79.3
Reference	10 p.	54.1

Related GWDB Results Several results were reported in the literature with respect to the same 20 pages of the George Washington database. However, there are significant differences in the keyword spotting task, the ground truth, and the evaluation method used. In Table 9.3 we indicate several reported MAP values that are closely related to our results. Four references are displayed in the upper part of Table 9.3 and our own results, including the DTW reference system, are listed below.

In [183], Rath et al. propose template-based keyword spotting using DTW in combination with a subset of the geometric features of our DTW reference system. The result indicated in Table 9.3 is obtained with respect to 10 relatively clean pages. Each word image is used as a keyword template and is employed to score all remaining word images. The reported MAP over all keywords is 41.0%. For the same task, Rothfeder et al. report a MAP of 36.2% in [195] using corner feature correspondences[8]. The MAP of our DTW reference system is 54.1% with respect to keyword templates from 10 training pages and keyword spotting in text lines from 5 test pages. This result is based on a manually corrected word segmentation. In contrast, an automatic segmentation is taken into account in [183, 195]. The comparison of the results demonstrates that the chosen DTW reference system is a strong reference with respect to template-based keyword spotting.

Other template-based keyword spotting results include two reports on spotting a special selection of 15 keywords. Although they are not directly comparable to our results, since only a small subset of the keywords is investigated, we list the reports here for the sake of completeness. In [231], Terasawa and Tanaka propose oriented gradient features in combination with

[8]When the keyword template is included in the test set, a significantly higher MAP is reported for both DTW and corner feature correspondences. With respect to 10 more degraded pages, a significantly lower MAP is reported.

DTW for keyword spotting. Using continuous dynamic programming, the DTW approach is generalized to spotting keywords in text line images. For the 15 keywords, a MAP of 79.1% is reported. In [138], Leydier et al. employ cohesive elastic matching of differential features to identify keywords in page images. In this promising segmentation-free approach, no segmentation into text line images is required. For the 15 keywords, an R-precision of 60.0% is reported[9].

Closely related to our results based on HMM and NN, two reports are listed in Table 9.3 that also employ a learning-based approach to spot keywords in text line images. Rath et al. report a MAP of 54.0% in [180] for a statistical approach based on image feature vocabularies when performing a 10-fold cross validation with 18 training pages and 2 test pages. For the same task Howe et al. report a MAP of 79.5% in [105] based on decision tree classification of word images. All keywords are taken into account that appear at least once in the training and the test set in the same way as in our experimental setup. However, we use only 10 pages for training, 5 pages for validation, and 5 pages for testing in a 4-fold cross validation. Disregarding the smaller size of the training set, the proposed HMM-based keyword spotting approach achieves a MAP of 79.3%, which is close to the best result reported so far in [105]. With NN-based keyword spotting, the achieved MAP of 91.5% is, to the knowledge of the author, the best result ever reported for the George Washington database.

9.4 Summary and Outlook

In this chapter, we present two novel keyword spotting systems that are based on trained character models. One system uses HMM characters and the other uses NN characters. Based on geometric sliding window features, the trained character models are employed in a lexicon-free approach to identify arbitrary keywords within text line images based on a confidence score. For HMM, the proposed confidence corresponds to the log-odds between a keyword HMM and a general filler HMM. For NN, the character posteriors are used directly as confidence.

Compared with previous word spotting methods, the proposed sub-word model approach has several advantages. First, the systems are segmentation-free at training stage as well as at recognition stage, i.e., they are not dependent on a segmentation of text lines into words which can be prone to errors. An advantage over learning-based methods at word level is given by the fact that only a small number of character classes needs to be trained for which a rather large number of training samples is available in a given handwritten text. Furthermore, the systems are able to spot arbitrary keywords that are not required to be known at training stage. Compared with transcription-

[9] R-precision is the precision at the point where recall equals precision.

based methods, the systems have the advantage that no lexicon is needed. First, such a lexicon might not be available for historical languages and secondly, lexicon-based recognition imposes a high computational effort for large vocabularies underlying natural language.

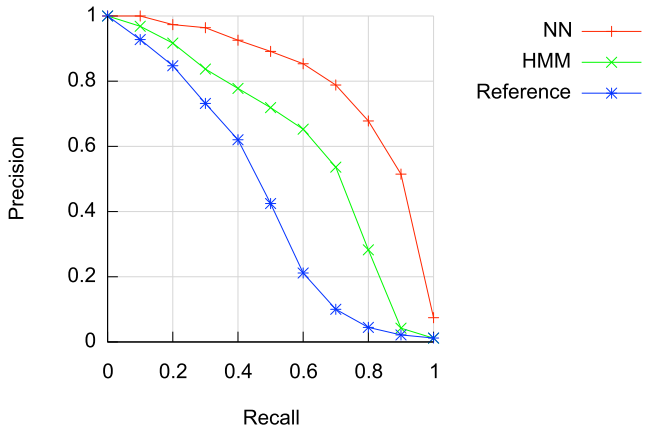
A known disadvantage of the proposed approach is, however, that it requires transcribed text line images for training which implies considerable human effort for historical documents. Furthermore, template-based image matching might be the only option available if neither language nor alphabet are known for a historical document. Finally, lexicon-based transcription is expected to outperform character-based spotting in terms of accuracy if reliable lexicons and language models are available.

An experimental evaluation on the historical George Washington and Parzival database as well as on the modern IAM database provides empirical evidence that the proposed learning-based systems significantly outperform a well-established template-matching approach. Among the proposed systems, the best results are achieved with NN-based spotting which outperforms HMM-based spotting. The overall best performance is achieved for the Gothic minuscules of the Parzival database. Here, we report an average precision of 95% with NN-based spotting and global thresholding. For the George Washington database, the achieved 91.5% mean average precision with NN-based spotting and local thresholding is, to the knowledge of the author, the highest score ever reported for this data set. On the modern IAM database, the generalization capability of the keyword spotting systems to unseen writing styles is demonstrated, which may also become important for historical documents when the spotting systems are used across different manuscripts.

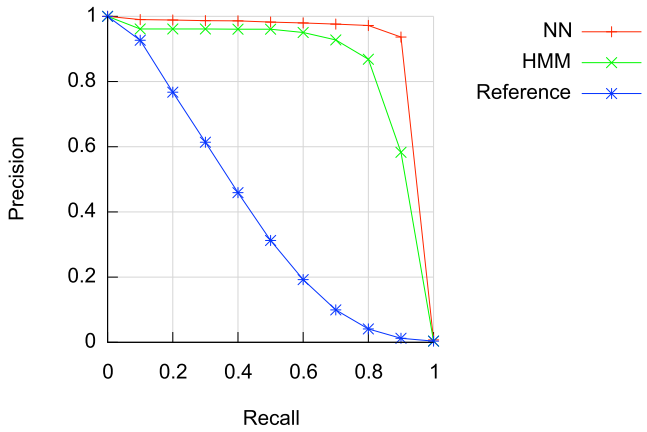
The keyword spotting results are promising for real-world application such as content-based indexing of historical manuscripts in digital libraries. Besides the high recall and precision scores, an important advantage over automatic transcription is given by the fact that no lexicon and language model is needed, which might be difficult to obtain. Instead, character models are sufficient for the identification of specific indexing terms. For large archives of historical documents, the high computational speed of the proposed systems is a great advantage. Considering for instance the entire George Washington collection, over 140,000 pages need to be processed [183]. In this context, we can provide a realistic time estimate with respect to a current personal computer disregarding all processing steps that are independent of a lexicon. Assuming that each page contains 20 lines and automatic transcription takes 1 minute per line, over 5 years are needed to extract content information. In contrast, when keyword spotting is achieved in 1 millisecond per line (see Sections 9.2.1 and 9.2.2), content information can be extracted in three quarters of an hour.

A promising line of future research is given by investigating alternative confidence models since their reliability is strongly related with the final

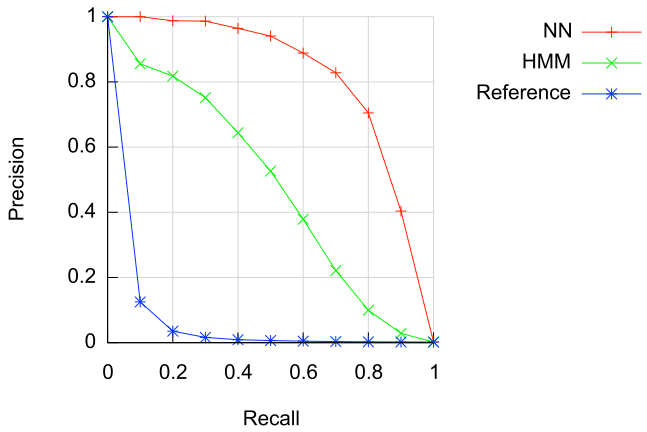
performance of the system. Furthermore, an open question concerns the situation where a lexicon and a language model are, in fact, available. In this scenario, an efficient integration of language model information into keyword spotting as well as a direct comparison with fully-fledged automatic transcription would be interesting. Finally, we have only reported spotting results for individual words as search terms. Further studies could take more complex queries into account, e.g., regular expressions.



(a) GWDB

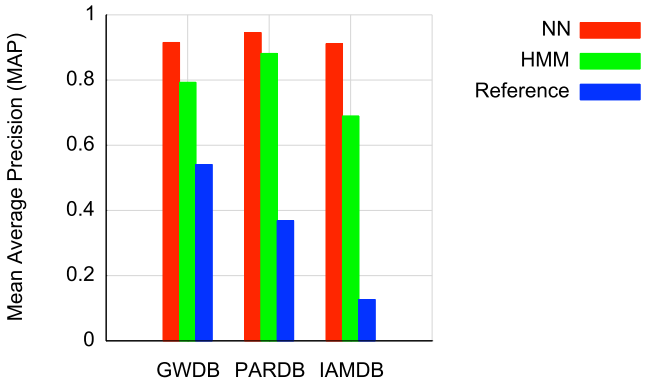


(b) PARDB

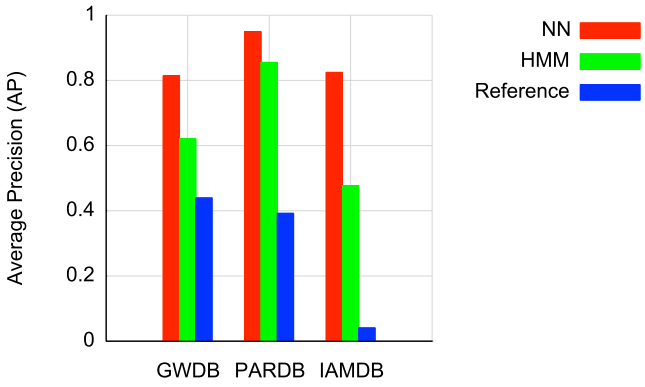


(c) IAMDB

Figure 9.4: Spotting Precision with Global Threshold



(a) Local Thresholds



(b) Global Threshold

Figure 9.5: Keyword Spotting Results

Chapter 10

Transcription Alignment

The acquisition of training data for handwriting recognition in historical documents is one of the major challenges in the field (see Section 1.4). The training data, consisting of pairs (I, T) of text images I and their machine-readable transcription T, has to be acquired from actual manuscripts rather than from special acquisition forms used for modern handwriting. First, this requires automatic methods for layout analysis and text image extraction. In difficult cases, human interaction may be needed. Secondly, an accurate transcription of the text images can often be provided by experts in historical languages only. Hence, the acquisition of training data involves time-consuming and expensive human interaction, which limits current endeavors towards mass digitization of historical manuscripts.

A standard approach to efficient training data acquisition includes interactive ground truth creation (see Chapter 7). Using semi-automatic pattern recognition methods, the efficiency of human users is increased for text image extraction and transcription assignment. Another promising approach is given by semi-supervised learning (see Section 1.2.2). In a supervised step, handwriting recognition systems are initialized with a small amount of training data (I, T). In a subsequent, unsupervised step, the handwriting recognition systems attempt to improve themselves without human interaction by taking text images into account that are extracted automatically and are not labeled with their transcription $(I^*, -)$. The starred I^* indicates that it remains unclear whether or not the image is extracted correctly since no human supervision is involved. Proposed methods include the automatic acquisition of additional training data in form of confidently recognized text images (I^*, T^*) [82]. Again, the starred T^* indicates that the recognition result attached to the image is not checked by human supervision. Semi-supervised learning makes use of the fact that text images without transcriptions $(I^*, -)$ are abundantly and readily available whenever automatic layout analysis and text image extraction is feasible.

An innovative approach based on massive-scale online collaboration has

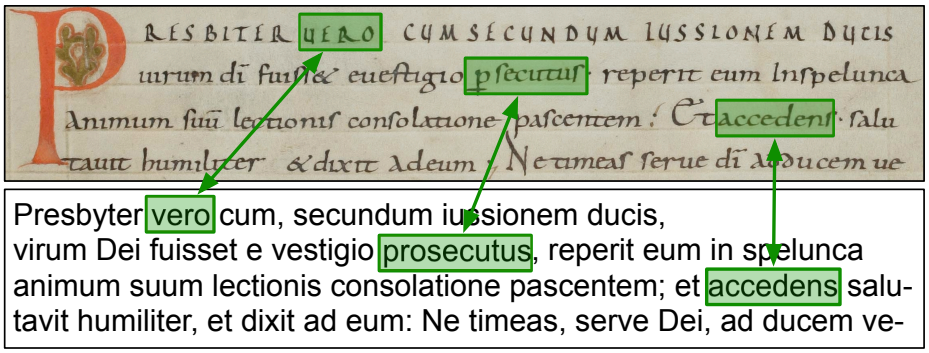


Figure 10.1: Transcription Alignment

recently been introduced in [248] for printed historical documents. The well-known CAPTCHA web security measure is considered that requires human users to enter the transcription of a heavily distorted text image in order to access an Internet service. Since current OCR fails for heavily distorted texts, automated programs can be prohibited from abusing Internet services[1]. In the reCAPTCHA framework, the human users are presented with two text images, one for security reasons, for which the correct transcription is known, and the other for transcribing historical prints. If several human users agree on the transcription of the same text image (I, T), it can directly be used in digital libraries with high confidence. According to an estimate in [248], over 100 million CAPTCHAs are solved every day and hence the potential is great to obtain large amounts of transcriptions following this approach. Principally, this method could also be used for the acquisition of training data in case of historical handwritings. However, in contrast to printed documents, the task might prove too difficult for the general public in case of historical handwritings. For special historical languages and scripts, expert knowledge is needed in order to provide an accurate transcription.

In this chapter, we present methods that aim at accessing machine-readable transcriptions, which were made by experts in the past and are now readily available. The idea is to align existing electronic text editions with manuscript images in order to create training data for handwriting recognition in historical manuscripts fully automatically. An example is shown in Figure 10.1. On top, a text image from the Saint Gall database is shown (Cod. Sang. 562, p. 25) and the corresponding part of the electronic Patrologia Latina text edition is displayed below (see Section 2.1).

[1]CAPTCHA stands for *Completely Automated Public Turing test to tell Computers and Humans Apart*. Its establishment as a widely used web security mechanism supports the argument that the transcription of difficult text images can, indeed, be regarded as a hard problem in artificial intelligence nowadays.

The goal is to establish links between word images and their transcription as illustrated in Figure 10.1. Each link corresponds to a pair (I^*, T^*), which can be used for training handwriting recognition systems. Similar to semi-supervised learning, neither the image extraction I^* nor the transcription assignment T^* are supervised by humans.

The challenge faced in this scenario is that text editions often do not correspond exactly with the handwriting image. Instead of transcribing each word and each character, human experts often write out abbreviations, correct mistakes, insert punctuation marks, change the capitalization, or even rephrase whole sentences for better readability of the text edition. All deviations from the actual word spelling in such *inaccurate transcriptions* pose challenges for extracting correct training samples automatically. An example is shown in Figure 10.1 for the word "prosecutus", which is written out full in the text edition but is abbreviated in the handwriting image[2]. Consequently, the alignment $(I^*, prosecutus)$ should be rejected by the system since the spelling does not correspond with the image[3]. An even better result would be to return the correct transcription label $(I^*, psecutus)$. Besides the possible transcription inaccuracy of electronic text editions, an additional problem is given by missing line break information. Text editions typically do not represent line breaks in the image. Instead, they have to be detected automatically.

Overall, recognition-free approaches, i.e., detecting gaps between words based on certain gap metrics, are not well-suited for aligning inaccurate transcriptions. Instead, handwriting recognition is needed to provide reliable results in case of missing words, additional words, and wrong word spellings. The methods proposed in this thesis are based on handwriting recognition systems that are initialized with a small set of training data (I, T) similar to the semi-supervised learning scenario. Instead of generating additional training samples by means of automatic transcription, existing electronic texts are aligned with manuscript images, which can be considered less demanding. In particular, the vocabulary and the order of recognizable words are drastically constrained with respect to the given electronic text edition.

In the following, the task of aligning inaccurate transcriptions is described in more detail in Section 10.1 including an evaluation measure and a discussion of related work. Afterwards, two HMM-based alignment systems are proposed in Sections 10.2 and 10.3 that target different error conditions of the texts. A summary and outlook to future work are given in Section 10.4.

[2]The term "abbreviation" is used in this thesis for words that are written with a reduced number of letters. We do not distinguish the linguistic semantics of such an abbreviation.

[3]At least for training character-based recognition systems it should be rejected. For template-based keyword spotting, for instance, returning the pair $(I^*, prosecutus)$ as a template image would make perfectly sense (see Chapter 9).

10.1 Task Description

The task of aligning inaccurate transcriptions is presented in this section in more detail. Conditions are stated in Section 10.1.1, objectives are formalized in Section 10.1.2, and an evaluation measure is introduced in Section 10.1.3. Afterwards, some constraints applied in this thesis are discussed in Section 10.1.4. Finally, related work is discussed in Section 10.1.5. Regarding the inaccuracy of the transcriptions under consideration, the task has, to the knowledge of the author, not been investigated before.

10.1.1 Conditions

Three conditions need to be satisfied for a successful application of the proposed methods.

1. An electronic text edition of the manuscript has to be available. It does not have to correspond exactly with the handwriting image. In particular, the alignment systems can cope with missing words, additional words, wrong word spellings, and missing line break information.

2. An automatic extraction of text line images from manuscript pages needs to be feasible with high accuracy since it is performed automatically without human supervision. The segmentation into word images, which is prone to errors without handwriting recognition, is performed by the alignment systems.

3. A number of initial training samples (I, T) has to be provided in order to initialize the recognition-based alignment systems. Experimental results obtained in Section 10.3 for Latin manuscripts indicate that high alignment accuracy can already be achieved for very small initial training sets, i.e., one or few manuscript pages.

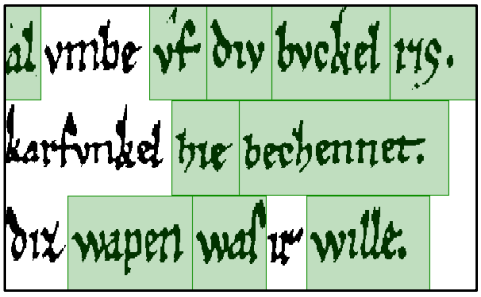
10.1.2 Objectives

The goal of aligning inaccurate transcriptions is to automatically extract as many pairs (I, T) of word images I alongside with their correct transcription T as possible (see Figure 10.1). Each such pair constitutes a training sample for handwriting recognition in historical documents.

An optimal alignment, i.e., the ground truth of the problem, consist of a sequence

$$(I_1, T_1), \ldots, (I_N, T_N)$$

with a maximum length of N correctly segmented word images I and words from the text edition T that correspond with the correct transcription. Word images are not included in the sequence if they have no correspondence in the text edition. Correspondence means that, first, the word spelling is



al trv̊g solhem v̂f div bvckel rîs. hie bechennet. kǒkesas heiden wapen was wille.

Figure 10.2: Optimal Alignment

correct and, secondly, the word order is not violated. Vice versa, words in the text edition without correspondence to an image are not included as well. An example is shown in Figure 10.2. On the left, three text line images from the Parzival database are displayed and on the right hand side, an electronic text edition is shown[4]. Corresponding words are highlighted in the image and underlined in the text edition. Altogether, the optimal alignment includes $N = 10$ pairings $(I_1, al), \ldots, (I_{10}, wille)$.

Note that constraining the optimal alignment to the correct word order cannot cope with swapped text. For instance, a final "diz" at the end of the text edition cannot be linked back to the word image on the bottom left in Figure 10.2. This leads to a loss of training samples (I, T) in case of swapped words, sentences, or paragraphs. We explicitly take such losses into account in order to state a simplified problem, which is expected to be solvable with higher accuracy. A more general definition could state the alignment problem as an assignment problem (see Section 4.3), which would take text swapping into account.

The ground truth can be established by means of string edit distance (SED) [249] between the correct transcription and the electronic text edition. The concept of edit distance is detailed in Section 4.3. For ground truth creation, costs 1 are used for word insertion and deletion. Non-identical words are substituted with costs 3 (insertion and deletion is always preferred) and identical words are substituted with costs 0. The optimal alignment corresponds with the substitutions performed in the edit path with minimal costs (optimal SED). Taking into account the correct boundaries of the word images, the resulting ground truth is used for evaluating the automatic alignment systems.

[4]The text edition is artificial. It is generated by random distortions of the perfect transcription as detailed in Section 10.2.

10.1.3 Evaluation

The result of the automatic alignment systems is a sequence

$$(I_1^*, T_1^*), \ldots, (I_n^*, T_n^*)$$

of n pairs (I^*, T^*) of segmented word images I^* together with words from the text edition T^*. The extracted word images can contain segmentation errors, i.e., wrong start and end positions within the text line images. The assigned words from the text edition can deviate from the correct transcription.

For evaluation, SED is used for comparison with the ground truth (see Section 10.1.2). Edit operations include deletions, insertions, and substitutions of ground truth pairs (I, T) and result pairs (I^*, T^*). Deletions signify missing pairs (they should have been returned) and insertions stand for additional pairs (they should not have been returned). Both are performed with costs 1. Substitutions of pairs with different word labels have cost 3 (insertion and deletion is always preferred), substitutions of pairs with equal word labels but wrong word segmentation have cost 1, and substitutions of perfect matches have cost 0. Similar to the word accuracy in case of automatic transcription (see Section 5.2.4) we define the alignment accuracy with respect to the optimal SED as

$$A = \frac{N - S - D - I}{N} \tag{10.1}$$

where N is the number of words in the ground truth, S is the number of wrong word segmentations (substitutions with costs greater than zero), D is the number of missing words, and I is the number of additional words. As for the word accuracy of transcription alignment, the alignment accuracy can become negative for large amounts of insertions.

For evaluating the correctness of the word segmentation, we employ the measure proposed in [115]. Here, the word start and end positions are required to be in within the space character before and after the word. The ground truth for word segmentation is created with HMM-based forced alignment and manual correction (see Chapter 7).

Besides the accuracy, precision and recall (see Chapter 9) can also be determined directly from the SED result. In the context of alignment, recall is the fraction of retrieved alignments and precision is the correctness of the returned alignments. They are given by

$$R = \frac{C}{C + D}, \; P = \frac{C}{C + S + I} \tag{10.2}$$

with respect to the number of correct words $C = N - S - D$.

10.1.4 Constraints

In order to focus on handwriting recognition, we constrain the alignment scenario in two ways. First, a perfect segmentation of the manuscript pages into text line images is considered. That is, potential errors stemming from text line extraction are not evaluated. Secondly, the text edition is considered to be aligned at page level. In real-world scenarios this might not be the case and such an alignment would need to be established prior to the methods discussed in this thesis, for instance based on spotting anchor text passages (see Chapter 9). The use of such text anchors has been proposed, e.g., in [68, 107, 233] to reduce the alignment scope to the space between two confidently recognized anchors.

In summary, we focus on two problems. First, the segmentation of text line images into words. Secondly, coping with inaccurate transcriptions with respect to missing words, additional words, wrong word spellings, and missing line break information. Both problems are solved conjointly in the proposed recognition-based systems.

10.1.5 Related Work

In case of a *perfect transcription* that contains each character and word that is present in the handwriting image, an alignment can be found by segmenting the image into a known number of words. This scenario has been extensively studied in the literature.

Without performing handwriting recognition, gap metrics can efficiently be applied to segment text lines into words, especially if the number of words is known for each text line [223, 262].

Another widely used approach is to generate multiple hypotheses for word segmentation and use dynamic programming, e.g., DTW (see Chapter 9), to find an optimal alignment between the segmented word images and the transcription words. Distance measures between word images and transcription words can be determined directly using heuristic features [131, 146] or by means of a word recognizer [107, 233]. Similarly, word HMM are used in [196] to recognize word segments and Viterbi decoding is employed for alignment.

Finally, character HMM can be employed for forced alignment of a complete text line image without prior word segmentation [115, 237, 258]. This approach is detailed in Section 5.2.3 and is used in Chapter 7 for ground truth creation.

The problem of *inaccurate transcriptions* considered in this thesis has been mentioned, e.g., in [131], but to the knowledge of the author it has not been studied so far. The basic assumption of current systems to map each word in the transcription with a word image [233] needs to be relaxed to deal with missing words, additional words, and wrong word spellings in

real-world text editions. Therefore, an off-the-shelf reference system is not available for comparison.

The closest link to current approaches is given between the method proposed in [258] and the HMM-based system proposed in Section 10.3. Our method can be seen as a direct adaptation of the work presented in [258] when taking errors in the word spelling into account. For the more general method proposed in Section 10.2, we could not find a direct correspondence in the literature.

10.2 Inaccurate Transcriptions

In this section, a general alignment system for inaccurate transcriptions is presented, i.e., for electronic text editions that do not match the handwriting image perfectly. A novel multi-pass approach based on character HMM is proposed that combines text line recognition, string alignment, and keyword spotting in order to cope with missing words, additional words, and wrong word spellings in the given text edition. The approach is segmentation-free in two ways. First, extracted text line images do not need to be segmented into words prior to alignment, neither for training nor for recognition. Secondly, no line break information is needed in the text edition. However, the text edition is assumed to be pre-segmented into manuscript pages.

Based on trained character HMM, alignment is achieved in three consecutive steps for each manuscript page. In the *first pass*, text line recognition is performed with respect to the lexicon and the bigram language model of the given page from the electronic text edition. In the *second pass*, the recognition output is aligned with the text edition in order to extract a sequence of labeled word images that do not violate the word order. In the final *third pass*, the result is improved by means of keyword spotting. On the Parzival database, results are reported for several degrees of artificial distortions of the correct transcription.

In the following, the first alignment pass is presented in Section 10.2.1, the second pass in Section 10.2.2, and the third pass in Section 10.2.3. Afterwards, experiments on the Parzival database are described in Section 10.2.4 and results are given in Section 10.2.5. Finally, Section 10.2.6 provides a discussion of the results.

10.2.1 First Pass

In a first step, character HMM are trained based on an initial set of training samples as usual. Training data is given as text line images together with their transcription. The images are normalized according to Section 3.2.2 and a sequence of feature vectors is extracted according to Section 3.3. Then, Baum-Welch training is applied according to Section 5.2.2.

For the first alignment pass we now consider a complete manuscript page. The input is twofold. First, a sequence of text line images is given and, secondly, a sequence of words from the text edition without line break information is provided. The goal of the first pass is to segment the text line images into words and to assign a word label to each word image. The resulting sequence of pairs $(I_1^*, T_1^*), \ldots, (I_n^*, T_n^*)$ between word images I^* and recognition results T^* cover the complete manuscript page and are therefore expected to contain a number of alignment errors in case of inaccurate transcriptions, mostly in form of additional pairs (I^*, T^*) that should not be returned in the end. Nevertheless, a first alignment result is achieved.

For page recognition, a sequence of feature vectors is extracted from the text line images and a vocabulary $w_1, \ldots, w_L$ of all L distinct words from the page text is recorded. Also, word unigrams $P(w_i)$ and back-off smoothed word bigrams $P(w_i|w_{i-1})$ are calculated according to Section 5.3.1 in order to provide a statistical language model of the page text. Using the trained character HMM, the page vocabulary, and the page language model, Viterbi recognition of the text lines is then performed according to Section 5.2.2. Hereby, two different systems are taken into account.

System TLR The standard TLR system recognizes each text line individually. For the first word, unigram probabilities are used during recognition and for each subsequent word, bigram probabilities are used.

System TLR^* In order to retain language model information over several text lines, the system TLR^* considers the manuscript page as a single, very long text line by concatenating the feature vector sequences of all text lines. Unigrams are only used for recognizing the first word on a page and all subsequent words are recognized with respect to bigrams. Unlike TLR, erroneous word images may span over two consecutive text lines after recognition. Since the line break positions are known within the feature vector sequence of the page, this can easily be corrected in a post-processing step. That is, whenever no word boundary is present at a line break position, the nearest word boundary is moved to the line break position explicitly.

Considering small initial training sets, appearance-based recognition is expected to commit a number of errors. Therefore, the role of the language model becomes more important. If the available electronic text edition deviates only slightly from the correct transcription, a strong language model influence is desired, which can be emphasized in the TLR^* system by retaining language model information at the line breaks. For both TLR and TLR^*, the general influence of the language model can be adjusted with the grammar scale factor GSF during Viterbi recognition (see Section 5.3.1). For text editions, which are close to the transcription, high GSF values are

expected to perform best. For text editions, which deviate heavily from the transcription, low GSF values are preferred. In short, the GSF value determines how close the recognition result should resemble the given electronic text edition. Since this parameter strongly affects the alignment result, it should be optimized on a validation set with respect to the alignment accuracy.

In terms of computational speed, the text line recognition performed in this first pass is the bottleneck of the proposed alignment method. Its time complexity is $O(L^2N)$ for Viterbi recognition with respect to the vocabulary size L and the feature vector sequence length N (see Section 5.2.2). Since only the page vocabulary of the text edition is taken into account for each manuscript page, usually a few hundred words, it is magnitudes faster than large-vocabulary recognition in case of unconstrained natural language, usually performed with tens of thousands of words.

10.2.2 Second Pass

The alignment result obtained in the first pass covers the complete manuscript page. That is, the text line images are completely segmented into words and each word image I^* is labeled with a word from the text edition T^*. In the second pass, we delete the alignments that do not appear at the correct position with respect to the text edition. That is, we impose a binary confidence model on the alignments (I^*, T^*). High confidence is assumed if an alignment does not violate the word order in the text edition. Low confidence is assumed if an alignment is found at the wrong position. Alignment results with low confidence are discarded.

The confidence model can be justified as follows. Considering the scenario that part of the manuscript image is not represented in the text edition, it is unlikely that a valid word from the page vocabulary is recognized that does, by chance, not violate the word order. The effective chance of random success is dependent on the vocabulary size (low chances for large vocabularies) and the GSF value (low chances for low GSF values).

The word order is enforced by means of SED between the sequence of recognized labels $T_1^*, \ldots, T_n^*$ and the words in the text edition. The same costs for SED are used as detailed in Section 10.1.2 in the context of optimal alignment. Pairs (I^*, T^*) are discarded from the alignment result if their labels T^* do not occur at the correct position with respect to the word order in the text edition.

After the second pass, the alignment result does no longer cover the entire manuscript page as illustrated in Figure 10.2. For each discarded alignment, a gap results in the image, i.e., image parts to which no text is assigned. On the other hand, we obtain gaps in the text edition, i.e., text parts to which no image is assigned.

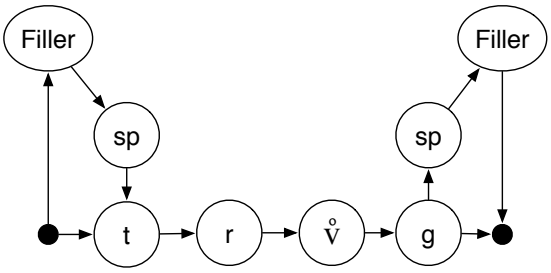


Figure 10.3: Third Pass

10.2.3 Third Pass

For each image gap in the alignment result after the second pass, the SED provides a list of candidate words from the transcription that could be matched to the image. For example, the words "trv̊g" and "solhem" are candidates for the first image gap in Figure 10.2. If the candidate list is not empty, it might contain one or several words that are, in fact, present in the image but were not correctly recognized in the first pass, e.g., due to image noise or because of a handwriting style that is not well represented by the character models. Therefore, confidence-based recognition is performed in a third and last pass in order to recover gap words with keyword spotting.

For each candidate word, a keyword HMM is constructed according to Section 9.2.1. In Figure 10.3, an example is shown for the word "trv̊g". The keyword is required to be present at the beginning, in the middle, or at the end of the gap observation sequence $\mathbf{x}$. The rest of the observation is modeled with a filler model that is given by a loop over all characters of the alphabet. Viterbi decoding with respect to the keyword HMM results in the most likely position of the keyword alongside with the keyword gap likelihood $P(\mathbf{x}|K)$. In a second recognition step the filler gap likelihood $P(\mathbf{x}|F)$ is obtained with respect to the filler model only. Then, the log-odds confidence model

$$\frac{P(\mathbf{x}|K)}{N \cdot P(\mathbf{x}|F)} > T \tag{10.3}$$

with respect to the gap observation length N is used for keyword spotting. That is, the gap word is added to the alignment result if the log-odds exceed a certain threshold T. Note that the rare case of overlapping spotting results amongst several candidate words is allowed. A refinement of the method would include a resolution of overlaps if large gaps and a large number of candidate words are frequent in a specific application.

The additional training samples added to the alignment result in the third pass are expected to improve the automatically generated training set not only quantitatively but also qualitatively. Since the retrieved alignments

do not correspond well to the trained character HMM, i.e., they were not recognized in the first pass, they contribute an additional image condition or writing style to the generated training set. Finding correct alignments in the third pass is supported by the narrow gap vocabulary, which consists of few candidate words obtained by SED alignment in the second pass.

Besides the GSF parameter used for the first pass, a second system parameter is given by the threshold T used for keyword spotting in the third pass. It determines the confidence needed for adding "badly written" words to the alignment results and should be optimized on a validation set with respect to the alignment accuracy.

10.2.4 Experiments

The proposed handwriting alignment system for inaccurate transcriptions is tested on the PARDB. We consider the text line normalization from Section 3.2.2, the geometric features from Section 3.3.2, the HMM architecture from Section 5.2.1, and the training, validation, and test sets from Chapter 8. That is, 2237 lines are used for training, 912 for validation, and 1328 for testing. For each manuscript page, the sequence of text lines that belong to the training set are considered as a training page. The validation and test pages are constructed in the same way. Each test page contains about 28 text lines on average.

In the following, the experimental setup is discussed with respect to the text editions, alignment systems, and system parameters considered.

Text Editions In order to obtain inaccurate transcriptions in a controlled manner, artificial distortions are applied to the correct transcription of the PARDB. For each page, the line break information is discarded and a sequence of words is taken into account. For each word of the sequence, an error is added randomly with probability $0 \leq \delta \leq 1$. Possible errors include word substitution, word deletion, and word insertion. For substitution and insertion, out-of-vocabulary words are used. Five different distortion degrees $\delta \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ are taken into account for experimental evaluation. In case of maximum distortion $\delta = 0.5$, half of the words are modified in the page transcription, either by substitution, deletion, or insertion. After distortion, an electronic text edition is obtained for each page that includes the desired errors.

For each test page, the transcription is randomly distorted five times. The reported results in Section 10.2.5 are given by the mean alignment accuracy. The proposed measure of alignment accuracy is detailed in Section 10.1.3. For the validation pages, which are used for parameter optimization, the transcription is distorted only once for each distortion degree.

Table 10.1: Transcription alignment errors.

System	S	D	I
TLR	121.4	260.2	22.8
TLR^*	127.6	207.2	26.2
KWS	137.4	126.6	33.4

Systems Three variants of the alignment system are tested and compared. First, standard text line recognition TLR in the first pass, followed by string alignment in the second pass. Secondly, the modified text line recognition TLR^* that concatenates all observation sequences in the first pass in order to retain language model information at the line breaks. And thirdly, the keyword spotting system KWS that includes the final third pass for retrieving words within image gaps in addition to the TLR^* system.

Parameters The validation set is used to optimize several system parameters. Standard parameters of the HMM architecture include the number of states and Gaussian mixtures G of the character HMM. The two parameters of the proposed alignment method include the GSF value in the first pass, which determines how close the recognition result should resemble the electronic text edition, and the keyword spotting threshold T in the third pass, which determines the recognition confidence needed for retrieving words within image gaps. Additionally, a word insertion penalty WIP is taken into account, which can often improve the recognition result of the first pass (see Section 5.3.1).

The number of HMM states are set for each character individually with respect to the estimated mean width. They are adopted from experiments on text line recognition, which are detailed in Chapter 8. The other parameters are optimized with respect to the alignment accuracy achieved on the validation pages. Due to the high computational speed of the alignment system, a comprehensive grid search was realized. We have tested $G \in \{1, 3, \ldots, 19\}$, $GSF \in \{0, 10, \ldots, 100\}$, and $WIP \in \{-100, -80, \ldots, 100\}$. All logg-odds that appear in the validation set were considered as possible keyword spotting thresholds T.

10.2.5 Results

The mean alignment accuracy achieved on the test set is shown in Figure 10.4 for the three systems TLR, TLR^*, and KWS and the different transcription distortion degrees δ. The achieved improvements of TLR^* over TLR as well as the improvements of KWS over both other systems are statistically

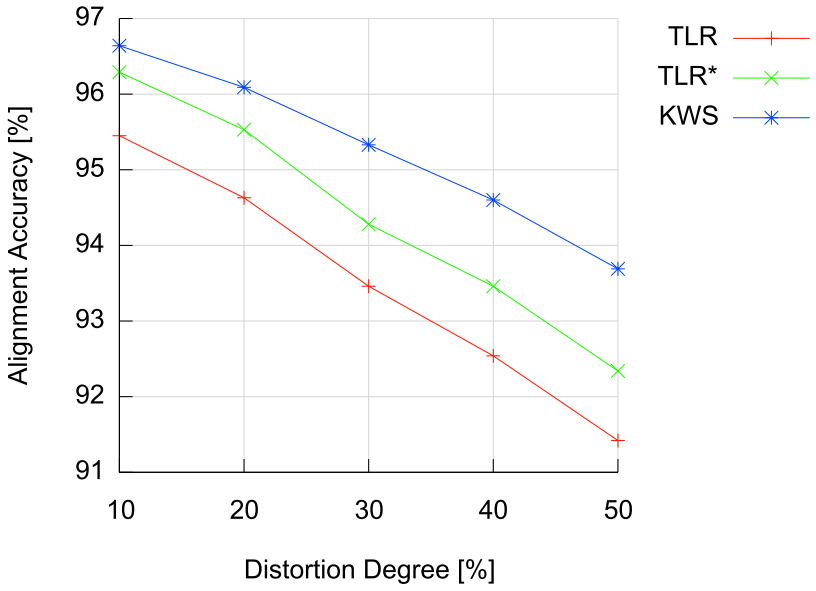


Figure 10.4: Transcription Alignment Results

significant for all distortions (t-test, $\alpha = 0.05$).

The error analysis given in Table 10.1 shows the behavior of the systems for the distortion degree $\delta = 0.5$. After distortion, $N = 4,714.8$ alignments (I, T) are expected to be returned on average for an optimal alignment. By retaining language model information at the line breaks, the TLR^* system returns more alignment results (I^*, T^*) than TLR. By filling recognition gaps by means of keyword spotting, KWS returns even more results. Although these additional results slightly increase the number of substitution errors S (wrong word image segmentation) and insertion errors I (wrongly returned alignments), the number of deletion errors D (missing alignments) can be reduced significantly. About 50% of the missing alignments are retrieved by KWS compared to TLR.

10.2.6 Discussion

The proposed multi-pass system for aligning inaccurate transcriptions shows a very promising performance. In an experimental evaluation on the Parzival database it was demonstrated that about 94% of the optimal alignments (I, T) could be returned correctly, even if each second word in the text edition was erroneous. Correctness includes an accurate segmentation of text lines into word images I as well as an accurate assignment of word transcriptions T. In case of only 10% erroneous words in the text edition, an alignment accuracy of about 97% is reported.

Since the task of aligning inaccurate transcriptions is novel, to the knowledge of the author, we could not compare the proposed system to other approaches from the literature. Instead, we provide an evaluation of the three different passes of the system given by text line recognition, string alignment, and keyword spotting. The final system that incorporates all three passes significantly outperforms intermediate system variants. Important factors for extracting accurate alignments are the page language model used in the first pass, the word order confidence model in the second pass, and the log-odds confidence model in the third pass. They all operate on top of appearance-based character HMM and have shown high potential to provide accurate alignments. The computational speed is magnitudes higher than for automatic transcription with large vocabularies, which underly natural language, due to the fact that the lexicon of recognizable words is limited to a small page lexicon from the text edition.

There are several open issues for real-world application with respect to the conditions formulated in Section 10.1. First, the results reported for the Parzival database are based on a perfect text line extraction from manuscript images. Any error in automatic layout analysis and text line extraction implies additional alignment errors.

Secondly, the electronic text edition is assumed to be aligned at page-level. If the text edition is not aligned with the manuscript images at all, anchor points are needed to establish alignment blocks between confidently recognized text. A promising approach is given by the keyword spotting methods presented in Chapter 9. They could be used to efficiently identify text anchors in form of several consecutive words from the text edition.

Finally, the artificial distortions of the transcription include an equal distribution of word insertions, deletions, and substitutions throughout the text edition. Error scenarios that differ from this condition were not investigated so far. For the special condition where no deletions appear at all, we present a different alignment system in the next section.

10.3 Incorrect Word Spelling

In this section, we concentrate on word substitution and insertion errors in the absence of deletion errors. This scenario assumes that the human expert may have changed the word spelling, e.g., when writing out abbreviations, and may have added additional words, e.g., when adding remarks. Otherwise, the text edition is assumed to correspond closely to the manuscript image. Without deletion errors, the alignment problem is essentially reduced to a word segmentation problem, since all words from the handwriting image appear in the text edition, possibly with spelling errors, and no image parts need to be rejected.

We have encountered this real-world error scenario for the Saint Gall

database. The goal is to align the Latin manuscript images of the Cod. Sang. 562 with the Patrologia Latina text edition (see Section 2.1). The electronic text edition is aligned at page-level with the manuscript images but deviates from the correct transcription as shown in Figure 10.1. First, line breaks are not given in the text edition[5]. Line break detection is complicated by the fact that no hyphens are used for breaking words at the line end. Secondly, word abbreviations are written out in full in the text edition. Last but not least, the capitalization and punctuation in the text edition does not match the handwriting image.

The predominant error observed in the text edition is given by word substitutions. The application of the general alignment system presented in Section 10.2 is not well-suited because word substitutions are rejected. For instance, the alignment result $(I^*, prosecutus)$ shown in Figure 10.1 is considered wrong, since the word label does not correspond with the transcription *psecutus*. On the one hand, a rejection is rendered difficult because the substituted word is very similar. On the other hand, more training samples could be extracted if the alignment is not rejected but instead corrected. Even if the correction fails, the result $(I^*, prosecutus)$ could be used for template-based keyword spotting disregarding the correct spelling (see Chapter 9). Therefore, we have developed a different alignment system for this particular scenario. The idea is to model the text edition more closely with a single page HMM, similar to the case of an error-free text edition, but to allow for spelling deviations.

In the following, the inaccuracy of the electronic text edition is described in more detail in Section 10.3.1. Then, the proposed alignment system is presented in Section 10.3.2. The experimental evaluation is outlined in Section 10.3.3 and results are reported in Section 10.3.4. Finally, Section 10.3.5 provides a discussion of the results.

10.3.1 Challenges

The Patrologia Latina text edition differs from the correct transcription in several ways.

- *Line breaks:* The line breaks are unknown in the text edition. Furthermore, 33.9% of the text lines contain a word break at the line end. No hyphen is present in the image at the word breaks.

- *Abbreviations:* Abbreviations are written out in full in the text edition. Hence, the spelling of the abbreviated words is unknown. Overall, 21.5% of the words are abbreviated.

[5] In Figure 10.1, line breaks were added manually to the text edition in order to improve the comparability with the handwriting image.

- *Capitalization:* The capitalization of the words in the text edition does not correspond with the image, i.e., the capitalization is unknown. Most of the text is written in Carolingian minuscules, yet 7.1% of the words contain capital letters.

- *Missing Text Lines:* 2.1% of the text line images are missing, because of errors in the interactive text line extraction (see Chapter 7)[6]. Because of these errors, 1.2% additional words are present in the text edition. Note that there are no line segmentation errors for the remaining text line images.

- *Other Issues:* The punctuation marks are unknown in the text edition. Also, the width of the word spacing shows considerable variation. Figure 10.1 gives an impression of this issue. Especially in case of prepositions, word spacing is often omitted completely. An example is given for "ad eum" and "ad ducem" in the last line of Figure 10.1.

10.3.2 System

The common HMM-based scenario is considered for character modeling. That is, character HMM are trained based on an initial set of training samples. Training data is given as text line images together with their transcription. The images are normalized according to Section 3.2.2 and a sequence of feature vectors is extracted according to Section 3.3. Then, Baum-Welch training is applied according to Section 5.2.2.

For alignment, a complete manuscript page is taken into account. The input consists of a sequence of text line images and a sequence of words from the text edition without line break information. Unlike the general alignment system presented in Section 10.2, missing words in the text edition do not have to be taken into account. Instead, only word substitutions and, in few cases, additional words have to be considered (see Section 10.3.1). Therefore, a single page HMM is constructed that returns an alignment $(I_1^*, T_1^*), \ldots, (I_n^*, T_n^*)$, which covers the complete page image. That is, each part of the manuscript image is assigned to a word image I^* and each word image is labeled with a recognition result T^*. The recognition result T^* corresponds to the most likely spelling variant of a word from the text edition. The word order in the text edition is not changed but the deletion of some additional words is allowed.

The page HMM is constructed from the text edition as follows. First, all letters are converted to lower case and all punctuation marks are removed

[6]Some of the users of the ground truth creation software have considered headings written in capital letters not to be part of the main text and did not extract them. In version 1.0 of the Saint Gall database, this inconsistency is not corrected. Instead, it is regarded as an additional challenge for the alignment task, since missing text lines are to be expected in case of fully automatic text line extraction.

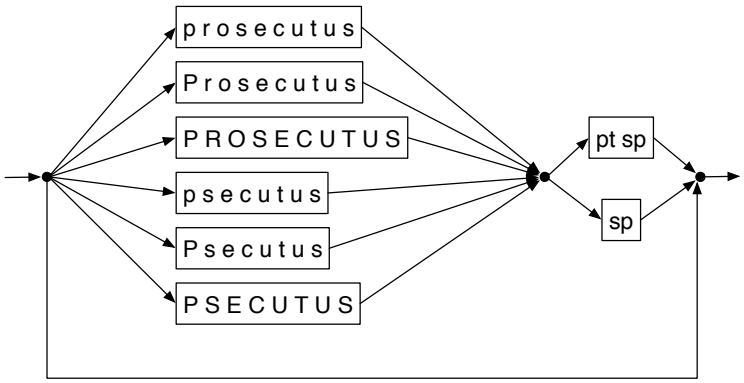


Figure 10.5: Alignment HMM

from the text edition, since neither capitalization nor punctuation are expected to match the manuscript image. Next, several spelling HMM are created for each word in the text edition. Besides the lower case character sequence, the first letter can be capitalized and, for headings, the whole word as well. Also, we add known spelling variants from the training set. An example is shown in Figure 10.5 for the word *prosecutus* and the variant *psecutus* that is assumed to be known. The page HMM is then given by the concatenation of all transcription words according to Figure 10.5. The word spacing is modeled with and without punctuation using the character models "sp" for white space and "pt" for punctuation. We also allow for word deletion to take into account additional words in the text edition due to missing text line images.

A special case is given when a character from the text edition does not appear in the training set and therefore no character HMM is available. In this case, the character is replaced with the general upper case HMM "UC" or the lower case HMM "lc". They are trained separately on the training set by replacing all transcription characters with "UC" and "lc", respectively.

For HMM-based recognition, the feature vector sequences of all text lines are first concatenated into a single page sequence. Then, the optimal word boundaries are found by means of the Viterbi algorithm with respect to the page HMM (see Section 5.2.2). Whenever no word boundary is present at a line end, the distance between the nearest word boundary and the line end position is checked. If it is very small, the word boundary is moved explicitly to the line end position. Otherwise, a word break is assumed and no correction is performed.

Viterbi recognition with the proposed page HMM is very efficient. The time complexity is $O(N^2T)$ with respect to the number of spelling variants N and the feature vector sequence length T. Therefore, the alignment is magnitudes faster than automatic transcription with large vocabularies

Table 10.2: Training sets.

Set	%Letters	%Spellings	Train_{lc}	Train_{uc}
T-1	54.2	1.0	28.5	1.0
T-20	97.9	15.4	837.9	31.3

underlying natural language, where N is the size of the vocabulary. In particular, the proposed alignment method is faster than the more general system presented in Section 10.2, where N corresponds to a few hundred words from the page vocabulary.

10.3.3 Experiments

Experiments are conducted on the SGDB. We consider the text line normalization from Section 3.2.2, the geometric features from Section 3.3.2, the HMM architecture from Section 5.2.1, and the training, validation, and test sets from Chapter 8. Each set consists of several manuscript pages. Up to 20 pages are used for training, 10 pages are used for validation, and 30 pages are used for testing. Page images are given as a sequence of text lines and the page text edition is given as a sequence of lowercase words without punctuation marks and without line break information. The spelling variants used for alignment consist of all word transcriptions in the training set that deviate from the lowercase words in the text edition.

The goal of the proposed alignment method is to provide reliable results with a small amount of initial training data. Therefore, different sizes of the training set are considered. The investigated number of training pages is $T \in \{1, 5, \ldots, 20\}$. For $T = 1$ and $T = 20$, statistics of the training sets are given in Table 10.2 including the percentage of known letters and known special spellings with respect to the test set, and the amount of training samples per lower case letter (Train_{lc}) as well as per upper case letter (Train_{uc}).

In the following, the experimental setup is discussed with respect to the evaluation measure, alignment systems, and system parameters.

Evaluation With respect to word spelling variants, the ground truth and the alignment accuracy are slightly different from Section 10.1.3. Instead of pairs (I, T), triplets (I, T, L) are taken into account that include the lowercase word label L from the text edition, which can deviate from the transcription T. For ground truth creation by means of SED, the optimal alignment $(I_1, T_1, L_1), \ldots, (I_N, T_N, L_N)$ includes words from the text edition even if $T \neq L$. Therefore, almost all words from the text edition appear in the ground truth, except some additional words in the text edition. This is

due to the more difficult goal pursued by the proposed alignment method, which includes an automatic spelling correction.

The result of the proposed alignment system is also given by a sequence of triplets $(I_1^*, T_1^*, L_1^*), \ldots, (I_n^*, T_n^*, L_n^*)$, which is compared with the ground truth by means of SED. For evaluating accuracy, precision, and recall, both T and L can be taken into account for substitution errors. If T is used, we will refer to the "spelling correctness" of the result and if L is used, we will refer to the "label correctness". The latter is interesting in the context of template extraction, e.g., for template-based keyword spotting, where pairs (I, L) are sufficient even if the spelling remains unknown. For character-based recognition, however, pairs (I, T) are needed for training handwriting recognition systems and therefore a high spelling correctness is important.

Systems Several system variants are compared that differ in the construction of the page HMM (see Section 10.3.2). The reference system REF takes only the lower case letters of each word label into account, then capitalization CAP, known spelling variants SPL, and the possibility to delete words SYS are added gradually. The reference system REF corresponds to forced alignment (see Section 5.2.3), which was proposed, e.g., in [258], for aligning perfect transcriptions. The final system SYS is given by the page HMM shown in Figure 10.5.

Parameters The validation set is used to optimize several system parameters. Since the proposed alignment system is not dependent on additional parameters, only the standard parameters of the HMM architecture have to be taken into account, i.e., the number of states S and the number of Gaussian mixtures G of the character HMM. Due to the high computational speed of the alignment system, a comprehensive grid search was realized. We have tested $S \in \{1, 5, \ldots, 20\}$ and $G \in \{1, 3, \ldots, 19\}$.

10.3.4 Results

The label alignment accuracy achieved on the test set is shown in Figure 10.6 for the four system variants and different numbers of training pages. While capitalization does not improve the result, adding known spelling variants and the option for deletion both improve the accuracy significantly. In particular, the improvements of the final system SYS over the forced alignment reference system REF are statistically significant for T-1 and T-20 (t-test, $\alpha = 0.05$). The optimum number of HMM states and Gaussian mixtures found on the validation set was $(S = 5, G = 1)$ for T-1 and $(S = 15, G = 13)$ for T-20. Not surprisingly, weak character models are preferable for small training sets, and stronger models for larger sets.

Depending on the number of training pages, the label alignment accuracy of SYS-1 and SYS-20 are listed in Table 10.3 as well as the recall and

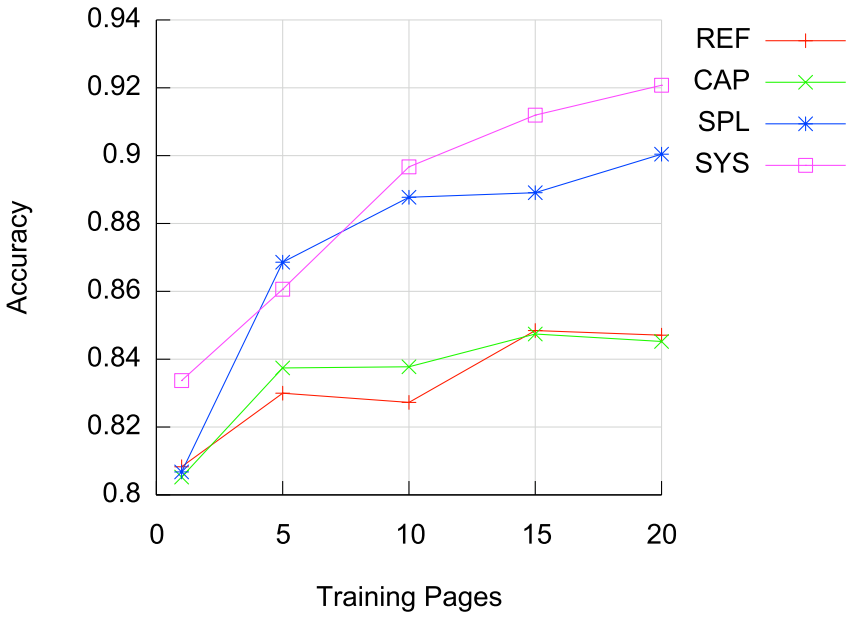


Figure 10.6: System Variants

Table 10.3: Test results.

System	Acc	R_L	P_L	R_S	P_S
SYS-1	83.37	98.35	84.61	97.88	65.51
SYS-20	92.07	96.40	95.35	95.97	84.81

precision with respect to label correctness (R_L, P_L) and spelling correctness (R_S, P_S). Surprisingly, a high label alignment accuracy of 83.37 is achieved with the proposed system even if only the first page of the manuscript is used for training the character models. 98.35 percent of the optimal alignments are retrieved by this system and 84.61 percent of the returned alignments are correct in terms of word image segmentation and word label. When the correct spelling of the words is considered, the system's precision drops to 65.51. This is not surprising considering the fact that only 54.2 percent of the letters from the test set are known (see Table 10.2).

The results of SYS-1 can be improved significantly if 20 pages are used for training. With respect to label correctness, an alignment accuracy of 92.07 is reported. Because more words are deleted during alignment, the recall of 96.40 is smaller than with SYS-1, yet a significant increase in precision up

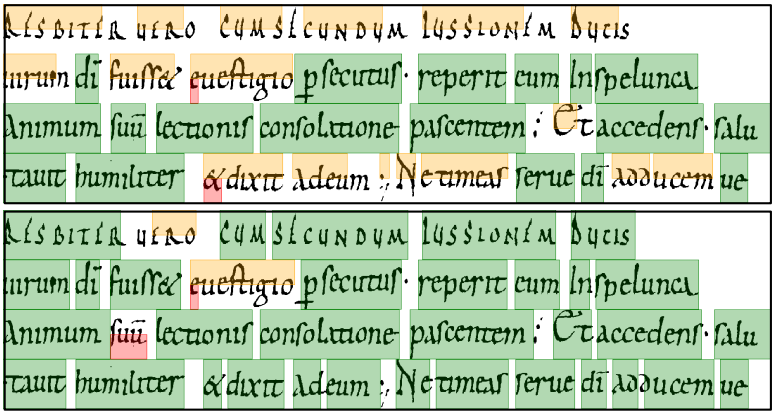


Figure 10.7: Alignment Example

to 95.35 is achieved. Since almost all letters from the test set are known as well as a considerable amount of special spellings, a high precision of 84.81 is achieved with respect to spelling correctness.

In Figure 10.7, some alignment results are illustrated for SYS-1 (top) and SYS-20 (bottom). Word image segmentation errors are displayed in the upper half of the text line, deletions in the lower half. While word breaks at the line end are handled very accurately throughout the test data, two error cases are predominant. First, since only few samples are available for upper case letters, headings written in upper case pose severe difficulties. Also in case of unknown special spellings, e.g., *suu* for the word *suum* in Figure 10.7, errors are frequent.

10.3.5 Discussion

On the Saint Gall database it was demonstrated that the proposed alignment system has a great potential to use inaccurate, electronic text editions for extracting training samples automatically. Even if only one manuscript page is used to train the system, about 83% of the optimal alignments (I, L) of the test set are returned correctly. This is surprising since only about half of the characters from the test set appear on the single manuscript page used for training. For 20 training pages, the accuracy is improved to about 92%. Correctness includes an accurate segmentation of text lines into word images I as well as an accurate assignment of word labels L from the text edition. With respect to the correct spelling (I, T), the precision of the system is about 66% for 1 training page and 85% for 20 training pages. The automatic extraction of training samples is magnitudes faster than automatic transcription with large vocabularies underlying natural language.

In contrast to the more general alignment system presented in Sec-

tion 10.2, the Patrologia Latina text edition differs from the manuscript image mostly in terms of spelling deviations. For this special case, the presented alignment system operates similar to forced alignment (see Section 5.2.3), which was proposed for aligning perfect transcriptions, e.g., in [258] in the context of word image extraction for ground truth creation. In both accurate and inaccurate scenarios, the problem of alignment can be formulated as a text line segmentation into a sequence of words that is provided by the electronic text edition. In contrast to forced alignment, spelling variations are incorporated in the proposed system. Also, the deletion of additional words in the text edition are allowed and instead of text line alignment, complete pages are aligned in order to cope with word breaks at the line ends. Experimental results indicate that the proposed adaptations significantly outperform standard forced alignment. In addition, the presented system allows for correcting spelling deviations, which is important to extract accurate training samples for character-based handwriting recognition systems.

Note that no spelling rules specific to Carolingian scripts are used in the proposed system. The system is designed for the general purpose of aligning manuscripts written in any alphabetic language. However, specific rules can be added in a straight-forward manner to the spelling variants of the individual words. Such additional rules are expected to further increase the accuracy of the system when constrained to Carolingian scrips.

10.4 Summary and Outlook

In this chapter, the alignment of electronic text editions with historical manuscript images is investigated. Taking into account readily available text editions created by human experts in the past, the goal is to extract training samples for handwriting recognition systems fully automatically. This procedure has the advantage that no time-consuming and expensive human interaction is needed for providing training samples, which can be regarded as one of the major obstacles for mass digitization of historical manuscripts nowadays.

The challenge faced for alignment is that electronic text editions often do not correspond exactly with the handwriting image. Instead of transcribing each word and each character, human experts may write out abbreviations, correct mistakes, insert punctuation marks, change the capitalization, or even rephrase whole sentences for better readability of the text edition. All deviations from the actual word spelling in such *inaccurate transcriptions* pose challenges for extracting correct training samples.

While a number of reports in the literature cover the case of aligning *accurate transcriptions* in the context of ground truth creation, we believe that we are the first to investigate the scenario of inaccurate transcriptions.

In Section 10.1, we formalize the goal and evaluation measure used for this task and state the conditions taken into account in this thesis. Two conditions are important with respect to the reported experimental results. First, in order to focus on handwriting recognition, text line extraction errors are not taken into account. Secondly, the text editions are assumed to be aligned at page-level. For real-world application, text line extraction errors would imply additional alignment errors and an additional extraction of alignment anchors may be necessary if the text edition is not aligned with the manuscript image at all.

Two alignment systems were developed that target different error conditions of the texts. In Section 10.2, a general alignment system is proposed that is able to cope with word substitution, deletion, and insertion errors in the text edition. The HMM-based system requires an initial set of training samples, given by text line images and their transcription, and performs text recognition, string alignment and keyword spotting in three consecutive passes. On an artificially distorted transcription of the Parzival database, a promising alignment accuracy of about 94% is reported if an error is generated for each second word of the transcription. In case of only 10% erroneous words, an alignment accuracy of about 97% is achieved.

The second system proposed in Section 10.3 concentrates on word substitution and insertion errors. In this scenario it is assumed that the human expert may have changed the word spelling, e.g., when writing out abbreviations, and may have added additional words, e.g., when adding remarks. Otherwise, the text edition is assumed to correspond closely to the manuscript image. An HMM-based system was devised for this scenario that is closely related to forced alignment but takes different word spellings into account. It was tested on the Saint Gall database and its Patrologia Latina text edition. An alignment accuracy of about 83% (92%) is reported for 1 (20) training pages with respect to word image extraction. The spelling precision of the automatically extracted training samples, given by word images together with their transcription, is about 66% (85%).

Currently, we cannot estimate how well the achieved alignment accuracies and spelling precisions perform for training handwriting recognition systems. Therefore, an important open question for future work is how well the automatically generated training samples perform in a semi-supervised scenario. In such a scenario, a few manuscript pages are used to initially train the alignment system in a supervised fashion. Then, training samples are generated automatically in an unsupervised step. Finally, they are used for training a handwriting recognition system, whose automatic transcription performance is measured in the end. Several semi-supervised iterations are possible, e.g., by retraining the alignment system itself in order to extract more reliable training samples in a further iteration. The goal would be to achieve high automatic transcription accuracy for small initial training sets, which are used for the first supervised step.

Chapter 11

Conclusions

This thesis investigates pattern recognition methods for handwriting recognition in historical documents. We have employed two state-of-the-art strategies to model and recognize characters, words, and sentences. First, a generative strategy using hidden Markov models (HMM) and secondly, a discriminative strategy using a special form of recurrent neural networks (NN). Both learning-based systems are generic in the sense that they can learn character models for arbitrary alphabetical languages as long as a number of training samples are provided. Furthermore, the recognizers are segmentation-free in the sense that no segmentation of the text line images into words and characters is required prior to training and recognition. This is appealing since recognition-free segmentation is prone to errors for touching characters, broken characters, variable word spacing, and difficult image conditions stemming, e.g., from damaged parchment, faded ink, and ink bleed-through.

To evaluate the performance of the recognizers, a data set of several historical scripts and languages was created. This was necessary because almost no publicly available benchmark data sets were available. The IAM historical document database (IAM-HistDB) includes Latin texts from the 9th century written in Carolingian minuscules (Saint Gall database), medieval German texts from the 13th century written in Gothic minuscules (Parzival database), and longhand English texts from the 18th century (George Washington database). Several hundred manuscript pages were segmented into text lines and words and were annotated with their transcriptions for training and evaluating handwriting recognition systems. Altogether, over hundred annotated manuscript pages were used for experimental evaluations in this thesis and were made publicly available on the Internet.

Several subproblems of handwriting recognition in historical documents have been investigated in this thesis. In the following, we draw separate conclusions for ground truth creation, automatic transcription, keyword spotting, and transcription alignment and provide a general outlook to future research at the end of this chapter.

Ground Truth Creation The generation of training samples for learning-based handwriting recognition is challenging for historical documents because actual manuscripts have to be taken into account rather than special acquisition forms used for modern handwritings. This thesis contributes a semi-automatic ground truth creation procedure that allows to annotate manuscript pages with text line locations and transcriptions in an interactive manner. Starting from page images and page transcriptions, a set of automatic methods and graphical user interfaces for manual correction is employed to generate text line and word images labeled with their transcription. On average, we report a manual interaction time of 5.5 minutes per text column for the IAM-HistDB. This bottleneck in terms of time expense consists of manual text area selection, correction of text line segmentation, and correction of transcription alignment.

Future work includes the integration of semi-automatic layout analysis, inaccurate transcription alignment, and handwriting recognition for supporting manual transcription if no text edition is available at all. A particular demand that was raised by the users of our ground truth creation procedure was that the automatic methods should learn from the corrections made by the human users in order to avoid the same mistake after few manual corrections.

Automatic Transcription The automatic transcription of historical documents has shown promising results. Based on a certain amount of training samples from single manuscripts, the neat and clean handwriting style encountered in the IAM-HistDB leads to relatively high recognition accuracies despite difficult image conditions. The best reported error rates are 4.0% for the Saint Gall database, 6.7% for the Parzival database, and 18.1% for the George Washington database. They are achieved with NN-based recognition, which outperforms HMM-based recognition significantly for all data sets when using the same text line normalization and geometric features. The use of unigram and bigram language models estimated on the training samples has improved complete text line recognition significantly. Despite the additional challenge of word segmentation during recognition, the achieved error rates are comparable with single word recognition. In fact, the best results for the Parzival and George Washington database were achieved with text line recognition in combination with word bigrams.

In real-world applications, an increase of the error rate has to be expected if automatic layout analysis and text line extraction is erroneous. Nevertheless, the recognition accuracy is remarkably high for the Carolingian script of the Saint Gall database and the Gothic script of the Parzival database compared with benchmark results for unconstrained modern handwritings. Although far from being perfect the performance is encouraging for content-based indexing of historical documents in digital libraries.

An important future line of research is language modeling for historical documents. In this thesis, we have considered word lexicons and statistical language models that are built from the training samples of a given historical manuscript. For large manuscript collections that share the same language it would be desirable to create comprehensive electronic text corpora for lexicon extraction and language modeling. A fundamental requirement for such text corpora would be a common encoding of special characters, preferably with Unicode. Also, providing spelling variants would be helpful in order to cope with non-standardized orthography. With respect to language models that incorporate more than n-gram statistics, any semantic labels attached to the words would be valuable in the future.

A missing piece of information that would be of great value is a listing of the most important scripts and languages that need to be processed worldwide for integrating historical manuscript in digital libraries. Such an overview would help to guide research towards significant manuscript clusters, especially with respect to research on recognizer adaptation across similar scripts and languages.

General Recognition Improvements In the context of automatic transcription, this thesis contributes several general improvements of the recognizers. First, the geometric handwriting features used throughout this thesis could be improved significantly by means of unsupervised feature selection. Besides linear principal component analysis (PCA) and independent component analysis (ICA), we have investigated the application of non-linear kernel PCA with a Gaussian kernel. For HMM-based recognition of historical as well as modern handwritings, feature selection could significantly outperform the baseline features. In particular, kernel PCA outperformed the linear methods. Future work includes the investigation of other features, recognizers, and kernels as well as supervised feature selection methods, e.g., based on character labels that are assigned by forced alignment.

Secondly, a novel algorithm for NN-based text line recognition is introduced. Based on ϵ-compression, prefix graphs, and token pruning, the proposed algorithm achieves a remarkable speed-up compared to the baseline algorithm. For a 20000 word lexicon, the algorithm was over 600 times faster than the baseline on the IAMDB. Only 5000 tokens were sufficient to achieve an optimal accuracy in the compact search space of BLSTM networks with CTC. Apart from the speedup, the algorithm returns a more versatile output in form of word lattices instead of only the best word sequence. Future work includes an assessment of the lattice quality, for instance in the context of classifier combination.

The token passing approximation has shown an impressive performance and the need for further studies is evident. It seems that the compactness of the search space for BLSTM and CTC allows for high-speed text recogni-

tion. In many scenarios, it would be of great interest to use suboptimal but fast algorithms to access the textual content of large document collections, most notably for indexing large historical archives and for semi-supervised learning.

Graph-Based Handwriting Recognition In the general context of statistical handwriting recognition, a novel feature set based on handwriting graphs is introduced. The proposed graph similarity features (GSF) embed handwriting graphs in a vectorial dissimilarity space with respect to character prototypes. Based on graph edit distance, arbitrary handwriting graphs can be taken into account. Prototypes are selected automatically from the training set and provide structural information for handwriting recognition. This approach is especially suited for the recognition of single historical manuscripts, where a small set of character prototypes is expected to cover the most important structures. For HMM-based recognition of historical handwritings, a significant increase in accuracy is reported for the structural GSF descriptor compared with two well-known statistical descriptors. A known disadvantage of the proposed structural descriptor, however, is its computational overhead compared with the statistical descriptors.

As a future line of research, the proposed graph similarity features open a door to handwriting recognition based on structural representation. By means of dissimilarity space embedding, a bridge between the high representational power of graphs and the large repository of algorithmic tools in statistical pattern recognition could be established. In this thesis, we have investigated a single graph model based on handwriting skeletons. Any other graph model could be used within the proposed framework. Furthermore, alternative dissimilarities could be investigated that are not necessarily based on graphs. Finally, alternative prototypes could be investigated that are not necessarily required to be characters. Learning-based prototype models would offer an intriguing alternative to the prototypes investigated so far.

Keyword Spotting Keyword spotting is a promising alternative to automatic transcription for the task of identifying search terms in historical documents, especially if no reliable lexicon and language model are available. Keyword spotting is also appealing for indexing large manuscript collections without the computational overhead of lexicon-based transcription. In this thesis, we contribute two novel algorithms for keyword spotting based on character models, one using HMM and the other using NN.

Compared with previous word spotting methods, the proposed sub-word model approach has several advantages. First, no segmentation of text lines into words is required which can be prone to errors. Compared with word models, the character models have the advantage that they can be trained across different words and can be used to spot arbitrary keywords that

are not required to be known at training stage. Compared with lexicon-based transcription, keyword spotting is magnitudes faster. In this context, we can provide a realistic time estimate with respect to a current personal computer disregarding all processing steps that are independent of a lexicon. With the proposed methods we could spot a search term in 140,000 pages of the George Washington collection within an hour while an automatic transcription would take several years.

A known disadvantage of the proposed approach is, however, that it requires transcribed text line images for training which implies considerable human effort for historical documents. Furthermore, template-based image matching might be the only option available if neither language nor alphabet are known for a historical document. Finally, lexicon-based transcription is expected to outperform character-based spotting in terms of accuracy if reliable lexicons and language models are available.

For keyword spotting in historical as well as modern handwritings we demonstrate that the proposed learning-based systems significantly outperform a well-established template-matching approach. A mean average precision of 95% is reported for the Parzival database and 91.5% for the George Washington database, which is the highest result ever reported for this data set. On modern documents, the generalization capability of the keyword spotting systems to unseen writing styles is demonstrated, which may also become important for historical documents when the spotting systems are used across different manuscripts. As for automatic transcription, NN-based recognition significantly outperformed HMM-based recognition when using the same text line normalization and geometric features.

A promising line of future research is given by investigating alternative confidence models which are needed to decide whether or not a keyword is present in a text line. So far, we employ log-odds to general filler models for HMM-based spotting and character posteriors for NN-based spotting. Another issue for future work regards the nature of the search terms. In this thesis, we report spotting results for individual words. It would be interesting to investigate more complex queries such as regular expressions.

Transcription Alignment The alignment of existing electronic text editions with manuscript images has a great potential for handwriting recognition in historical documents which has, in our opinion, not been fully explored so far. By taking manuscripts into account for which human experts have already provided a text edition, the human effort for integrating handwriting recognition into the process of mass digitization can be reduced significantly. The extraction of training samples is reduced to the problem of image segmentation and text alignment, which can be performed efficiently by laypersons as demonstrated for the IAM-HistDB. Although electronic text editions are clearly not available for every historical script and

language, it can be argued that the availability of text editions is directly correlated with the importance human experts attested to the respective cultural heritage.

In this thesis, we go a step further than the alignment of correct transcriptions with handwriting images in the context of ground truth creation. Instead, we take inaccurate texts into account, which do not correspond exactly with the transcription of the manuscript images, and consider the possibility to extract training samples fully automatically. Even if the training samples contain errors we believe that an initialization of automatic recognition systems could be feasible in this way without human supervision. By considering inaccurate text editions, we enlarge the scope of admissible texts since human experts tend to write out abbreviations, correct mistakes, insert punctuation marks, change the capitalization, or even rephrase whole sentences for better readability. To our knowledge, we are the first to investigate the problem of aligning inaccurate transcriptions.

Two alignment systems are proposed that target different error conditions of the texts. The first system takes a general set of word substitution, insertion, and deletion errors into account. An HMM-based system was devised for this scenario that performs text recognition, string alignment, and keyword spotting in three consecutive passes. On an artificially distorted transcription of the Parzival database, a promising alignment accuracy of about 94% is reported if an error is generated for each second word of the transcription. In case of only 10% erroneous words, an alignment accuracy of about 97% is achieved.

The second system concentrates on word substitution and insertion errors. In this scenario it is assumed that the human expert may have changed the word spelling, e.g., when writing out abbreviations, and may have added additional words, e.g., when adding remarks. Otherwise, the text edition is assumed to correspond closely to the manuscript image. An HMM-based system was devised for this scenario that is closely related to forced alignment but takes different word spellings into account. It was tested on the Saint Gall database and its Patrologia Latina text edition. An alignment accuracy of about 83% (92%) is reported for 1 (20) training pages with respect to word image extraction. The spelling precision of the automatically extracted training samples, given by word images together with their transcription, is about 66% (85%).

Currently, we cannot estimate how well the achieved alignment accuracies and spelling precisions perform for training handwriting recognition systems. Therefore, an important open question for future work is how well the automatically generated training samples perform in a semi-supervised scenario. In such a scenario, a few manuscript pages are used to initially train the alignment system in a supervised fashion. Then, training samples are generated automatically in an unsupervised step. Finally, they are used for training a handwriting recognition system, whose automatic transcrip-

tion performance is measured in the end. Several semi-supervised iterations are possible, e.g., by retraining the alignment system itself in order to extract more reliable training samples in a further iteration. The goal would be to achieve high automatic transcription accuracy for small initial training sets, which are used for the first supervised step. We believe that this semi-supervised alignment scenario has a great potential to initialize handwriting recognition systems at low cost for mass digitization of historical documents.

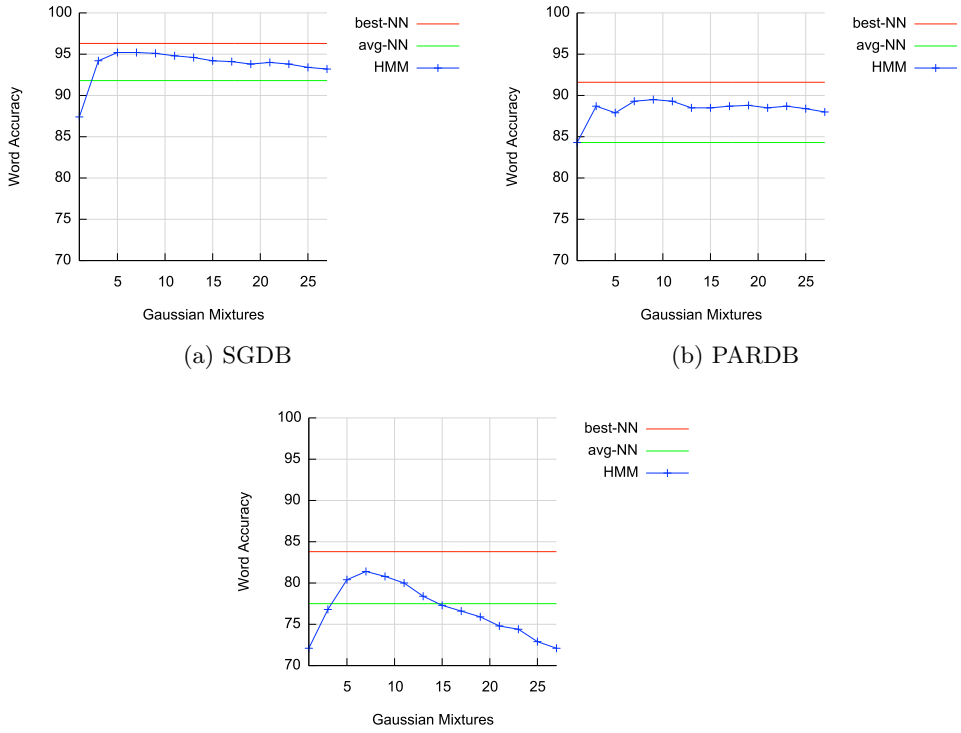
Outlook The extraction of textual content from historical documents can be seen as one of the hardest problems in handwriting recognition. Facing a large variety of historical scripts and languages, difficult image conditions, and a small amount of training samples, automatic recognition is bound to commit errors. Current research has developed several strategies to cope with this challenging situation. To conclude this thesis, we want to point out two trends that are promising for future research.

First, the results that are required from pattern recognition systems are shifting away from the single best solution that minimizes the error rate. Instead, the recognition confidence becomes more important in a scenario where errors are expected since they cannot be completely avoided. Confidence models are the core of many applications in historical document analysis, most notably keyword spotting and semi-supervised learning. The development of reliable confidence models is expected to play a central role in future research.

Secondly, the aim of pattern recognition is shifting away from the 100% accuracy with fully automatic systems. Instead, the support of human users with semi-automatic systems becomes more important. This can be observed, in particular, for interactive systems for ground truth creation. The development of intelligent human machine interfaces is an exciting challenge for future research. In particular, various forms of human feedback need to be integrated in the machine learning process.

Appendix A

Validation Results

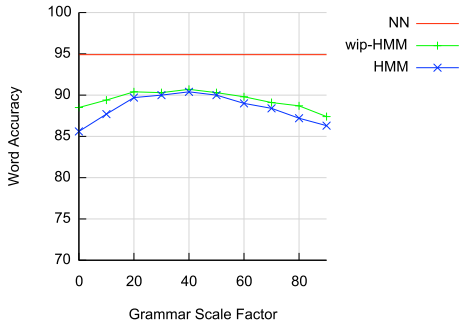


(a) SGDB

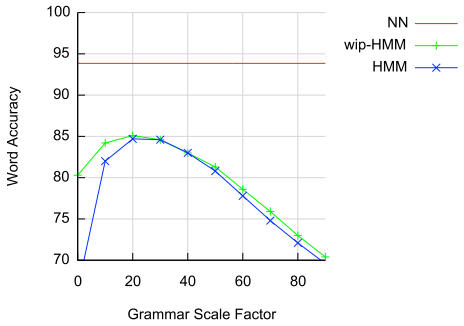
(b) PARDB

(c) GWDB

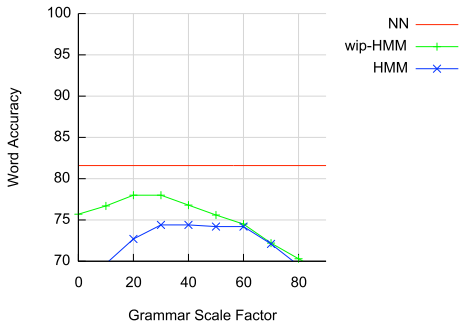
Figure A.1: Single word recognition. Word accuracy of the best neural network, the average neural network, and hidden Markov models with different numbers of Gaussian mixtures.



(a) SGDB

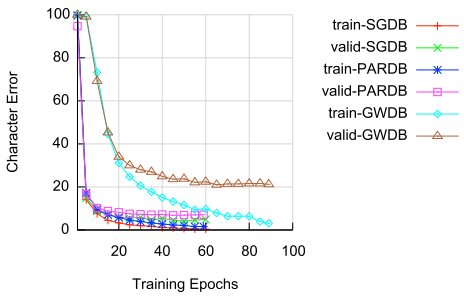


(b) PARDB

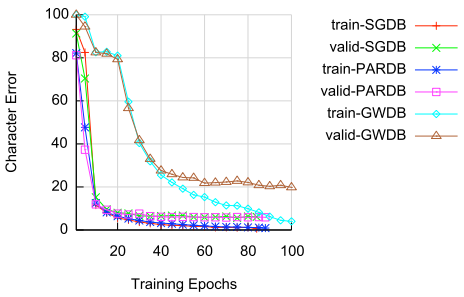


(c) GWDB

Figure A.2: Text line recognition. Word accuracy of the best neural network and hidden Markov model with bigram language models. Optimization of the word insertion penalty (wip) in addition to the grammar scale factor.



(a) Single Word Recognition



(b) Text Line Recognition

Figure A.3: Neural network training. Character error rate of the best neural network on the training and validation set.

Bibliography

[1] I. S. I. Abuhaiba, M. J. J. Holt, and S. Datta. Recognition of off-line cursive handwriting. *Computer Vision and Image Understanding*, 71(1):19–38, 1998.

[2] I. S. I. Abuhaiba, S. A. Mahmoud, and R. J. Green. Recognition of handwritten cursive Arabic characters. *IEEE Trans. PAMI*, 16(6):664–672, 1994.

[3] T. Adamek, N. E. Connor, and A. F. Smeaton. Word matching using single closed contours for indexing historical documents. *Int. Journal on Document Analysis and Recognition*, 9(2):153–165, 2007.

[4] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison Wesley, 1974.

[5] E. Alpaydin and F. Alimoglu. *Pen-Based Recognition of Handwritten Digits*. Dept. of Computer Engineering, Bogazici University, 1998.

[6] A. Antonacopoulos and A. Downton. Special issue on the analysis of historical documents. *Int. Journal on Document Analysis and Recognition*, 9(2):75–77, 2007.

[7] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

[8] F. R. Bach and M. I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[9] M. Baechler and R. Ingold. Medieval manuscript layout model. In *Proc. 10th ACM Symposium on Document Engineering*, pages 275–278, 2010.

[10] L. Bahl, S. De Gennaro, P. Gopalakrishnan, and R. Mercer. A fast approximate acoustic match for large vocabulary speech recognition. *IEEE Trans. on Speech and Audio Processing*, 1(1):59–67, 1993.

[11] C. Barrett, R. Hughey, and K. Karplus. Scoring hidden Markov models. *Computer Applications in the Biosciences*, 13:191–199, 1997.

[12] L. Baum and J. Egon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of American Mathematical Society*, 73:360–363, 1967.

[13] I. Bazzi, R. Schwartz, and J. Makhoul. An omnifont open-vocabulary OCR system for English and Arabic. *IEEE Trans. PAMI*, 21(6):495–504, 1999.

[14] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[15] R. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[16] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. PAMI*, 24(4):509–522, 2002.

[17] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe. Global optimization of a neural network-hidden Markov model hybrid. *IEEE Trans. on Neural Networks*, 3(2):252–259, 1992.

[18] R. Bertolami and H. Bunke. Hidden Markov model based ensemble methods for offline handwritten text line recognition. *Pattern Recognition*, 41(11):3452–3460, 2008.

[19] A. Bhardwaj, D. Jose, and V. Govindaraju. Script independent word spotting in multilingual documents. In *Proc. 2nd Int. Workshop on Cross Lingual Information Access*, pages 48–54, 2008.

[20] U. Bhattacharya and B. Chaudhuri. Handwritten numeral databases of Indian scripts and multistage recognition of mixed numerals. *IEEE Trans. PAMI*, 31(3):444–457, 2009.

[21] R. Bippus. 1-dimensional and pseudo 2-dimensional HMMs for the recognition of German literal amounts. In *Proc. 4th Int. Conf. on Document Analysis and Recognition*, pages 487–490, 1997.

[22] G. Boccignone, A. Chianese, L. P. Cordella, and A. Marcelli. Recovering dynamic information from static handwriting. *Pattern Recognition*, 26(3):409–418, 1993.

[23] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[24] F. L. Bourgeois and H. Emptoz. DEBORA: Digital AccEss to BOoks of the RenAissance. *Int. Journal on Document Analysis and Recognition*, 9:193–221, 2007.

[25] H. Bourlard and C. J. Wellekens. Links between Markov models and multilayer perceptrons. *IEEE Trans. PAMI*, 12(12):1167–1178, 1990.

[26] R. M. Bozinovic and S. N. Srihari. Off-line cursive script word recognition. *IEEE Trans. PAMI*, 11(1):68–83, 1989.

[27] A. Brakensiek and G. Rigoll. Handwritten address recognition using hidden Markov models. In A. Dengel, M. Junker, and A. Weisbecker, editors, *Reading and Learning*, pages 103–122. Springer, 2004.

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

[29] H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983.

[30] H. Bunke, R. Ammann, G. Kaufmann, T. Ha, M. Schenkel, R. Seiler, and F. Eggimann. Recovery of temporal information of cursively handwritten words for on-line recognition. In *Proc. 4th Int. Conf. on Document Analysis and Recognition*, volume 2, pages 931–935, 1997.

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

[32] H. Bunke and T. Varga. Off-line Roman cursive handwriting recognition. In B. Chaudhuri, editor, *Digital Document Processing*, pages 165–173. Springer, 2007.

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

[34] J. Cai and Z.-Q. Liu. Markov random field models for handwritten word recognition. In *Proc. Int. Conf. on Intelligent Processing Systems*, pages 1400–1404, 1997.

[35] H. Cao and V. Govindaraju. Template-free word spotting in low-quality manuscripts. In *Proc. 6th Int. Conf. on Advances in Pattern Recognition*, pages 135–139, 2007.

[36] R. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18:690–706, 1996.

[37] V. Chakravarthy and B. Kompella. The shape of handwritten characters. *Pattern Recognition Letters*, 24(12):1901–1913, 2003.

[38] R. Chamchong and C. C. Fung. Character segmentation from ancient palm leaf manuscripts in thailand. In *Proc. 1st Int. Workshop on Historical Document Imaging and Processing*, pages 140–145, 2011.

[39] J. Chan, C. Ziftci, and D. Forsyth. Searching off-line Arabic documents. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, pages 1455–1462, 2006.

[40] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for histogram-based image classification. *IEEE Trans. on Neural Networks*, 10(5):1055–1064, 1999.

[41] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006.

[42] F. R. Chen, L. D. Wilcox, and D. S. Bloomberg. Word spotting in scanned images using hidden Markov models. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 5, pages 1–4, 1993.

[43] J. Chen, H. Cao, R. Prasad, A. Bhardwaj, and P. Natarajan. Gabor features for offline Arabic handwriting recognition. In *Proc. 8th Int. Workshop on Document Analysis Systems*, pages 53–58, 2010.

[44] C. E. Cherry. Some experiments on the recognition of speech, with one and with two ears. *Journal of the Acoustical Society of America*, 25(5):975–979, 1953.

[45] C. Choisy. Dynamic handwritten keyword spotting based on the nshp-hmm. In *Proc. 9th Int. Conf. on Document Analysis and Recognition*, pages 242–246, 2007.

[46] C. Chow. On optimum recognition error and reject tradeoff. *IEEE Trans. on Information Theory*, 16(1):41–46, 1970.

[47] D. Ciresan, U. Meier, L. Gambardella, and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. In *Proc. 11th Int. Conf. on Document Analysis and Recognition*, pages 1135–1139, 2011.

[48] P. Comon. Independent component analysis, A new concept? *Signal Processing*, 36(3):287–314, 1994.

[49] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[50] B. Coüasnon, J. Camillerapp, and I. Leplumey. Access by content to handwritten archive documents: Generic document recognition method and platform for annotations. *Int. Journal on Document Analysis and Recognition*, 9(2):223–242, 2007.

[51] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. on Electronic Computers*, 14:326–334, 1965.

[52] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. on Information Theory*, 13(1):21–27, 1967.

[53] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[54] A. Cross, R. Wilson, and E. Hancock. Inexact graph matching using genetic search. *Pattern Recognition*, 30(6):953–970, 1997.

[55] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 39(1):1–38, 1977.

[56] P. Dickinson, H. Bunke, A. Dadej, and M. Kraetzl. Matching graphs with unique node labels. *Pattern Analysis and Applications*, 7(3):243–254, 2004.

[57] T. M. T. Do and T. Artieres. Maximum margin training of Gaussian HMMs for handwriting recognition. In *Proc. 10th Int. Conf. on Document Analysis and Recognition*, pages 976–980, 2009.

[58] D. Doermann and A. Rosenfeld. Recovery of temporal information from static images of handwriting. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, pages 162–168, 1992.

[59] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

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

[61] J. Edwards, Y. W. Teh, D. Forsyth, R. Bock, M. Maire, and G. Vesom. Making Latin manuscripts searchable using gHMM's. In *Advances in Neural Information Processing Systems*, volume 17, pages 385–392, 2005.

[62] M. Ekinci and M. Aykut. Palmprint recognition by applying wavlet-based kernel PCA. *Journal of Computer Science and Technology*, 23(5):851–861, 2008.

[63] R. El-Hajj, L. Likforman-Sulem, and C. Mokbel. Arabic handwriting recognition using baseline dependant features and hidden Markov modeling. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, pages 893–897, 2005.

[64] A. El Yacoubi, M. Gilloux, and J.-M. Bertille. A statistical approach for phrase location and recognition within a text line: An application to street name recognition. *IEEE Trans. PAMI*, 24(2):172–188, 2002.

[65] S. Espana-Boquera, M. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid HMM/ANN models. *IEEE Trans. PAMI*, 33(4):767–779, 2011.

[66] J. T. Favata, G. Srikantan, and S. N. Srihari. Handprinted character/digit recognition using a multiple feature/resolution philosophy. In *Proc. 4th Int. Workshop on Frontiers in Handwriting Recognition*, pages 57–66, 1994.

[67] S. Feng, N. Howe, and R. Manmatha. A hidden Markov model for alphabet-soup word recognition. In *Proc. 11th Int. Conf. on Frontiers in Handwriting Recognition*, pages 210–215, 2008.

[68] S. Feng and R. Manmatha. A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books. In *Proc. 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 109–118, 2006.

[69] S. Feng, R. Manmatha, and A. McCallum. Exploring the use of conditional random field models and HMMs for historical handwritten document recognition. In *Proc. 2nd Int. Conf. on Document Image Analysis for Libraries*, pages 30–37, 2006.

[70] A. Fischer and H. Bunke. Kernel PCA for HMM-based cursive handwriting recognition. In *Proc. 13th Int. Conf. on Computer Analysis of Images and Patterns*, pages 181–188, 2009.

[71] A. Fischer and H. Bunke. Character prototype selection for handwriting recognition in historical documents with graph similarity features. In *Proc. 19th European Signal Processing Conference*, pages 1435–1439, 2011.

[72] A. Fischer, V. Frinken, A. Fornés, and H. Bunke. Transcription alignment of latin manuscripts using hidden Markov models. In *Proc. 1st Int. Workshop on Historical Document Imaging and Processing*, pages 29–36, 2011.

[73] A. Fischer, E. Indermühle, H. Bunke, G. Viehhauser, and M. Stolz. Ground truth creation for handwriting recognition in historical documents. In *Proc. 9th Int. Workshop on Document Analysis Systems*, pages 3–10, 2010.

[74] A. Fischer, E. Indermühle, V. Frinken, and H. Bunke. HMM-based alignment of inaccurate transcriptions for historical documents. In *Proc. 11th Int. Conf. on Document Analysis and Recognition*, pages 53–57, 2011.

[75] A. Fischer, A. Keller, V. Frinken, and H. Bunke. HMM-based word spotting in handwritten documents using subword models. In *Proc. 20th Int. Conf. on Pattern Recognition*, pages 3416–3419, 2010.

[76] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, http://dx.doi.org/10.1016/j.patrec.2011.09.009, accepted for publication, 2011.

[77] A. Fischer, K. Riesen, and H. Bunke. An experimental study of graph classification using prototype selection. In *Proc. 19th Int. Conf. on Pattern Recognition*, pages 1–4, 2008.

[78] A. Fischer, K. Riesen, and H. Bunke. Graph similarity features for HMM-based handwriting recognition in historical documents. In *Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition*, pages 253–258, 2010.

[79] A. Fischer, M. Wüthrich, M. Liwicki, V. Frinken, H. Bunke, G. Viehhauser, and M. Stolz. Automatic transcription of handwritten medieval documents. In *Proc. 15th Int. Conf. on Virtual Systems and Multimedia*, pages 137–142, 2009.

[80] A. Fornés, V. Frinken, A. Fischer, J. Almazán, G. Jackson, and H. Bunke. A keyword spotting approach using blurred shape model-based descriptors. In *Proc. 1st Int. Workshop on Historical Document Imaging and Processing*, pages 83–90, 2011.

[81] W. Francis and H. Kucera. *Brown Corpus Manual. Manual of Information to Accompany a Standard Corpus of Present-Day Edited American English, for Use with Digital Computers.* Brown University, 1979.

[82] V. Frinken. *Semi-Supervised Learning and Keyword Spotting for Handwriting Recognition.* PhD thesis, University of Bern, 2011.

[83] V. Frinken, A. Fischer, and H. Bunke. A novel word spotting algorithm using bidirectional long short-term memory neural networks. In *Proc. 4th Int. Workshop on Artificial Neural Networks in Pattern Recognition*, pages 185–196, 2010.

[84] V. Frinken, A. Fischer, and H. Bunke. Improving handwritten keyword spotting with self-training. In *Proc. 26th Symposium On Applied Computing*, pages 838–843, 2011.

[85] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. Adapting BLSTM neural network based keyword spotting trained on modern data to historical documents. In *Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition*, pages 352–257, 2010.

[86] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. A novel word spotting method based on recurrent neural networks. *IEEE Trans. PAMI*, 34(2):211–224, 2012.

[87] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Co., 1979.

[88] T. Gärtner. A survey of kernels for structured data. *SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[89] A. Garz, R. Sablatnig, and M. Diem. Layout analysis for historical manuscripts using SIFT features. In *Proc. 11th Int. Conf. on Document Analysis and Recognition*, pages 508–512, 2001.

[90] B. Gatos, N. Stamatopoulos, and G. Louloudis. ICFHR 2010 handwriting segmentation contest. In *Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition*, pages 737–742, 2010.

[91] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1991.

[92] A. Giménez, I. Khoury, and A. Juan. Windowed Bernoulli mixture HMMs for Arabic handwritten word recognition. In *Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition*, pages 533–538, 2010.

[93] A. Gordo, D. Llorens, A. Marzal, F. Prat, and J. Vilar. State: A multimodal assisted text-transcription system for ancient documents. In *Proc. 8th Int. Workshop on Document Analysis Systems*, pages 135–142, 2008.

[94] 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.

[95] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. PhD thesis, Technische Universität München, 2008.

[96] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for improved unconstrained handwriting recognition. *IEEE Trans. PAMI*, 31(5):855–868, 2009.

[97] Z. Guo and R. Hall. Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, 32(3):359–373, 1989.

[98] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. UNIPEN project of on-line data exchange and recognizer benchmarks. In *Proc. 12th Int. Conf. on Pattern Recognition*, volume 2, pages 29–33, 1994.

[99] R. Haeb-Umbach and H. Ney. Improvements in beam search for 10000-word continuous-speech recognition. *IEEE Trans. on Speech and Audio Processing*, 2(2):353–356, 1994.

[100] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems, Science, and Cybernetics*, 4(2):100–107, 1968.

[101] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines: Global versus component-based approach. In *Proc. 8th Int. Conf. on Computer Vision*, pages 688–694, 2001.

[102] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735—1780, 1997.

[103] A. Honkela, H. Valpola, A. Ilin, and J. Karhunen. Blind separation of nonlinear mixtures by variational Bayesian learning. *Digital Signal Processing*, 17(5):914–934, 2007.

[104] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[105] N. Howe, T. M. Rath, and R. Manmatha. Boosted decision trees for word recognition in handwritten document retrieval. In *Proc. 28th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 377–383, 2005.

[106] N. R. Howe, S. Feng, and R. Manmatha. Finding words in alphabet soup: Inference on freeform character recognition for historical scripts. *Pattern Recognition*, 42(12):3338–3347, 2009.

[107] C. Huang and S. N. Srihari. Mapping transcripts to handwritten text. In *Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition*, pages 15–20, 2006.

[108] P. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

[109] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. PAMI*, 16(5):550–554, 1994.

[110] D. P. Huttenlocher, G. A. Klanderman, G. A. Kl, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. PAMI*, 15:850–863, 1993.

[111] A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. on Neural Networks*, 10(3):626–634, 1999.

[112] A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[113] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley-Interscience, 2001.

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

[115] E. Indermühle, M. Liwicki, and H. Bunke. Combining alignment results for historical handwritten document analysis. In *Proc. 10th Int. Conf. on Document Analysis and Recognition*, pages 1186–1190, 2009.

[116] A. Jain, J. Mao, and K. Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[117] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[118] F. Jelinek, R. Mercer, and L. Bahl. Continuous speech recognition: Statistical methods. In P. Krishnaiah and L. Kanal, editors, *Classification Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics*, pages 549–573. Elsevier, 1982.

[119] X. Jiang and H. Bunke. Optimal quadratic-time isomorphism of ordered graphs. *Pattern Recognition*, 32(17):1273–1283, 1999.

[120] S. Johansson, G. 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, 1978.

[121] I. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[122] B. H. Juang, S. E. Levinson, and M. M. Sondhi. Maximum likelihood estimation for multivariate mixture observations of Markov chains. *IEEE Trans. on Information Theory*, 32(2):307–309, 1986.

[123] B. H. Juang and L. R. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[124] 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.

[125] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[126] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 181–184, 1995.

[127] A. L. Koerich. Rejection strategies for handwritten word recognition. In *Proc. 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 479–484, 2004.

[128] A. L. Koerich, Y. Leydier, R. Sabourin, and C. Y. Suen. A hybrid large vocabulary handwritten word recognition system using neural networks with hidden Markov models. In *Proc. Int. Workshop on Frontiers in Handwriting Recognition*, pages 99–104, 2002.

[129] A. L. Koerich, R. Sabourin, and C. Suen. Large vocabulary off-line handwriting recognition: A survey. *Pattern Analysis and Applications*, 6:97–121, 2003.

[130] A. Kolcz, J. Alspector, M. Augusteijn, R. Carlson, and G. Viorel Popescu. A line-oriented approach to word spotting in handwritten documents. *Pattern Analysis and Applications*, 3(2):153–168, 2000.

[131] E. M. Kornfield, R. Manmatha, and J. Allan. Further explorations in text alignment with handwritten documents. *Int. Journal on Document Analysis and Recognition*, 10(1):39–52, 2007.

[132] S.-S. Kuo and O. E. Agazzi. Keyword spotting in poorly printed documents using pseudo 2-d hidden markov models. *IEEE Trans. PAMI*, 16(8):842–848, 1994.

[133] L. Lam, S.-W. Lee, and C. Y. Suen. Thinning methodologies – A comprehensive survey. *IEEE Trans. PAMI*, 14(9):869–885, 1992.

[134] V. Lavrenko, T. M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proc. Int. Workshop on Document Image Analysis for Libraries*, pages 278–287, 2004.

[135] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[136] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[137] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[138] Y. Leydier, F. LeBourgeois, and H. Emptoz. Text search for medieval manuscript images. *Pattern Recognition*, 40:3552–3567, 2007.

[139] Y. Leydier, A. Ouji, F. LeBourgeois, and H. Emptoz. Towards an omnilingual word retrieval system for ancient manuscripts. *Pattern Recognition*, 42(9):2089–2105, 2009.

[140] L. Likforman-Sulem, A. Zahour, and B. Taconet. Text line segmentation of historical documents: A survey. *Int. Journal on Document Analysis and Recognition*, 9(2):123–138, 2007.

[141] Y.-H. Liu, Y.-T. Chen, and S.-S. Lu. Face detection using kernel PCA and imbalanced SVM. In *Proc. 2nd Int. Conf. on Advances in Natural Computation*, pages 351–360, 2006.

[142] M. Liwicki and H. Bunke. IAM-OnDB - an on-line English sentence database acquired from handwritten text on a whiteboard. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, volume 2, pages 956–961, 2005.

[143] M. Liwicki, E. Indermühle, and H. Bunke. Online handwritten text line detection using dynamic programming. In *Proc. 9th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 447–451, 2007.

[144] G. Lorette. Handwriting recognition or reading? What is the situation at the dawn of the 3rd millenium? *Int. Journal on Document Analysis and Recognition*, 2(1):2–12, 1999.

[145] L. M. Lorigo and V. Govindaraju. Offline Arabic handwriting recognition: A survey. *IEEE Trans. PAMI*, 28(5):712–724, 2006.

[146] L. M. Lorigo and V. Govindaraju. Transcript mapping for handwritten Arabic documents. In *Proc. on Document Recognition and Retrieval XIV*, volume 6500, pages 1–8. SPIE, 2007.

[147] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[148] S. Madhvanath and V. Govindaraju. The role of holistic paradigms in handwritten word recognition. *IEEE Trans. PAMI*, 23(2):149–164, 2001.

[149] R. Manmatha, C. Han, and E. Riseman. Word spotting: A new approach to indexing handwriting. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, pages 631—637, 1996.

[150] R. Manmatha and J. L. Rothfeder. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Trans. PAMI*, 27(8):1212–1225, 2005.

[151] H. Mara, J. Hering, and S. Kromker. GPU based optical character transcription for ancient inscription recognition. In *Proc. Int. Conf. on Virtual Systems and MultiMedia*, pages 154–159, 2009.

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

[153] 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.

[154] 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(1):39–46, 2002.

[155] A. M. Martínez and A. C. Kak. PCA versus LDA. *IEEE Trans. PAMI*, 23:228–233, 2001.

[156] S. Marukatat, T. Artieres, P. Gallinari, and B. Dorizzi. Rejection measures for handwriting sentence recognition. In *Proc. 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 24–29, 2002.

[157] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943.

[158] M. Mohamed and P. Gader. Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. *IEEE Trans. PAMI*, 18(5):548–554, 1996.

[159] K. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *Neural Networks*, 12(2):181–202, 2001.

[160] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[161] C. Myers, L. Rabiner, and A. Rosenberg. An investigation of the use of dynamic time warping for word spotting and connected speech recognition. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 173–177, 1980.

[162] G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Trans. PAMI*, 22(1):38–62, 2000.

[163] M. Nakagawa and K. Matsumoto. Collection of on-line handwritten Japanese character pattern databases and their analysis. *Int. Journal on Document Analysis and Recognition*, 7(1):69–81, 2004.

[164] M. Neuhaus and H. Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific, 2007.

[165] M. Neuhaus, K. Riesen, and H. Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In *Proc. 11th Int. Workshop on Structural and Syntactic Pattern Recognition*, pages 163–172, 2006.

[166] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231:289–337, 1933.

[167] K. Ntzios, B. Gatos, I. Pratikakis, T. Konidaris, and S. J. Perantonis. An old Greek handwritten ocr system based on an efficient segmentation-free approach. *Int. Journal on Document Analysis and Recognition*, 9(2):179–192, 2007.

[168] E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13:411–430, 2000.

[169] M. Pechwitz, S. Maddouri, V. Maergner, N. Ellouze, and H. Amiri. IFN/ENIT - database of handwritten Arabic words. In *Proc. 7th Colloque International Francophone sur l'Ecrit et le Document*, pages 129–136, 2002.

[170] E. Pekalska and R. Duin. *The Dissimilarity Representations for Pattern Recognition: Foundations and Applications*. World Scientific, 2005.

[171] D. Pérez, L. Tarazón, N. Serrano, F.-M. Castro, O. Ramos-Terrades, and A. Juan. The GERMANA database. In *Proc. 10th Int. Conf. on Document Analysis and Recognition*, pages 301–305, 2009.

[172] F. Perronnin and J. Rodriguez-Serrano. Fisher kernels for handwritten word-spotting. In *Proc. 10th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 106–110, 2009.

[173] D. T. Pham and P. Garat. Blind separation of mixture of independent sources through a maximum likelihood approach. In *Proc. European Signal Processing Conference*, pages 771–774, 1992.

[174] J. Pitrelli, J. Subrahmonia, and M. P. Perrone. Confidence modeling for handwriting recognition: Algorithms and applications. *Int. Journal on Document Analysis and Recognition*, 8(1):35–46, 2006.

[175] R. Plamondon and C. M. Privitera. The segmentation of cursive handwriting: An approach based on off-line recovery of the motor-temporal information. *IEEE Trans. on Image Processing*, 8(1):80–91, 1999.

[176] R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. PAMI*, 22(1):63–84, 2000.

[177] T. Ploetz and G. A. Fink. Markov models for offline handwriting recognition: A survey. *Int. Journal on Document Analysis and Recognition*, 12(4):269–298, 2009.

[178] I. Pratikakis, B. Gatos, and K. Ntirogiannis. H-DIBCO 2010 – handwritten document image binarization competition. In *Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition*, pages 727–732, 2010.

[179] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[180] T. M. Rath, V. Lavrenko, and R. Manmatha. A statistical approach to retrieving historical manuscript images without recognition. Tech. Rep. MM-42, Center for Intelligent Information Retrieval, 2003.

[181] T. M. Rath and R. Manmatha. Features for word spotting in historical manuscripts. In *7th Int. Conf. on Document Analysis and Recognition*, pages 218–222, 2003.

[182] T. M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, pages 521–527, 2003.

[183] T. M. Rath and R. Manmatha. Word spotting for historical documents. *Int. Journal on Document Analysis and Recognition*, 9:139–152, 2007.

[184] T. M. Rath, R. Manmatha, and V. Lavrenko. A search engine for historical manuscript images. In *Proc. 27th Int. Conf. on Research and Development in Information Retrieval*, pages 369–376, 2004.

[185] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco. Connectionist probability estimators in HMM speech recognition. *IEEE Trans. on Speech and Audio Processing*, 2(1):161–174, 1994.

[186] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950–959, 2009.

[187] K. Riesen and H. Bunke. *Graph Classification and Clustering Based on Vector Space Embedding*. World Scientific, 2010.

[188] J. Rocha and T. Pavlidis. Character recognition without segmentation. *IEEE Trans. PAMI*, 17(9):903–909, 1995.

[189] J. Rodriguez and F. Perronnin. Local gradient histogram features for word spotting in unconstrained handwritten documents. In *Proc. 1st Int. Conf. on Frontiers in Handwriting Recognition*, pages 7–12, 2008.

[190] J. Rodriguez and F. Perronnin. Handwritten word-spotting using hidden Markov models and universal vocabularies. *Pattern Recognition*, 42(9):2106–2116, 2009.

[191] V. Romero, A. H. Toselli, L. Rodríguez, and E. Vidal. Computer assisted transcription for ancient text images. In *Proc. 4th Int. Conf. on Image Analysis and Recognition*, pages 1182–1193, 2007.

[192] R. C. Rose, B. H. Juang, and C. H. Lee. A training procedure for verifying string hypotheses in continuous speech recognition. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 281–284, 1995.

[193] R. C. Rose and D. B. Paul. A hidden Markov model based keyword recognition system. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 129–132, 1990.

[194] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[195] J. L. Rothfeder, S. Feng, and T. M. Rath. Using corner feature correspondences to rank word images by similarity. In *Proc. Workshop on Document Image Analysis and Retrieval*, pages 30–35, 2003.

[196] J. L. Rothfeder, R. Manmatha, and T. M. Rath. Aligning transcripts to automatically segmented handwritten manuscripts. In *Proc. 7th Int. Workshop on Document Analysis Systems*, pages 84–95, 2006.

[197] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pages 318–362. MIT Press, 1986.

[198] J. Sadri, C. Y. Suen, and T. D. Bui. Application of support vector machines for recognition of handwritten Arabic/Persian digits. In *Proc. 2nd Conf. on Machine Vision and Image Processing & Applications*, pages 300–307, 2003.

[199] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Trans. on Acoustics, Speech, & Signal Processing*, 26:43–49, 1978.

[200] S. Saleem, H. Cao, K. Subramanian, M. Kamali, R. Prasad, and P. Natarajan. Improvements in BBN's HMM-based offline Arabic handwriting recognition system. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 773–777, 2009.

[201] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, 1971.

[202] A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):353–363, 1983.

[203] G. Saon. Cursive word recognition using a random field based hidden Markov model. *Int. Journal on Document Analysis and Recognition*, 1(4):199–208, 1999.

[204] J. Sauvola and M. Pietikainen. Adaptive document image binarization. *Pattern Recognition*, 33:225–236, 2000.

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

[206] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[207] M. Schambach. Fast script word recognition with very large vocabulary. In *Proc. Int. Conf. on Document Analysis and Recognition*, volume 1, pages 9–13, 2008.

[208] A. Schlapbach and H. Bunke. Off-line handwriting identification using HMM-based recognizers. In *Proc. 17th Int. Conf. on Pattern Recognition*, volume 2, pages 654–658, 2004.

[209] A. Schlapbach and H. Bunke. Off-line writer verification: A comparison of a hidden Markov model (HMM) and a Gaussian mixture model (GMM) based system. In *Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition*, pages 275–280, 2006.

[210] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[211] B. Schölkopf, A. J. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 327–352. MIT Press, 1999.

[212] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. on Signal Processing*, 45(11):2673–2681, 1997.

[213] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 244(1309):21–26, 1991.

[214] S. M. Selkow. The tree-to-tree editing problem. *Information Processing Letters*, 6(6):184–186, 1977.

[215] N. Serrano, L. Tarazón, D. Pérez, O. Ramos-Terrades, and A. Juan. The GIDOC prototype. In *Proc. 10th Int. Workshop on Pattern Recognition in Information Systems*, pages 82–89, 2010.

[216] D. Shaked and A. M. Bruckstein. Pruning medial axes. *Computer Vision and Image Understanding*, 29(2):156 – 169, 1998.

[217] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[218] D. Shi, Y. S. Ong, and E. C. Tan. Handwritten Chinese character recognition using kernel active handwriting model. In *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 1, pages 251–255, 2003.

[219] J. Shlens. A tutorial on principal component analysis. `http://www.snl.salk.edu/~shlens/pca.pdf`, 2009. Accessed February 20, 2012.

[220] N. Smith and M. Gales. Speech recognition using SVMs. In *Advances in Neural Information Processing Systems*, volume 14, pages 1197–1204, 2002.

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

[222] S. N. Srihari, X. Yang, and G. R. Ball. Offline Chinese handwriting recognition: A survey. *Frontiers of Computer Science in China*, 1(2):137–155, 2007.

[223] N. Stamatopoulos, G. Louloudis, and B. Gatos. Efficient transcript mapping to ease the creation of document image segmentation ground truth with text-image alignment. In *Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition*, pages 226–231, 2010.

[224] P. Stathis, E. Kavallieratou, and N. Papamarkos. An evaluation survey of binarization algorithms on historical documents. In *Proc. 19th Int. Conf. on Pattern Recognition*, pages 1–4, 2008.

[225] I. Steinwart, D. R. Hush, and C. Scovel. An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels. *IEEE Trans. on Information Theory*, 52(10):4635–4643, 2006.

[226] B. Su, S. Lu, and C. L. Tan. Binarization of historical document images using the local maximum and minimum. In *Proc. 9th Int. Workshop on Document Analysis Systems*, pages 159–166, 2010.

[227] P. N. Suganthan, E. K. Teoh, and D. P. Mital. Pattern recognition by homomorphic graph matching using Hopfield neural networks. *Image and Vision Computing*, 13(1):45–60, 1995.

[228] P. N. Suganthan and H. Yan. Recognition of handprinted Chinese characters by constrained graph matching. *Image and Vision Computing*, 16(3):191–201, 1998.

[229] T. Takiguchi and Y. Ariki. Robust feature extraction using kernel PCA. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 509–512, 2006.

[230] L. Tarazón, D. Pérez, N. Serrano, V. Alabau, O. Ramos-Terrades, A. Sanchis, and A. Juan. Confidence measures for error correction in interactive transcription handwritten text. In *Proc. 15th Int. Conf. on Image Analysis and Processing*, pages 567–574, 2009.

[231] K. Terasawa and Y. Tanaka. Slit style HOG features for document image word spotting. In *Proc. 10th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 116–120, 2009.

[232] S. Thomas, C. Chatelain, L. Heutte, and T. Paquet. An information extraction model for unconstrained handwritten documents. In *Proc. 20th Int. Conf. on Pattern Recognition*, pages 3412–3415, 2010.

[233] C. Tomai, B. Zhang, and G. Govindaraju. Transcript mapping for historic handwriting document images. In *Proc. 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 413–418, 2002.

[234] F. Tortorella. An optimal reject rule for binary classifiers. In *Proc. Joint IAPR Int. Workshops on Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition*, pages 611–620, 2000.

[235] A. H. Toselli, A. Juan, J. González, I. Salvador, E. Vidal, F. Casacuberta, D. Keysers, and H. Ney. Integrated handwriting recognition and interpretation using finite-state models. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 18(4):519–539, 2004.

[236] A. H. Toselli, V. Romero, M. Pastor, and E. Vidal. Multimodal interactive transcription of text images. *Pattern Recognition*, 43(5):1814–1825, 2010.

[237] A. H. Toselli, V. Romero, and E. Vidal. Viterbi based alignment between text images and their transcripts. In *Proc. Workshop on Language Technology for Cultural Heritage Data*, pages 9–16, 2007.

[238] G. T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12:261–268, 1980.

[239] O. D. Trier, A. K. Jain, and T. Taxt. Feature extraction methods for character recognition – A survey. *Pattern Recognition*, 29(4):641–662, 1996.

[240] O. D. Trier and T. Taxt. Evaluation of binarization methods for document images. *IEEE Trans. PAMI*, 17:312–315, 1995.

[241] W. H. Tsai and K. S. Fu. Error-correcting isomorphism of attributed relational graphs for pattern analysis. *IEEE Trans. on Systems, Man, and Cybernetics*, 9(12):757–768, 1979.

[242] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42, 1976.

[243] C. Viard-Gaudin, P. M. Lallican, P. Binter, and S. Knerr. The IRESTE on/off (IRONOFF) dual handwriting database. In *Proc. 5th Int. Conf. on Document Analysis and Recognition*, pages 455–458, 1999.

[244] A. Vinciarelli and S. Bengio. Off-line cursive word recognition using continuous density HMMs trained with PCA and ICA features. In *Proc. 16th Int. Conf. on Pattern Recognition*, volume 3, pages 81–84, 2002.

[245] A. Vinciarelli, S. Bengio, and H. Bunke. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Trans. PAMI*, 26(6):709–720, 2004.

[246] A. Vinciarelli and J. Luettin. Off-line cursive script recognition based on continuous density HMM. In *Proc. 7th Int. Workshop on Frontiers in Handwriting Recognition*, pages 493–498, 2000.

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

[248] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[249] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

[250] P. J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[251] F. Wessel, R. Schlüter, K. Macherey, and H. Ney. Confidence measures for large vocabulary continuous speech recognition. *IEEE Trans. on Speech and Audio Processing*, 9:288–298, 2001.

[252] M. Wienecke, G. Fink, and G. Sagerer. Toward automatic video-based whiteboard reading. *Int. Journal on Document Analysis and Recognition*, 7:188–200, 2005.

[253] R. Wilson, E. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. *IEEE Trans. PAMI*, 27(7):1112–1124, 2005.

[254] M. Wüthrich, M. Liwicki, A. Fischer, E. Indermühle, H. Bunke, G. Viehhauser, and M. Stolz. Language model integration for the recognition of handwritten medieval documents. In *Proc. 10th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 211–215, 2009.

[255] R. Xu and D. Wunsch. Survey of graph clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[256] S. J. Young, N. H. Russell, and J. H. S. Thornton. Token passing: A simple conceptual model for connected speech recognition systems. CUED/F-INFENG/TR38, Cambridge University Engineering Dept., 1989.

[257] B. Zhang, S. N. Srihari, and C. Huang. Word image retrieval using binary features. In *Proc. on Document Recognition and Retrieval XI*, volume 5296, pages 45–53. SPIE, 2003.

[258] M. Zimmermann and H. Bunke. Automatic segmentation of the IAM off-line database for handwritten English text. In *Proc. 16th Int. Conf. on Pattern Recognition*, volume 4, pages 35–39, 2002.

[259] M. Zimmermann and H. Bunke. Hidden Markov model length optimization for handwriting recognition systems. In *Proc. 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 369–374, 2002.

[260] M. Zimmermann and H. Bunke. N-gram language models for offline handwritten text recognition. In *Proc. 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 203–208, 2004.

[261] M. Zimmermann, J.-C. Chappelier, and H. Bunke. Offline grammar-based recognition of handwritten sentences. *IEEE Trans. PAMI*, 28(5):818–821, 2006.

[262] S. Zinger, J. Nerbonne, and L. Schomaker. Text-image alignment for historical handwritten documents. In *Proc. on Document Recognition and Retrieval XVI*, volume 7247, pages 1–8. SPIE, 2009.

Erklärung

gemäss Art. 28 Abs. 2 RSL 05

Name/Vorname: .....

Matrikelnummer: .....

Studiengang: .....

Bachelor ☐ Master ☐ Dissertation ☐

Titel der Arbeit: .....

.....

.....

LeiterIn der Arbeit: .....

.....

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe o des Gesetztes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.

.....
Ort/Datum

.....
Unterschrift

Curriculum Vitae

Personal Data

Name	Andreas Fischer
Date of birth	March 27, 1983
Address	Quellgasse 15
	2502 Biel, Switzerland
Home town	Bern, Switzerland
Nationality	Swiss

Education

2008 – 2012	PhD studies in Computer Science, University of Bern
2008	MSc in Computer Science, University of Bern
2003 – 2008	Computer Science studies, University of Bern
2001 – 2002	Physics studies, ETH Zürich
2001	High school graduation, Gymnasium Bern-Neufeld

Awards

IAPR Best Scientific Paper award for A. Fischer, A. Keller, V. Frinken, and H. Bunke. *HMM-Based Word Spotting in Handwritten Documents Using Subword Models*, Int. Conf. on Pattern Recognition, 2010.

IAM Alumni outstanding achievement award for master thesis. *Classification of Dissimilarity Space Embedded Graphs*, University of Bern, 2008.

List of Publications

Journal Articles and Book Chapters

[1] A. Fischer, V. Frinken, and H. Bunke. Application of hidden Markov models for handwriting recognition. To appear in *Handbook of Statistics*, volume 31. Elsevier, 2012.

[2] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, http://dx.doi.org/10.1016/j.patrec.2011.09.009, accepted for publication, 2012.

[3] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. A novel word spotting method based on recurrent neural networks. *IEEE Trans. PAMI*, 34(2):211–224, 2012.

Conference Papers **(peer-reviewed)**

[1] A. Garz, A. Fischer, R. Sablatnig, and H. Bunke. Binarization-free text line segmentation for historical documents based on interest point clustering. In *Proc. 10th Int. Workshop on Document Analysis Systems*, accepted for publication, 2012.

[2] A. Fischer, E. Indermühle, V. Frinken, and H. Bunke. HMM-based alignment of inaccurate transcriptions for historical documents. In *Proc. 11th Int. Conf. on Document Analysis and Recognition*, pages 53–57, 2011.

[3] V. Frinken, A. Fischer, H. Bunke, and A. Fornés. Co-training for handwritten word recognition. In *Proc. 11th Int. Conf. on Document Analysis and Recognition*, pages 314–318, 2011.

[4] E. Indermühle, V. Frinken, A. Fischer, and H. Bunke. Keyword spotting in online handwritten documents containing text and non-text using BLSTM neural networks. In *Proc. 11th Int. Conf. on Document Analysis and Recognition*, pages 73–77, 2011.

[5] A. Fischer, V. Frinken, A. Fornés, and H. Bunke. Transcription alignment of Latin manuscripts using hidden Markov models. In *Proc. 1st Int. Workshop on Historical Document Imaging and Processing*, pages 29–36, 2011.

[6] A. Fornés, V. Frinken, A. Fischer, J. Almazán, G. Jackson, and H. Bunke. A keyword spotting approach using blurred shape model-based descriptors. In *Proc. 1st Int. Workshop on Historical Document Imaging and Processing*, pages 83–90, 2011.

[7] A. Fischer and H. Bunke. Character prototype selection for handwriting recognition in historical documents with graph similarity features. In *Proc. 19th European Signal Processing Conference*, pages 1435–1439, 2011.

[8] V. Frinken, A. Fischer, and H. Bunke. Improving handwritten keyword spotting with self-training. In *Proc. 26th Symposium On Applied Computing*, pages 838–843, 2011.

[9] A. Fischer, K. Riesen, and H. Bunke. Graph similarity features for HMM-based handwriting recognition in historical documents. In *Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition*, pages 253–258, 2010.

[10] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. Adapting BLSTM neural network based keyword spotting trained on modern data to historical documents. In *Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition*, pages 352–257, 2010.

[11] A. Fischer, A. Keller, V. Frinken, and H. Bunke. HMM-based word spotting in handwritten documents using subword models. In *Proc. 20th Int. Conf. on Pattern Recognition*, pages 3416–3419, 2010.

[12] A. Fischer, E. Indermühle, H. Bunke, G. Viehhauser, and M. Stolz. Ground truth creation for handwriting recognition in historical documents. In *Proc. 9th Int. Workshop on Document Analysis Systems*, pages 3–10, 2010.

[13] V. Frinken, A. Fischer, and H. Bunke. A novel word spotting algorithm using bidirectional long short-term memory neural networks. In *Proc. 4th Int. Workshop on Artificial Neural Networks in Pattern Recognition*, pages 185–196, 2010.

[14] V. Frinken, A. Fischer, and H. Bunke. Combining neural networks to improve performance of handwritten keyword spotting. In *Proc. 9th Int. Workshop on Multiple Classifier Systems*, pages 215–224, 2010.

[15] A. Fischer, M. Wüthrich, M. Liwicki, V. Frinken, H. Bunke, G. Viehhauser, and M. Stolz. Automatic transcription of handwritten medieval documents. In *Proc. 15th Int. Conf. on Virtual Systems and Multimedia*, pages 137–142, 2009.

[16] A. Fischer and H. Bunke. Kernel PCA for HMM-based cursive handwriting recognition. In *Proc. 13th Int. Conf. on Computer Analysis of Images and Patterns*, pages 181–188, 2009.

[17] V. Frinken, T. Peter, A. Fischer, H. Bunke, T.-M.-T. Do, and T. Artieres. Improved handwriting recognition by combining two forms of hidden Markov models and a recurrent neural network. In *Proc. 13th Int. Conf. on Computer Analysis of Images and Patterns*, pages 189–196, 2009.

[18] M. Wüthrich, M. Liwicki, A. Fischer, E. Indermühle, H. Bunke, G. Viehhauser, and M. Stolz. Language model integration for the recognition of handwritten medieval documents. In *Proc. 10th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 211–215, 2009.

[19] A. Fischer, K. Riesen, and H. Bunke. An experimental study of graph classification using prototype selection. In *Proc. 19th Int. Conf. on Pattern Recognition*. IEEE, December 2008.