Language Model Integration for the Recognition of Handwritten Medieval Documents

Markus Wüthrich[*], Marcus Liwicki[†], Andreas Fischer[⋆], Emanuel Indermühle[⋆], Horst Bunke[⋆]
Gabriel Viehhauser[‡], Michael Stolz[‡]

Abstract

Building recognition systems for historical documents is a difficult task. Especially, when it comes to medieval scripts. The complexity is mainly affected by the poor quality and the small quantity of the data available. In this paper we apply an HMM *based recognition system to medieval manuscripts from the 13th century written in Middle High German. The recognition system, which was originally developed for modern scripts, has been adapted to medieval scripts. Beside the data processing, one of the major challenges is to create a suitable language model. Because of the lack of appropriate independent text corpora for medieval languages, the language model has to be created on the base of a rather small number of manuscripts only. Due to the small size of the corpus, optimizing the language model parameters can quickly lead to the problem of overfitting. In this paper we describe a strategy to integrate all available information into the language model and to optimize the language model parameters without suffering from this problem.*

1. Introduction

In recent years, interest in the analysis and recognition of historical documents has grown strongly [2]. In order to preserve valuable old handwritings, a huge number of original documents have been photographed or scanned and are available in form of digital images. Examples include writings of famous presidents, *e.g.,* George Washington's papers at the Library of Congress, or scientists, *e.g.,* Sir Isaac Newton's writings at the University of Cambridge Library. Together with the creation of these large collections, there is an increasing demand for accessing text document images in digital libraries [1]. A content based indexing is required to search and browse the images. Hence, analysis and recognition of the handwriting is needed for automation.

The automatic reading of historical documents is an *off-line* task, where only the images of the documents are available. This task is considered to be harder than *on-line* recognition, where also temporal information can be exploited [14]. For restricted domains with modern scripts, commercial *off-line* systems are available, *e.g.,* for postal address [17] and bank check reading [10]. For the task of historical document processing, however, only little work exists. This is due to many difficulties, including the low quality of the original paper or parchment, ink bleed-through, stains, holes, and other adverse artifacts.

Considering the difficulties mentioned it is not astonishing that the problem of automatically generating textual transcriptions of historical documents is still unsolved [5]. Some approaches try to avoid the complete transcription of the documents. On the one hand, word spotting aims at efficiently matching keywords against the document images directly by segmenting the page into word images, performing a clustering based on global features, and matching the keywords against the labeled word clusters [15]. On the other hand, computer aided manual transcription is attempted [3]. While avoiding to generate a complete transcription, these systems do not benefit from any linguistic information, which has shown to be successful for modern language recognition.

In the present paper we apply an *HMM* based recognition system to medieval manuscripts from the 13th century written in *Middle High German*. This system was adapted from another recognizer that has proven to be quite successful for modern handwritings [13]. Since language information is very important for generating good recognition systems [16, 18], we investigate different strategies for generating and optimizing statistical language models in this paper. In contrast to modern languages, where the language information can be extracted from huge corpora, such as the Brown Corpus [7], the language model for *Middle High*

[*]Institute of Computer Science and Applied Mathematics, University of Bern, Neubrückstrasse 10, CH-3012 Bern, { mwuethri, afischer, eindermu, bunke }@iam.unibe.ch

[†]German Research Center for Artificial Intelligence (DFKI), Trippstadter Strasse 122, D-67663 Kaiserslautern, marcus.liwicki@dfki.de

[‡]Institut für Germanistik, University of Bern, Länggassstrasse 49, CH-3012 Bern, { michael.stolz,viehhauser }@germ.unibe.ch

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.17

German had to be constructed from the manuscript data. In the application considered in the present paper, finding a good optimization strategy of the language model parameters is rather delicate. Due to the small size of the corpus, we are facing a considerable overfitting problem. To provide a solution to this problem is one of our main concerns in this paper.

The rest of the paper is organized as follows. First, Section 2 describes the medieval data set. Second, in Section 3 our *HMM* based recognition system is introduced. Next, Section 4 motivates the use of language models and presents our strategies. Subsequently, experimental results are reported in Section 5. Finally, Section 6 draws some conclusions and gives an outlook.

2. Data

The data used in this paper originate from an epic poem called *Parzival*. This poem by *Wolfram von Eschenbach* dates approximately from the 13th century and is written in *Middle High German*.

Parzival is arranged into 16 books and the poem is written in pairwise rhyming lines. There exist multiple manuscripts of this epic poem, which differ in several ways. For example, they have different writing styles, different dialects, and even the contents slightly differ. Furthermore, the available transcriptions differ heavily between the manuscripts. The manuscript used for the experiments described in this paper is *St. Gall, collegiate library, cod. 857*. It consists of 318 folios and the transcription of '*Parzival*' is available for only about 45 folios. The pages have two columns of text. There are totally 4,478 lines of text with transcriptions available. Figure 1 shows an example page from the manuscript used in our experiments.

Because of the age and the rather low quality of the parchment, medieval manuscripts need special preprocessing. Since the focus of this paper is on the recognition and the language models, some preprocessing steps are manually performed. In the following, the problems occurring in the images and the preprocessing steps applied are listed:

1. The manuscript contains colored initial characters and decorations (see Figure 1 for an example). In some cases, these decorations are within the text column or even span multiple columns. In order to simplify the recognition task, such decorations and characters have been removed manually from the affected pages.
2. There are also other characteristics of the parchment that make line segmentation and the recognition difficult, *e.g.,* seams or holes. These types of artifacts are also manually removed.
3. The text is written in two columns. In order to apply line segmentation after the preprocessing steps, the columns are separated from each other.
4. The result of the last step is one image per text column. These images still contain the colored background. Therefore, a *Difference of Gaussian* (DoG) edge-detection method was automatically applied on these images. This method produces a grayscale picture of the writing without the background.
5. A binarization of the images is performed to eliminate the remaining background noise.

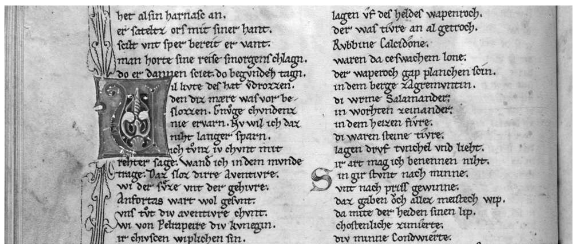

Figure 1. Example page from the epic poem. St. Gall, collegiate library, cod. 857, page 262

The transcriptions of the manuscript are not directly available in *ASCII* text, because they are stored in the *TUSTEP*[1] file format, which is a powerful tool for managing transcriptions for Latin and non-Latin manuscripts. This format was converted into a format suitable for recognition. For detailed information we refer to [19].

3. Recognition System

The recognition system used in this paper is based on Hidden Markov Models (*HMM*). Some of its basic components are similar to the recognizer described in [13].

The input to the recognition system is a single line of text. In order to get these single text lines, a line segmentation method based on dynamic programming is applied on the pages of the manuscript [12]. To obtain better recognition results, several preprocessing steps are applied to the image data. Firstly, there are the special preprocessing steps described in Section 2, which deal with the problems of medieval manuscripts and are performed on complete manuscript pages. Secondly, there are several preprocessing steps that are applied on the segmented text lines, *i.e.,* skew correction, line positioning and width correction. The skew correction horizontally aligns the writing, the line positioning normalizes the extend of the three main writing zones (lower, middle, and upper) and the width normalization affects the width of the characters. Because of the regular upright writing, no slant correction is needed.

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

The features for the recognizer are extracted using a sliding window. In this paper the same features as in [13] are used, namely, the number of black pixels, the position of the uppermost pixel, the orientation of the uppermost pixel, the position of the lowermost pixel, the orientation of the lowermost pixel, the proportion of black pixels to the number of pixels between uppermost and lowermost pixel, the center of gravity, the second derivative of the moment in vertical direction, and the number of black-to-white transitions in vertical direction.

4. Language Models

The purpose of integrating a language model is to improve a recognition system by using statistical information about the language structure in the recognition process [13]. This is motivated by the fact that humans can easily predict a word while reading.

In this paper we use statistical *n*-gram language models. An *n*-gram language model stores statistical information about word sequences. Such a language model can be used to predict a word if the *n - 1* preceding words are known. The probability $p(W)$ of a word sequence $W = (w_1, ..., w_m)$ is given by Eq. (1). Eq. (2) is an approximation of Eq. (1) by limiting the context to *n - 1* words. The resulting probabilities are called *n*-gram language model.

$$p(W) = p(w_1) \prod_{i=2}^{m} p(w_i | w_1, ..., w_{i-1}) \quad (1)$$

$$p(W) \simeq p(w_1) \prod_{i=2}^{m} p(w_i | w_{i-n+1}, ..., w_{i-1}) \quad (2)$$

The probabilities of word sequences are usually extracted from a large set of texts (corpus), independent of the training, validation and test set. A simple method to estimate $p(w_i | w_{i-n+1}, ..., w_{i-1})$ is to divide the number of occurrences of word sequence $w_{i-n+1}, ..., w_i$ by the number of occurrences $w_{i-n+1}, ..., w_{i-1}$.

However, in practice, this estimation is unreliable because there are many word sequences which never occur in the text corpus. These word sequences have a probability of zero. Especially for small text corpora, this phenomenon occurs frequently. However, a word sequence should not have a probability of zero because this possibly leads to a situation where the recognition system is not able to find the correct solution. There are several smoothing methods available that solve this problem [4]. In this paper, the *Kneser-Ney* smoothing method is used [11].

The problem for the *Parzival* manuscript is that there are no text corpora available that contain sufficiently large amounts of text in the *Middle High German* language in order to estimate the probabilities $p(w_i | w_{i-n+1}, ..., w_{i-1})$ reliably. One possible solution for this problem is to use a combination of the training and the validation[2] set to compile a useful corpus for the creation of the language models.

The *HMM* based recognition system used in this paper supports the integration of *n*-gram language models during the decoding. In an *HMM* based recognition system the decoding is performed with the Viterbi algorithm, which finds the optimal path through the states of an *HMM*. The integration of the *n*-gram language models can be recursively expressed as follows:

$$\begin{aligned} \phi(s_i) &= \phi(s_{i-1}) + \log p(F_i | w_i) + \\ &\quad \alpha \log p(w_i, w_{i-n+1}, ..., w_{i-1}) + \beta \end{aligned}$$

where $\phi(s_i)$ is the score of the word sequence $s_i = (w_1, ..., w_i)$ and $p(F_i | w_i)$ is the likelihood returned by the word *HMM* for w_i and feature sequence F_i. The probability $p(w_i, w_{i-n+1}, ..., w_{i-1})$ is the *n*-gram language model probability. The parameter α is also known as the *Grammar Scale Factor* (GSF), because it weights the influence of the language model, while the parameter β controls the segmentation rate of the recognizer and is called *Word Insertion Penalty* (WIP). Both parameters are usually determined experimentally on the validation set.

5. Experiments

For the experiments 4,478 text lines from the manuscript described in Section 2 were used. The data was divided into three sets: a training set, a validation set, and a test set. The sets are equally distributed over the pages because it is not desirable to train a recognizer on pages containing poor quality data and test it on pages containing good quality data, and vice versa. The training set contains 2,237 text lines ($\sim$ 50%), the validation set 912 text lines ($\sim$ 20%), and the test set 1,329 text lines ($\sim$ 30%). Training and validation set together contain 3,980 and the full database contains 4,937 distinct word classes.

The recognition system uses an *HMM* for each of the characters in the alphabet. These *HMMs* have a linear topology and were trained on the training set. Each *HMM* uses 16 hidden states including the non-emitting start and end states. The parameter optimization on the validation set includes the number of Gaussian mixture components, the grammar scale factor, and the word insertion penalty. Four iterations were made for each increase of the number of Gaussian mixture components. This setting has been adopted from [9].

In order to measure the influence of the vocabulary size and the language model, several versions of the recognition

[2]Depending on the actual task, the test set may also be used.

Systems	A	B	B*	C
1,000	63.52	63.83	64.15	65.39
2,000	66.79	66.79	**68.50**	70.71
3,000	69.94	68.59	**70.36**	73.89
3,980	-	70.73	**73.00**	77.34
4,937	-	-	-	81.50

Table 1. Word accuracy (%) on the test set depending on different sizes of the vocabulary.

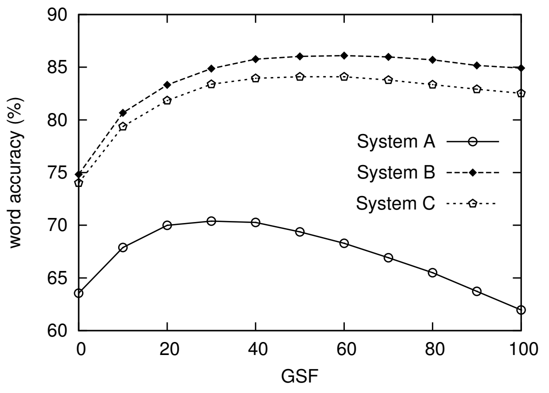

Figure 2. Influence of the grammar scale factor on the word accuracy on the validation set for the different systems.

system were created and compared with each other. The differences between these systems are the source of their vocabulary and the corpus from which their language model is generated. All language models in this paper are bigram language models. As basic performance measure the *word accuracy* was used.

The following systems were initially tested.

System A uses the vocabulary and the language model created from the training set.

System B uses the vocabulary and the language model created from the union of the training and the validation set.

System C is built from the training, validation and test set. The vocabulary is built from all sets, while the language model is only created from the training and the validation set. It can be argued that the test set should not be touched for the creation of a recognition system. However, the purpose of System C is to provide an upper limit of the recognition rate by incorporating all words occurring in the test set.

All three systems were tested with different sizes of their vocabulary, including the $1,000$, $2,000$, and $3,000$ most frequent words. Additionally, a version of each system B and C with the full vocabulary of the training and validation set, and a version of System C with the full vocabulary of training, validation, and test set were tested.

The results of the systems on the test set are summarized in Table 1. The results show that System A and System B perform similarly. Obviously, System C performs best because, in contrast with systems A and B, its vocabulary contains all words occurring in the test set.

Figure 2 shows the influence of the grammar scale factor on the word accuracy for the validation set. The optimal grammar scale factor is between 50 and 60 for the systems B and C. The optimum for System A is 30, which is considerably smaller than for the other two systems. This shows that the grammar scale factor has an important influence on the word accuracy.

From Table 1 and Figure 2 we conclude that System A and System B perform similarly on the test set, whereas System B and System C perform similar on the validation set. System A has the worst performance on the validation set. The reason is that the language model and the vocabulary from System B and System C were built from the validation set itself. Furthermore, the influence of the grammar scale factor is different for System A. This leads to the assumption that the system suffers from an overfitting effect on the validation set. Therefore, a new system (System B*) was created and tested.

System B* and System B use the same vocabulary and language model. However, instead of using the optimized parameter values (grammar scale factor, word insertion penalty, and number of Gaussian mixture components) from the validation set, System B* uses the optimized values from System A. This strategy is motivated by the fact that the language model for System A has only been generated on the data of the training set. Therefore, the data for optimizing the language model parameters are from different sets, preventing the system from overfitting.

A comparison of systems B and B* is given in Table 1. System B* performs statistically significantly better than System B for vocabulary sizes of $2,000$ and more (indicated by bold face). It can be concluded that the assumption of an overfitting is true and that the described approach can be used to overcome this problem.

A closer look at Table 1 reveals furthermore that the relative improvement increases for larger vocabulary sizes. This is an interesting observation, because we are facing an open vocabulary recognition task. Larger vocabulary sizes will possibly lead to an even better performance.

6. Conclusion and Outlook

In this paper we address the problem of recognizing medieval manuscripts from the 13th century written in *Middle High German*. We apply an *HMM* based recognition system, which was originally developed for modern handwritings. The best word accuracy on the test set is 73.00% with an open vocabulary. This is a very promising result for the difficult task of automatically generating textual transcriptions of historical documents. All experiments were performed in a writer-independent fashion. Therefore, it can be expected that the system can be adapted to other historical documents with minimum effort.

In modern handwriting recognition, language models are built from large independent text corpora. This is in most cases not feasible for old languages, because such corpora do not exist. Therefore, people are forced to use the ground truth of the available manuscripts for the creation of the language models. In a straight forward approach, all available data, *i.e.,* the training and the validation set, would be used to create a language model. However, optimizing the language model parameters on the validation set is rather delicate and can lead to overfitting.

Therefore, we propose to use only part of the available data, *i.e.,* the training set, to create an initial language model and optimize the language model parameters on the independent validation set. In a second step, the language model is generated from all available data and is used in the recognition system with the optimized parameters from the first step. In doing so, the language model takes all available data into account without suffering from overfitting. In our experiments a significant increase of the word accuracy was observed with applying this approach. We are aware of other approaches, which determine the language model parameters using the statistics of the data [6]. However, in our approach, we directly use the resulting word accuracy for optimization instead of estimating the possible recognition rate. This has been shown to be efficient in previous work as well.

There are still many issues about medieval manuscripts that need to be investigated, *e.g.,* the automatic extraction and the recognition of the initial letters, the recognition of abbreviations, experiments on larger data sets, or the use of other recognizers (see for example [8]).

7. Acknowledgment

This work has been supported by the Swiss National Center of Competence in Research (NCCR) under the project Interactive Multimodal Information Management (IM2).

References

[1] *Second International Workshop on Document Image Analysis for Libraries (DIAL 2006), 27-28 April 2006, Lyon, France*. IEEE Computer Society, 2006.

[2] *International Journal on Document Analysis and Recognition, Special Issue on the Analysis of Historical Documents*, volume 9. Springer, 2007.

[3] F. L. Bourgeois and H. Emptoz. Debora: Digital access to books of the renaissance. *International Journal on Document Analysis and Recognition (IJDAR)*, 9(2-4):193–221, 2007.

[4] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University, 1998.

[5] A. Downton, J. He, and S. Lucas. User-configurable ocr enhancement for online natural history archives. *International Journal on Document Analysis and Recognition (IJDAR)*, 9(2):263–279, 2007.

[6] G. Fink. *Markov Models for Pattern Recognition: From Theory to Applications*. Berlin Springer, 2007.

[7] W. Francis and H. Kucera. *Manual of information to accompany a standard sample of present-day edited American English for use with digital computers*. Department of Linguistics, Brown University, 1979.

[8] A. Graves, S. Fernandez, J. Schmidhuber, M. Liwicki, and H. Bunke. Unconstrained online handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems 21*, 2007.

[9] S. Günter and H. Bunke. HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and gaussian components. *Pattern Recognition*, 37:2069–2079, 2004.

[10] S. Impedovo, P. Wang, and H. Bunke. *Automatic Bankcheck Processing*. World Scientific, 1997.

[11] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Proc. International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184, 1995.

[12] M. Liwicki, E. Indermühle, and H. Bunke. On-line handwritten text line detection using dynamic programming. In *Proc. 9th Int. Conf. on Document Analysis and Recognition*, pages 447–451, 2007.

[13] U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Journal of Pattern Recognition and Art. Intelligence 15*, pages 65–90, 2001.

[14] R. Plamondon and S. Srihari. Online and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:63–84, 2000.

[15] T. M. Rath and R. Manmatha. Word spotting for historical documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 9(2-4):139–152, 2007.

[16] R. Rosenfeld. Two decades of statistical language modeling: where do we go from here? In *Proc. of the IEEE*, volume 88, pages 1270–1278, 2000.

[17] S. Srihari, Y. Shin, V. Ramanaprasad, and D. Lee. A system to read names and addresses on tax forms. *Proceedings of the IEEE*, 84(7):1038–1049, 1996.

[18] A. Vinciarelli, S. Bengio, and H. Bunke. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Trans. PAMI*, 26:709–720, 2003.

[19] M. Wüthrich. Automatic Recognition of Medieval German Manuscripts. Master's thesis, University of Bern, Switzerland, 2008.