© 2021 World Scientific Publishing Company
https://doi.org/10.1142/9789811203244_0005

Chapter 5

Automatic Handwriting Recognition in Historical Documents

Andreas Fischer

5.1 Introduction

Automated reading of handwriting is an intriguing and largely unsolved problem in computer science [Lorette (1999)]. Despite decades of continuous progress, the goal to match or even surpass the astounding capabilities of humans to read handwritten text is still far from being reached [Bunke and Varga (2007)]. One of the main reasons is that automatic systems still lack semantic understanding that is often needed to decipher difficult cases, such as handwritten notes taken years ago with crossed out text and abbreviated words. This is especially true for ancient manuscripts, where the transcription of a phrase may only be found through expert knowledge in paleography, linguistics, and history.

Standard recognition techniques for optical character recognition (OCR) that are based on character segmentation, followed by character recognition, typically fail in the case of handwriting. When facing variable character shapes and touching characters, recognition is needed to perform segmentation and, *vice versa*, segmentation is needed to perform recognition. This conundrum, also known as Sayre's paradox [Sayre (1973)], requires special recognition techniques that perform segmentation and recognition at the same time.

A well-established approach to handwriting recognition is based on Hidden Markov Models (HMM) [Ploetz and Fink (2009)], inspired by their success for modeling sequences in speech recognition [Rabiner (1989)]. In the past

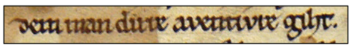
dem man dirre aventivre giht.

Fig. 5.1: Learning sample for handwriting recognition.

decade, recurrent neural networks have also been applied with great success to the task of handwriting recognition, especially when using long short-term memory cells (LSTM) [Graves *et al.* (2009)]. Both HMM and LSTM are able to perform segmentation-free recognition, i.e. they do not require annotated start and end positions of characters and words within text line images. Instead, it is sufficient to know the transcription of the text lines, which greatly reduces the effort for human annotators when preparing the training data. An example is illustrated in Figure 5.1.

Besides the training of appearance models, HMM as well as LSTM allow to integrate language models during recognition [Marti and Bunke (2001); Zamora-Martínez *et al.* (2014)], which are trained on electronic text corpora and guide the recognition towards the most plausible recognition alternative in the underlying language.

In the HisDoc project, we have adopted the strategy to work with text line images and character models. For the layout analysis module (see Chapter 4), extracting text line images is a feasible goal since they can often be found with high accuracy based on their regular structure without recognizing individual characters or words. For the text recognition module, modeling characters has several advantages when compared with modeling words. Most importantly, word models would require human annotators to provide several learning samples for each word class. Character models are shared across different words and can be combined for modeling any word, even if it is not listed in a dictionary. From the beginning of the project, both HMM and LSTM have been investigated for character modeling in historical manuscripts [Fischer *et al.* (2009)].

In the following, we present the HisDoc handwriting recognition system in more detail. Section 5.2 describes image preprocessing and feature extraction, Section 5.3 elaborates character modeling with HMM and LSTM, Section 5.4 presents experimental results for automatic transcription, and Section 5.5 describes several extensions that have been proposed during the HisDoc project. Concluding remarks are provided in Section 5.6.

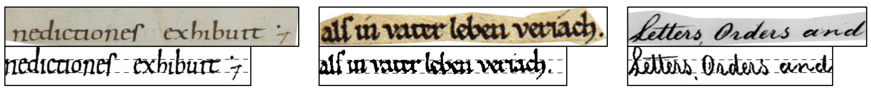

Fig. 5.2: Text line image normalization.

5.2 Image Preprocessing and Feature Extraction

Handwriting may differ greatly in style, size, and orientation, even when focusing on the main text of a single historical manuscript. Also, the page background can change significantly from one page to another. Image preprocessing aims to remove some of these variations to support handwriting recognition. However, if the preprocessing fails, unwanted distortions may be introduced that render recognition more difficult. Hence, a balance has to be found how much preprocessing, if any, should be applied to the original text line images.

In the HisDoc system, we first perform a binarization in order to separate the ink foreground from the page background. The binarization method consists of a local edge enhancement with a Difference of Gaussians filter, followed by a global threshold. Text lines are then extracted from the binarized page image according to their enclosing polygons. Afterwards, a series of transformations are applied to the text line images as suggested in [Marti and Bunke (2001)]. First, a rotation is applied to correct the skew, i.e. the inclination of the text line. It is estimated using a linear regression on the lower contour. Second, a shearing operation is applied to remove the slant, i.e. the inclination of the letters. It is estimated on long vertical segments of the handwriting. Third, vertical scaling is applied to obtain three writing regions of equal height, i.e. the descenders, the lowercase letters, and the ascenders. Here, the height of the lowercase letters is estimated using a linear regression on the upper contour. Finally, horizontal scaling is used to standardize the width of the characters. To that end, the number of black-white transitions on the middle line from left to right is normalized. Figure 5.2 provides examples for the three datasets of the IAM-HistDB (see Chapter 2), highlighting the baseline and the height of the lowercase letters with dashed lines.

After image preprocessing, a sliding window is moved in reading direction from left to right over the normalized text line image as illustrated in Figure 5.3. The aim of this sliding window approach is to segment the handwriting into smaller parts that can then be connected to characters

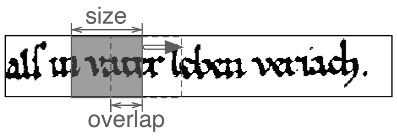

Fig. 5.3: Feature extraction with sliding window.

and words during recognition. Each window is described with n real-valued features, leading to a representation of the text line with a feature vector sequence

$$x = x_1, \ldots, x_N, \quad x_i \in \mathbb{R}^n. \tag{5.1}$$

Feature extraction can help to focus the recognition system on a few important properties of the handwriting, thus reducing the complexity of the character models. However, it can also lead to the loss of important information present in the original image. Similar to image preprocessing, a balance has to be found how much data reduction, if any, should be applied.

In the HisDoc system, we have mainly used the nine geometric features proposed in [Marti and Bunke (2001)]. They are based on a narrow sliding window with a size of one image column. Three global features include the fraction of black pixels, the center of gravity, and the second order moment. Six local features include the position of the upper and lower contours, the gradient of the contours, the fraction of black pixels between the contours, and the number of black-white transitions.

5.3 Character Modeling

When pursuing the sliding window approach, handwriting recognition can be stated as a sequence-to-sequence problem. The goal is to extract a character sequence $c_1, \ldots, c_M$ from the feature vector sequence $x_1, \ldots, x_N$ where the number of characters is less than or equal to the number of feature vectors ($M \leq N$).

We have investigated two types of machine learning models for this task, which are described briefly in the following. HMM are generative models that aim to estimate the probability density function of the features in order to deduce the most probable character sequence. LSTM are discriminative models that aim to directly estimate the posterior probabilities of the characters for each feature vector. Both models rely on efficient dynamic

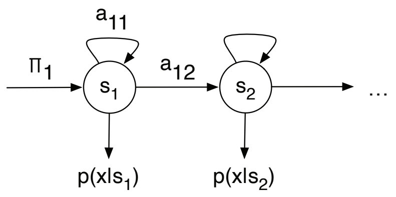

Fig. 5.4: HMM character model with linear topology.

programming algorithms for aligning the character sequence with the feature vector sequence during training and recognition.

5.3.1 *HMM Character Models*

For HMM-based recognition, character models with linear topology are considered as illustrated in Figure 5.4. The different states represent the beginning, middle, and end parts of the character along the feature vector sequence. The model starts in the first state (initial probability $\pi_1 = 1.0$), where it either stays with probability a_{11} or changes to the second state with probability a_{12} (transition probabilities $a_{11} + a_{12} = 1.0$). Similarly, the model either rests in a state or changes to the next one until the final state is reached. At each state s_i, the probability density function of the n real-valued features is modeled with $p(x|s_i)$, $x \in \mathbb{R}^n$. For handwriting, the character shapes and the connection between two characters can be highly variable, leading to complex probability density functions. We model them with a mixture of Gaussians, using diagonal covariance matrices in order to reduce the number of trainable model parameters.

During training, the HMM character models are concatenated according to the text line transcription. Their model parameters are optimized with the Baum-Welch algorithm [Rabiner (1989)], which uses dynamic programming to take all possible state transitions into account. During recognition, as illustrated in Figure 5.5, HMM character models are concatenated to words from a dictionary and the text line is interpreted as a sequence of words, which are separated by the whitespace character “sp”. The most likely sequence of state transitions is found with the Viterbi algorithm [Rabiner (1989)], which also uses dynamic programming to efficiently explore all possible state transitions.

Although the use of a word dictionary makes it impossible for the system to recognize out-of-vocabulary words, it is considered a best practice for

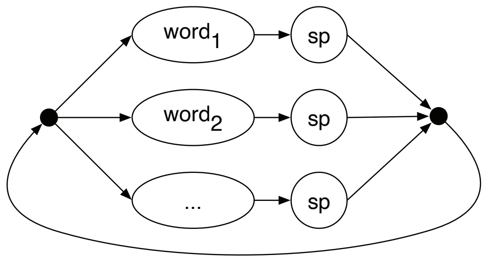

Fig. 5.5: Text line recognition model.

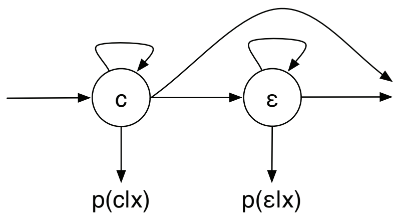

Fig. 5.6: LSTM character model.

handwriting recognition. Taking into account a relatively low character recognition accuracy when compared with printed text, using a dictionary allows to recover from character recognition errors and ensures that only valid words are transcribed. Furthermore, it allows to integrate language models at word level, which estimate the *a priori* probability of word sequences. In the HisDoc system, we rely on statistical n-gram language models with Kneser-Ney smoothing in order to account for unseen word combinations [Kneser and Ney (1995)].

Formally, following Bayes' rule, the HMM-based approach finds the best word sequence w^* with respect to

$$w^* = \arg\max_{w \in \mathcal{W}} p(w|x) = \arg\max_{w \in \mathcal{W}} p(x|w)p(w) \tag{5.2}$$

where $\mathcal{W}$ is the set of all valid word sequences. The posterior probability $p(w|x)$ is maximized indirectly by means of the HMM likelihood $p(x|w)$ and the language model prior $p(w)$. In practice, likelihood and prior are balanced with a grammar scale factor, and a word insertion penalty is used to control the number of recognized words [Marti and Bunke (2001)].

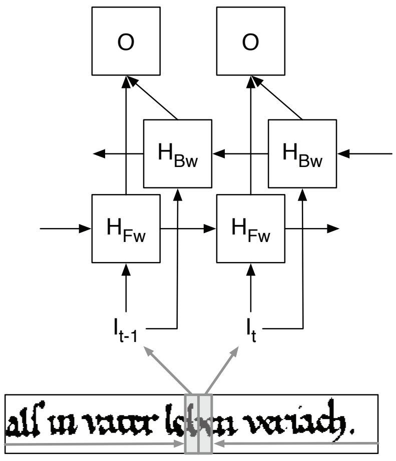

Fig. 5.7: Bidirectional LSTM network.

5.3.2 *LSTM Character Models*

For LSTM-based recognition, we consider a two-state character model suggested in the context of connectionist temporal classification (CTC) [Graves *et al.* (2009)]. The model is either in the character state “c” or in the no-character state “ϵ”. In both states, the output neuron of a recurrent neural network estimates directly the state posteriors $p(c|x)$ and $p(\epsilon|x)$, respectively, for the feature vector $x \in \mathbb{R}^n$. The model is allowed to switch directly from one character to the next, or to a no-character state in between.

Figure 5.7 illustrates the bidirectional, recurrent neural network that is used in the HisDoc system. At each position of the sliding window, the feature vector is provided as input to two hidden layers of the neural network, one for the forward direction and one for the backward direction. The hidden layers also receive input from the previous time step (forward direction) and the next time step (backward direction), respectively. Both hidden layers are connected to the same output layer, which contains one output neuron per character, plus an additional neuron for the no-character state.

Instead of standard perceptrons, long short-term memory cells (LSTM) [Hochreiter and Schmidhuber (1997)] are used in the recurrent hidden layers. Overcoming the vanishing gradient problem, they are able to keep information over a relatively large number of time steps, controlling the flow of information with special input, output, and forget gates. The gates

Table 5.1: Datasets used for evaluating automatic transcription. Number of text lines in the training, validation, and test sets, number of characters in the alphabet, and number distinct words in the dictionary.

Dataset	Text Lines			Alphabet	Dictionary
	Train	Valid	Test		
Saint Gall	468	235	707	49	5762
Parzival	2237	912	1328	93	4934
George Washington	325	168	163	82	1471

are perceptrons that are either open or closed according to their activation function, i.e. the network is able to learn when to accept new input, when to produce an output, and when to reset the memory.

Similar to HMM, the LSTM character models are concatenated according to the text line transcription during training. The loss function is computed at each time step with the CTC [Alex Graves *et al.* (2006)] algorithm, which uses dynamic programming to consider all possible state transitions. The weights of the neural network are then optimized by means of backpropagation through time.

Recognition is also performed similar to HMM. The characters are concatenated to words of a dictionary, followed by space (see Figure 5.5), and the optimal sequence of words is found by means of dynamic programming based on their posterior probabilities. In addition to the LSTM-based appearance models, language model probabilities can be taken into account as well. For more details on the bidirectional LSTM architecture, we refer the reader to [Graves *et al.* (2009); Fischer *et al.* (2009)].

5.4 Automatic Transcription

The HisDoc handwriting recognition systems, both HMM-based and LSTM-based, have been experimentally evaluated on the three datasets of the IAM-HistDB (see Chapter 2), namely the Saint Gall, Parzival, and George Washington datasets.

Table 5.1 summarizes the experimental setup for each of the datasets. Several hundred text lines are used for training the machine learning systems, validating their meta-parameters, and testing their final performance. The number of character models that have to be trained, including special characters, varies from 49 (Saint Gall) to 93 (Parzival). Because it is difficult to compare recognition systems with different out-of-vocabulary conditions, we have adopted a common evaluation protocol with closed vocabulary, i.e. all words from the entire dataset are included in the recognition dictionary.

Two recognition tasks are considered: single word recognition and text line recognition. The former is based on a human-corrected segmentation of the text lines into isolated words. Each word of the dictionary is assumed to have the same *a priori* probability to be present in the isolated word image. The latter takes complete text lines into account and performs the recognition with the help of a word bigram language model. The language model is trained and optimized on the transcriptions of the training and validation sets. It predicts the probability of the next word based on its immediate predecessor during recognition.

The implementation of the HisDoc system is based on the HTK toolkit[1] for HMM-based recognition and on an early version of the RNNLIB library[2] for LSTM-based recognition. Language models are based on the SRILM toolkit[3] in both cases. For text line recognition with LSTM character models and word bigram language models, we have implemented our own decoder based on beam search and token passing [Fischer (2012)], which is capable to efficiently cope with very large dictionaries containing hundreds of thousands of words.

Several meta-parameters of the machine learning systems are optimized with respect to the recognition accuracy achieved on the validation set. For HMM-based recognition, parameters include the number of states and the number of Gaussian mixtures for the character models, as well as the grammar scale factor and the word insertion penalty for the language model integration. For LSTM-based recognition, most parameters are fixed with respect to suggestions in [Graves *et al.* (2009)] and some preliminary experiments, including a learning rate of 10^{-4}, a momentum of 0.9, and 100 LSTM cells both in the forward and backward hidden layers. Furthermore, the language model integration requires no parameter optimization, i.e. a grammar scale factor of 1 and a word insertion penalty of 0 are used. The validation set is only used for selecting the best out of ten random initialized networks, which is then evaluated on the test set.

[1]http://htk.eng.cam.ac.uk
[2]http://sourceforge.net/projects/rnnl/
[3]http://www.speech.sri.com/projects/srilm/

Table 5.2: Word recognition accuracy in percentage for automatic transcription of single word images and text line images, respectively.

Dataset	System	Single words	Text lines
Saint Gall	HMM	93.6	89.4
	LSTM	96.0	93.8
Parzival	HMM	85.9	84.5
	LSTM	91.1	93.3
George Washington	HMM	69.5	75.9
	LSTM	80.3	81.9

Table 5.2 presents the word recognition accuracy achieved for automatic transcription on the independent test set. In the case of single word recognition it corresponds directly with the percentage of correctly recognized words. In the case of text line recognition it is calculated as

$$Acc = \frac{N - S - D - I}{N} \tag{5.3}$$

with respect to the total number of words N in the ground truth transcription, the number of word substitution errors S, word deletion errors D, and word insertion errors I. The different types of errors are identified using the string edit distance, also known as Levenshtein distance, between the output of the recognition system and the ground truth transcription.

Overall, the HisDoc system achieves promising word accuracies over 80% on the George Washington dataset and over 90% on the Saint Gall and Parzival datasets. In our experiments, LSTM-based handwriting recognition systematically outperformed HHM-based recognition in all cases when using the same nine geometric features (see Section 5.2), highlighting the great potential of LSTM memory cells for modeling and recognizing complex sequential data.

When comparing single word recognition with text line recognition, we can observe that they achieve similar accuracies. This encourages the recognition at text line level, as they are often easier to extract from the page image than individual words.

The lower performance on the George Washington dataset can be explained by the fact that only 325 text lines are available for training a total amount of 82 character models. Furthermore, the George Washington dataset contains cursive handwriting with complex connections between the characters. More transcribed text line images for training are expected to further improve the results.

Note that the recognition accuracies reported in Table 5.2 should be interpreted as upper bounds. In a real-world scenario, any error during the text line extraction step (see Chapter 6) and any out-of-vocabulary word that is not present in the dictionary may lead to additional errors in the automatic transcription.

5.5 Extensions

One of the main constraints of automatic transcription based on machine learning is the relatively large number of training samples that human annotators have to provide. During the HisDoc project, we have developed two extensions of the handwriting recognition systems that specifically target this constraint, namely keyword spotting and transcription alignment.

Keyword spotting has been proposed early on in the literature [Manmatha *et al.* (1996)] as a means to identify specific search terms in handwritten historical documents, rather than performing a full transcription. In the query-by-example approach, only a single example image of the keyword is needed to retrieve similar images from a document collection. In the query-by-string approach, the user enters a search term with the keyboard and a weakly trained model is used for retrieval. We have contributed to the latter approach by introducing an HMM-based [Fischer *et al.* (2010, 2012)] and an LSTM-based [Frinken *et al.* (2012)] keyword spotting system that operate at the sub-word level, i.e. they are based on trained character models similar to automatic transcription. When compared with keyword spotting approaches at the word level [Rodriguez and Perronnin (2009)], they have the advantage that character models can be trained across different words and thus it is not necessary to collect examples of the search term beforehand. For more details on keyword spotting, we refer to Chapter 6.

Transcription alignment aims to match existing transcriptions with their corresponding images, which is not only useful for the end user when searching and browsing historical documents but also for automatically generating learning samples for handwriting recognition systems. Although many text editions of historical manuscripts have been made available by researchers in the humanities, most of them are not aligned at the text line level, i.e. it is not known where a specific text line is located in the image. In the case of diplomatic transcriptions, the characters in the transcription perfectly match the characters visible in the image and the problem of transcription alignment is reduced to finding the start and end positions of the individual words [Kornfield *et al.* (2004)]. However, the transcription can also deviate from the image, for example when abbreviations are written out in full or when the order of the words is changed for better readability. During the HisDoc project, we have addressed this problem with alignment systems based on character models and demonstrated their effectiveness even when the character models are only weakly trained [Fischer *et al.* (2011a,b)].

5.6 Conclusions

Although the accuracies achieved for handwritten text recognition in historical documents are still far from being perfect, the results obtained with the HisDoc system on the IAM-HistDB are promising. When recognizing text line images with LSTM character models, a training set of hundreds of transcribed text lines, a closed vocabulary, and word bigram language models estimated on the manuscript itself, we report a word accuracy of 93.8% on the Saint Gall dataset, 93.3% on the Parzival dataset, and 81.9% on the George Washington dataset.

The reported results should be interpreted as an upper bound of the performance that can be expected in practice, as they do not take errors into account stemming from text line extraction and out-of-vocabulary words. However, there are also possibilities to increase the performance when taking external resources into account, such as character models that are trained on similar handwritings and language models that are estimated on large electronic text corpora written in the language of the manuscript at hand.

Technically, handwriting recognition has further evolved in the past decade. For Latin manuscripts, working at text line level and performing the recognition with LSTM character models has remained the state of the art [Puigcerver (2017)]. One of the major improvements we have observed is related to image preprocessing and feature extraction (see Section 5.2). The current trend is to replace handcrafted features with features that are learned by deep convolutional neural networks from the data [Toledo *et al.* (2017)], leading to significant improvements of the overall recognition accuracy.

Considering the remaining errors, the results of automatic transcription are not directly usable for generating digital editions of historical documents. However, they offer the possibility to index the documents and make them searchable by content, allowing paleographers, historians, linguists, as well as other experts and the general public alike, to explore large collections of handwritten historical documents via text search.

References

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber (2006). Connectionist temporal classification: Labelling unsegmented sequential data with recurrent neural networks, in *23rd Int. Conf. on Machine Learning*, pp. 369–376.

Bunke, H. and Varga, T. (2007). Off-line Roman cursive handwriting recognition, in B. Chaudhuri (ed.), *Digital Document Processing* (Springer), pp. 165–173.

Fischer, A. (2012). *Handwriting Recognition in Historical Documents*, Ph.D. thesis, University of Bern, Switzerland.

Fischer, A., Frinken, V., Fornés, A., and Bunke, H. (2011a). Transcription alignment of latin manuscripts using hidden Markov models, in *Proc. 1st Int. Workshop on Historical Document Imaging and Processing*, pp. 29–36.

Fischer, A., Indermühle, E., Frinken, V., and Bunke, H. (2011b). HMM-based alignment of inaccurate transcriptions for historical documents, in *Proc. 11th Int. Conf. on Document Analysis and Recognition*, pp. 53–57.

Fischer, A., Keller, A., Frinken, V., and Bunke, H. (2010). HMM-based word spotting in handwritten documents using subword models, in *Proc. 20th Int. Conf. on Pattern Recognition*, pp. 3416–3419.

Fischer, A., Keller, A., Frinken, V., and Bunke, H. (2012). Lexicon-free handwritten word spotting using character HMMs, *Pattern Recognition Letters* **33**, 7, pp. 934–942.

Fischer, A., Wüthrich, M., Liwicki, M., Frinken, V., Bunke, H., Viehhauser, G., and Stolz, M. (2009). Automatic transcription of handwritten medieval documents, in *Proc. Int. Conf. on Virtual Systems and Multimedia*, pp. 137–142.

Frinken, V., Fischer, A., Manmatha, R., and Bunke, H. (2012). A novel word spotting method based on recurrent neural networks, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **34**, 2, pp. 211–224.

Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for improved unconstrained handwriting recognition, *IEEE Trans. PAMI* **31**, 5, pp. 855–868.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory, *Neural Computation* **9**, 8, pp. 1735—1780.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling, in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 181–184.

Kornfield, E. M., Manmatha, R., and Allan, J. (2004). Text alignment with handwritten documents, in *Proc. 1st Int. Workshop on Document Image Analysis for Libraries*, pp. 195–209.

Lorette, G. (1999). Handwriting recognition or reading? What is the situation at the dawn of the 3rd millenium? *Int. Journal on Document Analysis and Recognition* **2**, 1, pp. 2–12.

Manmatha, R., Han, C., and Riseman, E. (1996). Word spotting: A new approach to indexing handwriting, in *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, pp. 631—637.

Marti, U.-V. and Bunke, H. (2001). Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system, *International journal of Pattern Recognition and Artificial intelligence* **15**, 01, pp. 65–90.

Ploetz, T. and Fink, G. A. (2009). Markov models for offline handwriting recognition: A survey, *Int. Journal on Document Analysis and Recognition* **12**, 4, pp. 269–298.

Puigcerver, J. (2017). Are multidimensional recurrent layers really necessary for handwritten text recognition? in *Proc. 14th Int. Conf. on Document Analysis and Recognition*, pp. 67–72.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE* **77**, 2, pp. 257–285.

Rodriguez, J. and Perronnin, F. (2009). Handwritten word-spotting using hidden Markov models and universal vocabularies, *Pattern Recognition* **42**, 9, pp. 2106–2116.

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

Toledo, J. I., Dey, S., Fornés, A., and Lladós, J. (2017). Handwriting recognition by attribute embedding and recurrent neural networks, in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, Vol. 1 (IEEE), pp. 1038–1043.

Zamora-Martínez, F., Frinken, V., Espana-Boquera, S., Castro-Bleda, M. J., Fischer, A., and Bunke, H. (2014). Neural network language models for off-line handwriting recognition, *Pattern Recognition* **47**, 4, pp. 1642–1652.