The snowball principle for handwritten word-image retrieval

The importance of labelled data and humans in the loop

Jean-Paul van Oosten

The snowball principle for handwritten word-image retrieval

The importance of labelled data and humans in the loop

Jean-Paul van Oosten

Cover illustration by Bente van der Graaf
Printed by Proefschriftmaken (www.proefschriftmaken.nl)

This research was made possible thanks to the SNN project *Target*
This research was also in part made possible by the University of Groningen, Slimmer AI and the Human Interface Laboratory at Kyushu University
Special thanks to the National Archive for the use of the collection *Kabinet der Koningin*

rijksuniversiteit
groningen

The snowball principle for handwritten word-image retrieval

The importance of labelled data and humans in the loop

Proefschrift

ter verkrijging van de graad van doctor aan de
Rijksuniversiteit Groningen
op gezag van de
rector magnificus prof. dr. C. Wijmenga
en volgens besluit van het College voor Promoties.

De openbare verdediging zal plaatsvinden op

vrijdag 26 maart 2021 om 16.15 uur

door

Jean-Paul van Oosten

geboren op 2 september 1984
te Westvoorne

Promotor
Prof. dr. L.R.B. Schomaker

Beoordelingscommissie
Prof. dr. A.P.J. van den Bosch
Prof. dr. M. Biehl
Prof. dr. R.I. Ingold

CONTENTS

1

INTRODUCTION

1 FROM MONKS TO THE MONK SYSTEM

In the current computer-age, a large part of our communication is born digital. However, there are still large paper-based archives that contain handwritten materials such as letters, royal decrees, diaries and much more. These materials are usually historically relevant. For example, the important Qumran collection, also known as the Dead Sea scrolls, is a collection of handwritten texts that are among the oldest manuscripts included in the bible. A more recent, but also historically relevant example is a collection from the Dutch National Archive, the Cabinet of the King (van der Zant et al., 2009, 2008b): This collection contains Dutch royal decrees such as appointments of judges, rulings and laws from around the 1900s.

Archives that maintain collections such as the Cabinet of the King or the Qumran scrolls often have the important task of providing access to their collections. The historical relevance of the materials attracts many scholars and people from the general public interested in the contents of the documents. Because these archives are vast (the Cabinet of the King fills about 3km of shelf-space with handwritten documents), it is cumbersome to find information: An interested researcher usually has to consult multiple indices to find documents that are relevant to his or her query. The subject of this thesis is how to build a search engine for historical handwritten document collections and its building blocks. Using a search engine for handwritten pages reduces

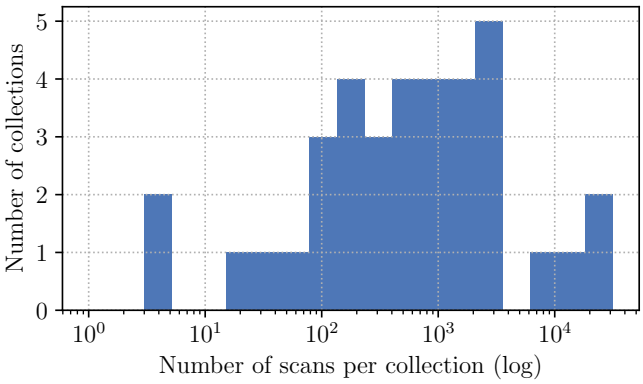


Figure 1.1: Histogram of number of scans per collection for 24 selected collections, with a total of 62408 scans in Monk (2017).

the amount of work needed to find a page containing relevant information. Moreover, the knowledge about the contents and how to read the material is preserved in the search engine itself—Knowledge that until now is mainly stored in the minds of historians and archivists.

Another important reason to study handwriting recognition is the task itself of reading historical documents. This is a challenge for human readers, let alone for machines: Unfamiliar scripts and unknown abbreviations and shorthand make it difficult to decipher the information that a document contains. The learning of reading skills is interesting to Artificial Intelligence researchers from two perspectives: First, the human intelligence perspective is interesting because it teaches about learning and pattern recognition in biology. Secondly, the machine intelligence perspective provides the challenge of getting a machine to read.

To study handwriting recognition from the machine intelligence perspective, the Monk system (Schomaker, 2016) was developed. At its core, Monk is a search engine for handwritten documents (van der Zant et al., 2009) that makes many collections in many different scripts, from many historical periods by many writers accessible: From the Cabinet of the King, to personal communication from the interbellum, medieval collections of jury verdicts,

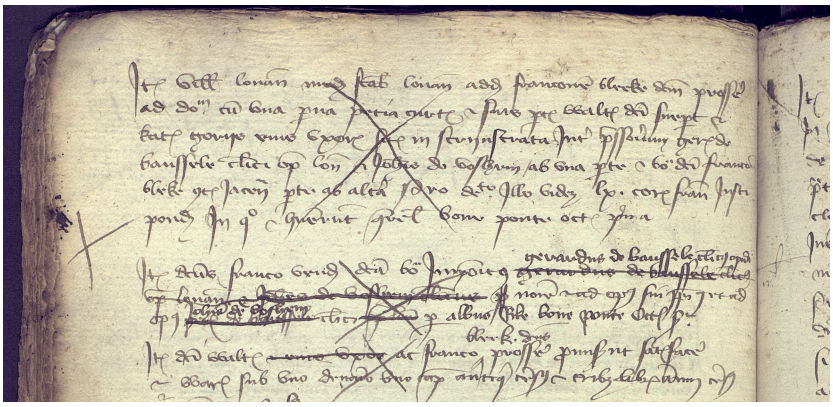


Figure 1.2: An example of a page from the Leuven alderman scroll collection (1421 A.D.). The alderman scrolls document the proceedings of law and the fines that citizens had to pay. When the fines were paid, and the case was closed, a mark was made through the record, indicating that the debt was settled. This example features some challenges a handwriting researcher has to face: abbreviations, strike troughs, writing between lines, ink smudges and old paper.

Chinese and Arabic letters as well as a collection of fragments from the Dead Sea scrolls. See Figure 1.1 for an illustration of the amount of scans available in Monk. These collections can be quite difficult to read properly, due to, e.g., degraded ink, abbreviations that are no longer in use or strike troughs. Figure 1.2 shows an example of such difficult material.

Unfortunately, handwriting recognition techniques are often tested on relatively clean datasets, such as the isolated digit collection MNIST (Collobert et al., 2006; Bhowmik et al., 2011)

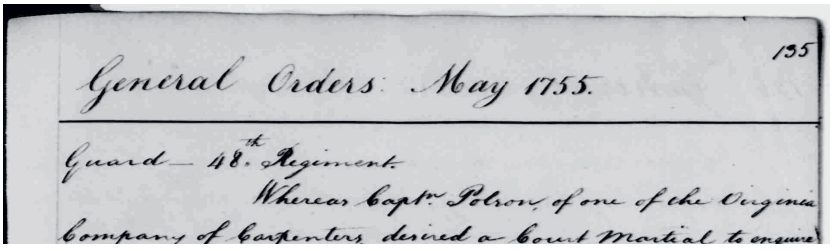


Figure 1.3: An example from the well-studied Washington collection. Please notice the crisp letters, consistent, clear writing style and quality of the background, which are in contrast with Figure 1.2

and the neatly written letters by president George Washington (see Figure 1.3 and Fischer et al., 2012; Wei et al., 2013). The application of techniques that work well for these academic datasets on the collections in Monk proved difficult. It was not straight-forward to reproduce the results that were reported in the literature. Also, for most methods an extensive amount of labelled data is necessary, which is unavailable for newly scanned historical documents. The problem of starting from scratch is called the bootstrapping problem. Solving this problem was one of the design goals of Monk. We will study bootstrapping and a number of other issues that we encountered while trying to reproduce the results from the literature and during the development of Monk.

A big issue that prevented a rapid growth of labelled data in the early stages of the Monk system was the slow progress of annotation. At first, some experiments were performed for line retrieval (Schomaker, 2007), but a text line is not a natural object for search. For these experiments, a line-based web-interface was used to annotate line by line, starting at the first page. This means that pages that have been completely annotated are also fully searchable. However, unseen pages cannot be used as input for the machine learning methods. This can be an issue for instance when writing styles change over time. Furthermore, this does not use the full potential that the computer has to offer. A switch was therefore made to an approach based on data-mining, where segmented words, presented in a hit-list interface, can be annotated throughout the entire collection.

By engaging the human annotators differently—by annotating through a hit list instead of transcribing text line by line—we realised that humans can be involved in machine-based handwriting recognition in different ways: 1. By developing the techniques that learn from observations (machine learning), 2. by designing feature extraction methods that transform the written words into mathematical vectors to be used by the machine learning, and 3. by providing the labels: the knowledge of which word is depicted on an image—the ground truth.

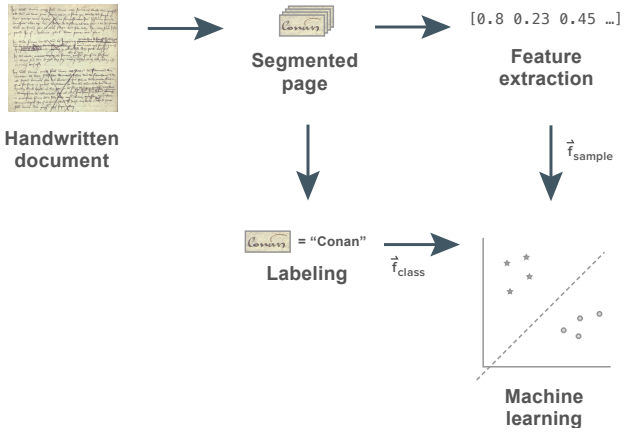


Figure 1.4: A common way of representing the handwriting recognition process. A handwritten document is segmented into lines and words. Each word is then pre-processed and transformed into a feature vector using a feature extraction method. Machine learning can be applied to these feature vectors. The ground truth for the machine learning methods is provided by the human labelling process. The segmentation and pre-processing steps are outside of the scope of this thesis.

In this thesis, we argue that the machine learning methods should not get the singular focus of the handwriting recognition research community. There is also a need to further develop the fields of labelling and feature design. The goal of this thesis is to study the assumptions in all three aspects of involvement such that we can improve the handwriting recognition process. The next section will give an overview of the topics that are studied in this thesis and show how the different aspects of human involvement are related to each other.

2 HUMAN INVOLVEMENT IN THE HANDWRITING RECOGNITION PIPELINE

Figure 1.4 shows the handwriting recognition process as it is frequently presented: As a pipeline. The first step is to pre-process a document and segment it into individual word images. These are then transformed into a feature vector: a robust representation suitable for numeric computation. Together with a dataset of

manually labelled word images, a machine learning method can be applied to learn how to classify unseen word images.

The first step of the pipeline, the pre-processing and segmentation, is important. Because this is the first step, all other steps can be affected by errors here (this is related to the concept "garbage in, garbage out"). There have been numerous studies on segmentation-free methods (Rothacker and Fink, 2015; Almazán et al., 2014; Lorigo and Govindaraju, 2006) to prevent errors in segmentation to have such a cascading effect. Recurrent neural networks such as LSTMs can operate without segmentation beforehand. However, even if the input can be processed without segmentation in the first step, a post-processing step that does the segmentation—usually in ASCII-space by specifying codes for blanks, line endings and paragraphs or by introducing blank tokens between repeated characters (Bluche et al., 2015; Hannun, 2017)—is still required.

A common argument against segmentation is overcommitment: The errors made in the segmentation step can not be corrected in the later stages of the pipeline. For image retrieval, overcommitment is not a big issue because of over-segmentation. This means that a word-image can have many overlapping word zone *candidates*, as shown in Figure 1.5. The assumption is that the correct word zone candidate will be ranked higher in a hit list because it is more *prototypical* (see also Chapter 4). Oversegmentation looks expensive, but the number of word zone candidates, typically in the dozens or a few hundreds, is much smaller than the total number of horizontal pixel positions along the x-axis, which is currently typically in the order of several thousands.

The main focus of this thesis will not be on segmentation and preprocessing, but on the other three steps in the pipeline: Machine learning, feature extraction and labelling. Specifically, we will discuss how we, as researchers, can be involved in the handwriting recognition process. In the design and development of a large, trainable retrieval engine for handwriting such as Monk, we stumbled upon several common assumptions that either proved to be misleading or required a twist in order to be useful for

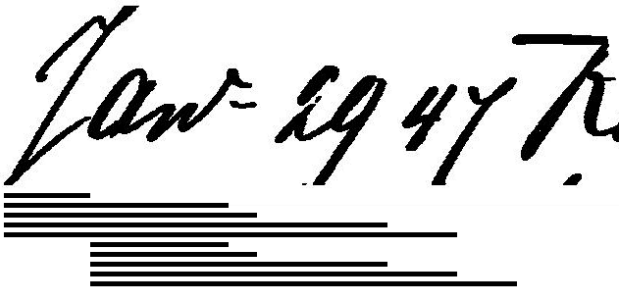


Figure 1.5: Monk uses over-segmentation for finding word zone candidates. Each line below the words indicates the width of a number of word zone candidates. The word zone candidate corresponding to the second line then shows the word "Jan", the abbreviation of "January". This word zone candidate will rank very high in the *"Jan" hit list*.

obtaining effective retrieval and recognition performances. These assumptions will be treated in the following sections.

2.1 Assumptions in machine learning algorithms

The first part of the pipeline that we will examine in this thesis is the machine learning process. Specifically, we will look at common assumptions in hidden Markov models (HMMs) and support vector machines (SVMs). Even though they have gained considerable attention, artificial neural networks and deep learning are not studied in depth. HMMs and SVMs are interesting because they are strong classifiers and have been very popular in the handwriting recognition field.

Training HMMs is a non-trivial problem because observations only give indirect evidence of the hidden states. This means there are some—well-known—issues with training HMMs. Most notably, the algorithm will optimize towards a local optimum (a sub-optimal solution), not necessarily to the *global* optimum. Unfortunately, the phenomenon of local optima itself is not the subject of many studies, while this is such an important property of the Baum-Welch training method.

We study the local optima to get a better understanding of how and when they are reached. It is interesting to get a global perspective on the training of HMMs because this will give more insights into how models behave. We compare models with known parameters to trained models. The assumptions that are examined are related to this issue: Are models that are close to the global optimum better than models that are further away?

2.2 Assumptions on features

Hidden Markov models learn both the structure and the observations that provide the partial evidence of the structure. The observations are usually features that are extracted from, in our case, images of handwritten words. The assumption related to features that we test in this thesis is that the underlying structure is considered to be more important than the observations.

We test this assumption by answering two questions. First, can we find the structure we know is present in the data by only using observations? And secondly, can we still perform classification

when we remove temporal structure from a model? Again we use models and data with known parameters for comparison with trained models and calculation of the classification accuracy of the trained models.

We are interested in seeing the influence the transition probabilities have on the classification accuracy, and how important the observed features are. This insight is useful in the discussion of where to focus our engineering effort: On the part that learns the hidden parameters or the observations, i.e., on the Machine Learning part of the pipeline, or on the feature extraction.

2.3 **Assumptions on the origin and availability of labels**

The final assumptions that we will study are related to labels. Labels provide the ground truth that allows the machine learning to update the parameters and generalise to unseen data. There are two assumptions to be studied related to the labels. First, there is the assumption that there is a dataset available that is properly labelled. Generally, researchers use existing, academic datasets or collect labels outside of the recognition process. Secondly, there is the assumption that the handwriting recognition process is static, as described in Figure 1.4. Instead, we consider the process to be a loop like in Figure 1.6 instead, incorporating all elements in a continuous learning cycle.

This loop is facilitated by using a hit list interface. Word images are divided into different classes and then ranked such that the images at the top are most likely to be correct and useful for the labelling process. A human annotator can then easily select the word images that are correctly labelled and update the label store with many labels at once. The classifier and ranking method are then retrained which updates the hit lists for the annotator, allowing for even more labels to be added to the system, and yielding a snowball effect.

The final assumption discussed in this thesis is then the assumption that both classifying and ranking should be done by the same mechanism. We believe that there are actually two func-

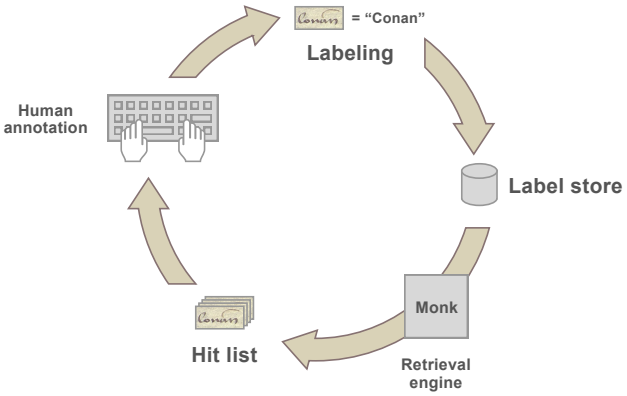


Figure 1.6: Overview of how we consider the process to be a loop instead of a static pipeline. The hit list is an important concept that contains a list of retrieved images for a certain class that can be easily and quickly labelled by a human annotator. Each update is used by the retrieval engine to update the classification and ranking methods, which provides better hit lists. This way, a snowball effect can occur.

tions to be optimised. This is interesting because in earlier work, using a single method for both functions yielded unintuitive results at the top of the hit lists. Having a well-ranked hit list is essential for achieving a snowball effect.

3 RESEARCH QUESTIONS

The main focus of this thesis is the relationship between the three aspects of human involvement discussed above. Because the handwriting recognition community has a strong focus on the machine learning aspect of handwriting recognition, it is argued in this thesis that especially the labelling part of the handwriting recognition loop has been neglected. The main, general research question is related to this argument.

General Research Question

Where can we have the most impact on the results of a search engine for historical handwritten documents? Should the attention be focused on improving the machine learning methods, the feature engineering methods or the labelling methods? To answer this question, we will look at the individual parts of the handwriting recognition process. For each aspect we consider the more concrete questions.

Research questions related to Machine Learning

Since handwriting concerns patterns of variable width (analogous to variable duration in speech) a predominant model consisted of hidden Markov models, that were well studied in the literature since 1988. However, there are a few assumptions to be aware of when using HMMs. The models are trained using the Baum-Welch algorithm, which is a type of Expectation Maximisation (EM) algorithm. These types of algorithms are used when certain parameters of a model are not directly observable. This is exactly the case in HMMs: The observations only give a partial evidence of the underlying structure. Of course, this is a difficult problem to solve: How can we model a set of parameters for which we only have indirect evidence?

The Baum-Welch training algorithm works by randomly creating a model (the initial model) and updating it iteratively. After a number of iterations, or until the updates have become too small, this process stops. The resulting model (the learned model)

can be stuck in a local optimum. A relatively straight-forward method to deal with local optima is to train multiple models and choose the highest performing one. Even though there have been some studies on how to converge to the global optimum (For example, Lee and Park, 2006; Siddiqi et al., 2007, but see Chapter 2 for a more in-depth discussion), this "restarting" scheme is still a popular method to deal with local optima.

The following questions are intended to study the phenomenon of local optima:

- Is there a relation between the distance from a learned model to the global optimum and the final performance of the learned model?
- Is there a relation between the distance of the initial, random model and the distance of the final, trained model?

The assumptions that are questioned here are (a) that the closer a model is to the global optimum, the better a model should perform, and (b) that the closer the initial, random model is to the global optimum, the closer to the global optimum that model will end up after training.

Research questions related to Feature Extraction

Since hidden Markov models are aimed at modelling time series, an important component of such models is the transition matrix. This matrix defines the probability for switching from one state to another and is therefore the temporal part of the model. However, typically, the underlying temporal information is not directly observable: The observations only give partial evidence of the underlying state.

The general structure of the transition probabilities is called the topology and indicates which state transitions are allowed and which are not. For example, the Bakis topology only allows transitions from a state S_j to state S_{j+1} or to itself, as shown by the topology diagram in Figure 1.7. Of course, from the observations we cannot directly determine whether the structure

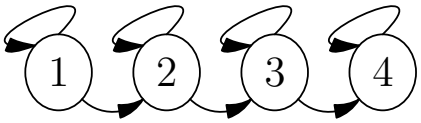


Figure 1.7: Illustration of the Bakis topology in hidden Markov modelling. The arrows indicate the possible transitions between the numbered states (it does not show the actual probability of a transition). See Chapter 3 for more details.

is organised in a Bakis topology or something else, but it is very common to force the models in handwriting recognition to have such a topology (Zimmermann and Bunke, 2002; Bunke et al., 1995; Britto et al., 2001).

The observation probability distributions provide the other important part of the models. They model the probabilities that a certain feature will be observed in a certain state. The observations are features that are extracted from the raw image pixels—usually an abstraction such that they are robust representations of the (sub-) characters or words.

The questions related to Feature Extraction are intended to study the relation between transition and observation probabilities:

- Since the observations only provide partial evidence for the underlying states, can we learn the topology (the general structure) of the transition matrix from observations alone?
- Since HMMs model time series, and the temporal information is such a central part of the models, is classification performance reduced when temporal information is removed from hidden Markov models?

Research questions related to Labelling

The Monk system implements both classification and searching methods to provide access to historical documents. However, the use of the support vector machine (SVM), a strong classifier, for ranking images in search results, showed unintuitive results in the top of the search results. This was surprising because

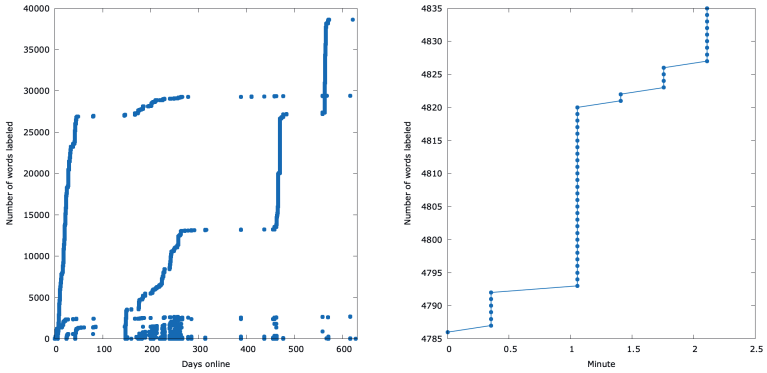


Figure 1.8: Example of the snowball effect, marked by large jumps in number of labels for a collection in Monk. Each data point represents one label being added to the system, showing the time since this collection has first been online and the number of images labelled at that point in time. (a) A global view of almost two years of labelling activity in a single collection. (b) A close-up at a point in time where, with a single action in the interface, many labels are generated (shown at roughly the 1 minute mark).

the SVM generally reports high accuracy on image classification tasks. Since we also use these lists to solicit valuable feedback from the users, the top of the list is very important: With one press of a button, the top results can easily be confirmed to be correct.

This hit-list based approach to labelling enables the snowball effect. By providing more labels, Monk increases its performance, which in turn generates a better hit list that makes it easier to label more images. This effect creates jumps in the number of labelled images, which remind us of phase transitions in physical systems. Figure 1.8 shows an example of a collection in the Monk system with these jumps in the number of labels. This effect is not possible in a left-to-right annotation process, because there is no mechanism to use feedback to speed up the labelling process.

The following questions were raised while observing the counter-intuitive results in the top of hit lists, and are related to how to effectively gather good quality labels and achieve a snowball effect:

- Why are SVMs not suited for ranking handwritten word images?
- How do we get intuitive hit lists?

4 OUTLINE OF THE THESIS

This thesis has three chapters related to answering the research questions posed in the previous section. In Chapter 2 we address the machine learning questions by studying HMMs, while Chapter 3 discusses HMMs from a feature perspective. Finally, Chapter 4 brings these subjects together and adds the labelling perspective to answer the question of how to get intuitive hit lists.

The last chapter, Chapter 5, concludes the thesis with a summary of all the findings and a discussion of the main subject of the thesis: loops instead of pipelines. We also discuss the connection to methods such as active learning and deep learning.

2

EXAMINING COMMON ASSUMPTIONS ABOUT THE CONVERGENCE OF THE BAUM-WELCH TRAINING ALGORITHM FOR HIDDEN MARKOV MODELS

Abstract

Hidden Markov models (HMMs) model time series and have many applications. However, reliably training an HMM has proven to be non-trivial. One of the challenges is that the Baum-Welch algorithm finds a local, instead of a global optimum. In this paper, we study the conditions in which the local optimisation algorithm finds an optimum using global information from artificially generated models and data. Using a known global optimum, we can look at the distance and performance, in terms of log-likelihood, of a model at any point during or after training. We can then test a number of common assumptions, such as whether a model that is close to the global optimum actually has a good performance. We find that most common assumptions do not hold and question whether the optimisation criterion used by Baum-Welch is the most effective. Finally, we offer a number of considerations for future work that aim to help find a better optimum than using log-likelihood as an optimisation criterion alone.

1 INTRODUCTION

Hidden Markov models (HMMs) have been used extensively to model time series in applications of speech recognition (Rabiner, 1989), handwriting recognition (Bunke et al., 1995; Plötz and

Fink, 2009) and gene sequence segmentation (Eddy, 1998). However, non-Markov, neural methods for sequence classification are gaining considerable momentum in the handwriting recognition and speech recognition fields. The deep learning method Bidirectional Long Short Term Memory networks (BLSTM) (Graves et al., 2009; Frinken et al., 2012) delivers very promising results, at the cost of studies involving HMMs. HMMs still have an active following, and it is interesting to look at some of the assumptions that are commonly associated with the training algorithms for HMMs, and why this technique may be losing popularity.

In a previous study (van Oosten and Schomaker, 2014a), we investigated the role of the transition probabilities in hidden Markov modeling and showed that it is hard to learn the correct, known properties, such as the topology of the transition matrix, of a Markov process, from artificially-generated sequences of observations. In the current study we will apply similar methods to look at convergence of HMMs. We are particularly interested in the conditions where models seem to converge to a local optimum, as opposed to the global optimum. This is an interesting topic because generally, it is assumed that the closer the trained models are to the global optimum, the better the performance (usually measured by log-likelihood or classification accuracy).

The canonical training methods for HMMs are Baum-Welch (Rabiner, 1989) and Viterbi or Segmented k-means training (Rabiner et al., 1986). Both methods are expectation maximization (EM) techniques and the final solution is highly dependent of the initialisation. That means that the starting point, i.e., the set of initial model parameters — often chosen at random or by using a clustering method for parts of the model (Bhowmik et al., 2011) — determines whether or not a model will end up in a local or a global optimum. The only guarantee that EM gives is that the likelihood of a model does not decrease during training, i.e., the likelihood of modelling the training samples increases or stays the same: $P(O|\lambda_{i+1}) \geq P(O|\lambda_i)$, where λ_i is the model at iteration i and O is the training set of observation sequences.

When the solution landscape is not smooth and having a single optimum, which is almost always the case for non-trivial HMMs,

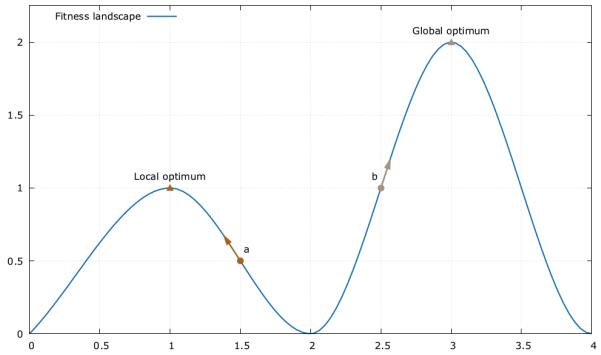


Figure 2.1: The final optimum that a model will reach, when trained with Baum-Welch, depends on the initialisation. Model a will converge to the local optimum at the left, while model b will converge to the global optimum on the right. The optimum is in this case defined as the maximum. Conversely, in gradient descent one would *minimise* the loss function, but the problem of the presence of several local and a single best solution is similar, there.

as opposed to, e.g., the convex SVM loss function, and if the initial model is not located on a direct path to the global optimum, a local optimum will be reached. See Figure 2.1 for a schematic representation of the difference between local and global optima: Trajectory a leads to a local optimum, whereas trajectory b leads to the global maximum.

There are a number of studies that investigate achieving a better, or even a global optimum. These studies can be divided into two categories: I) Meta search algorithms, that employ global search over a local search method; and II) Modifications to the EM algorithm itself.

The first category concerns the meta search algorithms. A straightforward method is the 'restart' method: train a number of models and select the best one. However, these restarts can be costly and do not guarantee convergence to the global optimum. Zhang et al. (2008) also employ restarts, but devised a method that quickly prunes unsuccessful attempts. A related method is described by Lee and Park (2006) which uses simulated annealing, together with Baum-Welch, to optimize the models. This method also restarts the EM-algorithm, but instead of starting from a new

randomly initialised model, selects the new initial parameters using the simulated annealing method. Another method that uses restarts is described by Gil and Williams (2009). This method clusters models by their parameters and selects a new model to optimise by finding a point outside a cluster.

The second type of solutions covers modifications to the training algorithm itself. A study by Farago and Lugosi (1989) explains how to use an alternative to Viterbi training (i.e., only updating the most likely path) to reach a global optimum in the restricted case of left-right models. Siddiqi et al. (2007) show a method that not only avoids local optima, but also searches for the optimal number of states by 'splitting' states with very similar observation probabilities. There are also spectral techniques, such as the technique described by Hsu et al. (2012), for finding the true parameters under certain conditions. These techniques do not necessarily find the traditional transition and observation probabilities, but rather a representation that is closely related to them.

Since the prevailing method of learning the parameters of hidden Markov models is still the Baum-Welch method, we are interested in the conditions in which this method converges to local optima. This interest grew from the observation in a previous study (van Oosten and Schomaker, 2014a) that it is hard to find the topology of the underlying transition matrix of a known model. We found this by comparing a trained model to the model that generated the observation sequences.

In this paper we will take the method of generating both model and data a step further and introduce a framework to investigate the inner workings of the Baum-Welch algorithm from a convergence perspective. We will do this by artificially generating sequences of discrete tokens[1] and comparing the trained model with the original model that was used to generate the sequences. As mentioned earlier, the general assumption is that finding the global optimum is good for performance. We will test this in Section 3 by observing whether there is a relation between the

1 We use discrete tokens for simplicity. In principle, the methods can also be applied to continuous-density HMMs.

distance to the global optimum and performance, in this case measured by log-likelihood. We expect to see an increase in log-likelihood when the distance to the global optimum is smaller. We then also expect to see that distance decreases during the training of a model, because if the likelihood increases, which is what Baum-Welch does, the distance should go down.

Another assumption that will be tested is whether the global optimum is easy to find when the distance is already small. This means that there is a relation between the distances of the initial model and the trained model to the global optimum (see Section 4). Related to this question is whether the global optimum can be reliably found when parts of the model are already known. For instance, in gene segmentation (Azad and Borodovsky, 2004), the transition matrix can usually be estimated directly from labelled transitions. We expect the distances to the global optimum to be very low if we do not need to estimate certain parts of the model (Section 5).

Finally, in Section 6 we touch on some ideas that still need to be developed further. We will investigate how much global information is actually needed to guide the local process to a global optimum. We believe that the methods of inspecting the inner workings of the Baum-Welch algorithm, a local search algorithm, by using global information are very promising.

2 METHOD

To investigate the convergence towards a global optimum, we need to know what the global optimum is. As we mentioned in the introduction, we will generate artificial data by running the algorithm by Rabiner (1989) using randomly chosen models. When we train a model on the data generated from a model, we call the generating model, λ_O, the original model and consider it to be the global optimum for the Baum-Welch training algorithm. We can compare the newly trained model, λ_T, to the original model. Figure 2.2 shows this procedure schematically.

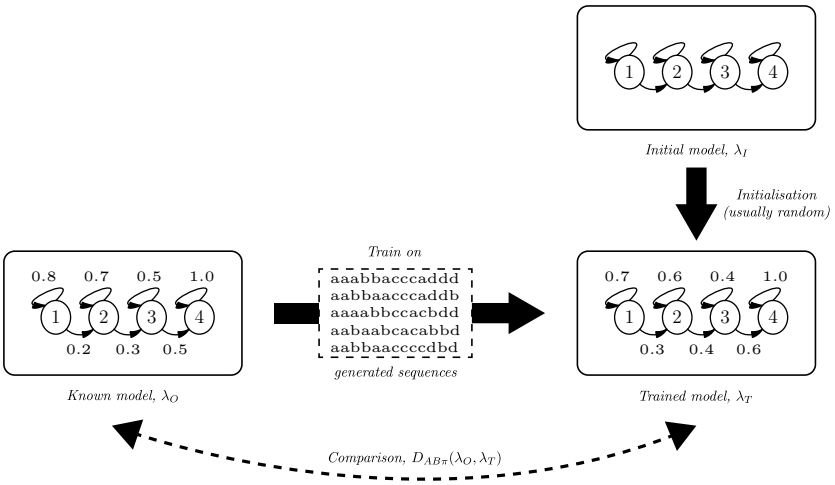


Figure 2.2: Method to train and compare models: By generating data from a model with known parameters, and training on that data, we can compare the model with known parameters with the newly trained model. Model initialisation is usually done by randomly generating model parameters, but some experiments have a different method of initialisation.

Since we now define the original, generating model to be the global optimum, we need to know whether another model is 'close' to it. Rabiner (1989) also describes a distance measure $D(\lambda_p, \lambda_q)$, that is "a measure of how well model λ_p matches observations generated by model λ_q,relative to how well model λ_q matches observations generated by itself."

However, using a distance measure based on generated data by either model, we can not measure the distance of any part of a model $\lambda = (A, B, \pi)$ separately. More importantly, Rabiner's distance measure uses log-likelihood: a measure that is used to optimize the models in the first place. We therefore propose to use a different distance measure that is based on the χ^2 distance. The distance metric is the sum of all the χ^2 distances over all rows of the three matrices of a model λ, and we can selectively choose what distances to use in our comparison: $D_A(\lambda_p, \lambda_q)$ for comparing only transition matrices, $D_B(\lambda_p, \lambda_q)$ for only comparing observation matrices, $D_\pi(\lambda_p, \lambda_q)$ for the initial state distributions,

and any sum of these three distances. We define $D_A(\lambda_p, \lambda_q)$ as follows:

$$D_A(\lambda_p, \lambda_q) = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{(A_{ij}^p - A_{ij}^q)^2}{A_{ij}^p + A_{ij}^q}$$

where A_{ij}^p denotes the transition probability from state i to state j in model λ_p. $D_B(\lambda_p, \lambda_q)$ and $D_\pi(\lambda_p, \lambda_q)$ are defined similarly. We also define the sum of the distances of A, B and π as $D_{AB\pi}(\lambda_p, \lambda_q) = D_A(\lambda_p, \lambda_q) + D_B(\lambda_p, \lambda_q) + D_\pi(\lambda_p, \lambda_q)$.

The log-likelihood measure is what the Baum-Welch training algorithm optimises, and is therefore usually seen as the performance metric for a single model. When doing a simple classification task to decide which model is the most likely model to have generated a certain sequence, the common method is to return the model with the highest log-likelihood:

$$class(\vec{o}) = \underset{i}{\operatorname{argmax}}[\log \mathcal{L}(\vec{o}|\lambda_i)]$$

In a Bayesian sense, the actual log-likelihood is now less interesting, as long as the 'correct' model has the highest log-likelihood. It can be argued, however, that the goal of the training should be to reach the global optimum, provided that the assumption holds that a small distance to the global optimum leads to a high log-likelihood. In most experiments in this study, we use the average log-likelihood on a dataset as a measure of performance. That is: we compute the log-likelihood for all observation sequences in that dataset and use their average value as a measure of performance.

The models in this study have $N = 20$ states and a set of $M = 20$ discrete observable tokens. We generated 1500 sequences per model with a length of $|\vec{o}| = 50$ tokens. The sequences were of fixed length to reduce complexity and prevent potential problems with variable sequence length. The length of these sequences was chosen to be roughly the average sequence length of our previous experiments with sliding windows over words in our handwriting recognition system.

A dataset of 5000 random model initialisations will be used frequently in this study. Training new models starting at these

initialisations takes roughly 8 minutes per model. Distributing the training of 5000 models over 8 cores takes about 3.5 days to complete.

3 RELATION BETWEEN DISTANCE TO THE GLOBAL OPTIMUM AND PERFORMANCE IN TERMS OF MAXIMUM LIKELIHOOD

The first assumption tested in this section is that models that are close to the global optimum also perform better. We use the average log-likelihood on the training set as a measure of performance. When measuring the performance of a single model, we cannot use accuracy since there are no models to compare to.

We randomly initialised 5000 models and trained them on a single dataset. These models then all represent different starting points in the 'fitness' landscape. The dataset of 1500 instances was generated by using the Rabiner algorithm on model λ_O and used to train all models. λ_O has a transition matrix with a Bakis topology. To prevent also having to estimate the topology from scratch, the random models were limited to a Bakis model as well. Note that this restriction should make the model estimation easier than is the case for arbitrary topologies.

After training the 5000 randomly initialised models on the training set, we plot $D_{AB\pi}(\lambda_T, \lambda_O)$, the distance of the trained model to the original, against the average log-likelihood. Figure 2.3 shows the results of this experiment. For comparison, the average log-likelihood of the generating model attained the value -143.27.

We can see that there is no clear relationship between distance and performance. However, we can see that the few models that do end up at a very low distance from the original model λ_O, tend to have a better performance. That does not mean that only models with a low distance show a high performance (Figure 2.3, upper left data points). On the contrary: the models with an average log-likelihood of approximately -143.2 show a wide range of distances of 1.8 to 6.5.

Figure 2.3: Scatter plot of obtained log-likelihood values on a training set with respect to the measured χ^2 distance between the true (original) model and many obtained models (N=5000) that were trained from random initial conditions. Log-likelihood for the generating model is −143.27. The scatter plot appears to indicate that there is a wide dispersion of solutions, both in terms of log-likelihood and model distance. Also, in log-likelihood space there appear to be 'plateaus' of solutions, with a wide variation in the distance between trained model and original.

In the remainder of the paper we will address a series of concrete **Findings**, that are based on our empirical work with the simulations described in this study. These simulations are inspired by our work in handwriting recognition.

Finding 1: The relation between distance to the original model and the final performance measure for HMM training, i.e., the maximum likelihood, is not clear. This means that distance is a poor predictor of (best) ML estimate and vice versa.

There is another assumption that is related to the question whether distance is a good predictor of performance: if the performance during training goes up, the distance to the global optimum should go down. It is not unreasonable to assume this if there is a clear relation between distance and performance, because it would mean that a better performance would lead to models closer to the global optimum.

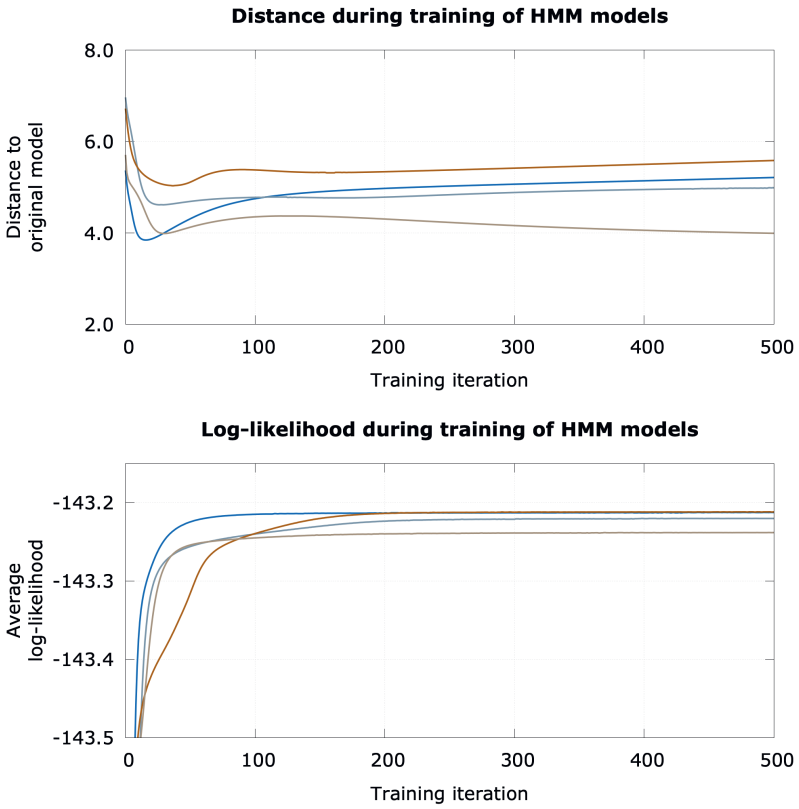


Figure 2.4: Distance and likelihood over iterations of Baum-Welch training, for four different random initialisations. The top figure shows the distance of the model in training while the bottom figure shows the average log-likelihood (the actual metric that is being optimized) on the training set.

This can be tested by plotting the distance and performance against the training iteration. We randomly selected four models from the 5000 models from the previous experiment, and plotted the distances and average likelihood in Figure 2.4.

We can see that the model distance to the original model λ_O, after a small dip, actually goes up again. Depending on how long we let the models train, the distance might even end up larger than at the very first training iteration). At the same time, we see that the average log-likelihood shows a common pattern of quick

improvements and then levelling off to an asymptote, reaching more or less the same log-likelihood for all four models.

Finding 2: The distance of the parameters of a model to the global optimum does not necessarily go down during training, in contrast to the log-likelihood, that increases to an asymptote as a property of the Baum-Welch algorithm.

It should be noted that the implementation of the Baum-Welch algorithm in this study was evaluated against other implementations, without revealing significant differences in an earlier study (van Oosten and Schomaker, 2014a).

4 RELATION BETWEEN INITIAL AND TRAINED DISTANCE TO THE GLOBAL OPTIMUM

The final optimum that an EM algorithm reaches is very dependent of the initial start condition. In this section, we test the assumption that the closer we are to the global optimum (i.e., the smaller the distance from a model to λ_O), the easier it is to find the global optimum. In other words, if the assumption is valid, we expect there to be a relation between $D_{AB\pi}(\lambda_I, \lambda_O)$ and $D_{AB\pi}(\lambda_T, \lambda_O)$.

In this section we are looking at the same 5000 models that were trained in the previous experiment, but now from the perspective of the assumption that the closer we are to the global optimum initially, the closer the final trained model will be. However, there is a model that is even closer to the global optimum than any of the random initialisations: the original model λ_O, which obviously has a distance of 0.

We initialise the Baum-Welch training with the original model (i.e., $\lambda_I \leftarrow \lambda_O$) and train on the data generated using λ_O until we reach a convergence-criterion[2]. It is interesting to note that it took the Baum-Welch algorithm around 1200 iterations to reach

2 In this case, a very strict criterion of 50 iterations with a change in average log-likelihood of smaller than 10^{-12}.

convergence. The final distance $D_{AB\pi}(\lambda_T, \lambda_O)$, averaged over ten different models, is 0.43.

Note that this means that the training makes the model drift *away* from the original model. This can be explained by the fact that we do not have an infinitely large dataset. The experiments are performed with 1500 sequences, but this is still only a limited view on all 310 parameters for this model (10×10 state transitions, 10×20 observation probabilities and 10 initial state probabilities). The view is limited because the observations only give partial evidence for the underlying states. Getting a perfectly trained model λ_T would require an infinitely large number of generated sequences, and thus an infinite amount of training time.

Finding 3: When initialised perfectly at the global optimum, the final model will have drifted away a bit, but still ends up very close to the global optimum.

Returning to the 5000 models that we trained, we expect that when running Baum-Welch algorithm on a model with a small initial distance to the original model will lead to a final trained model with a smaller distance to the original model.

Figure 2.5 shows the results of plotting these distances against each other, using densities. Using the diagonal line, we can see that all the models that end up above the line increased in distance, while everything below it lowered the distance to the original model. We can see here that even though the strict majority of the models ended up (78%) below the line, there is still a large number of models that moved away from the global optimum. We can also see that there is not a very clear relation between the two distance measures. On average, the Baum-Welch training will have a slight beneficial effect, by improving the distance by -1.41.

Finding 4: $D_{AB\pi}(\lambda_i, \lambda_O)$, the distance of the initial model to the original model is a bad predictor for $D_{AB\pi}(\lambda_T, \lambda_O)$, the distance of the trained model.

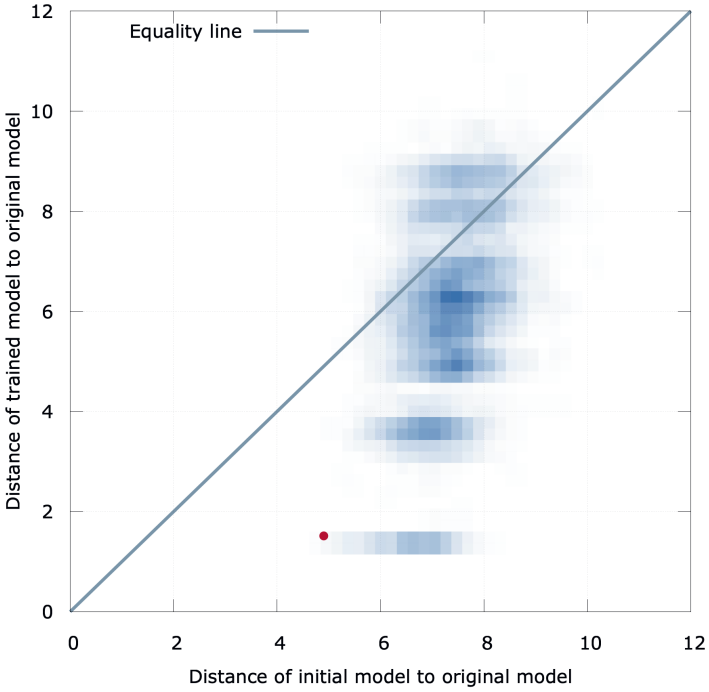


Figure 2.5: Density plot of the distance of the initial model λ_I to the original model λ_O versus the distance of the trained model λ_T. The diagonal equality line shows that models that fall above this line have moved away from the global optimum, while models below the line have improved their distance. 3924 models fall below the equality line, 1076 models above it. The red dot shows the model with the lowest distance. The shape of the densities shows no clear relationship between initial model distance and final trained distance. A histogram of 50×50 model distances was used to compute the densities

Another way to look at the relation between initialisation and the found optimum is by plotting the distance during training. Figure 2.6 shows the trajectories of four models over time, represented by the distances of transition and observation probabilities. The initial model is indicated in the figure by a circle, while the final model is indicated by a triangle. The figure clearly shows the sub-optimal path through the D_A versus D_B landscape.

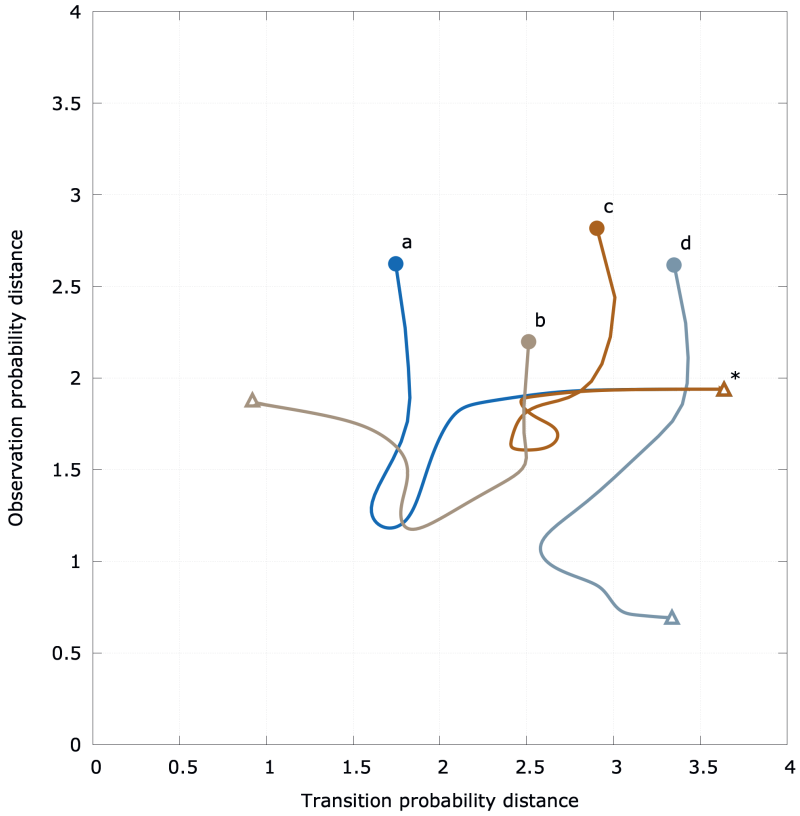


Figure 2.6: Trajectories of four random models during training, represented by the distances between in-training model and original: transition probabilities versus observation probabilities (i.e., $D_A(\lambda_i, \lambda_O)$ versus $D_B(\lambda_i, \lambda_O)$, where λ_i is the model at iteration i). The trajectories start at the circle and end at the triangle. They are haphazard, as opposed to converging to the lower left corner.

Trajectory 'a' starts promising, but suddenly deteriorates in transition probability distance. The trajectory migrates to the strange attractor indicated by '*'. Trajectory 'b' is reasonable: the distance of the transition matrix improves a lot, and the distance of the observation probabilities improve slightly as well. Trajectory 'c' is more chaotic and, like 'a', also ends up in the attractor indicated by '*'. Trajectory 'd' has, at the end of the training, not changed much when looking at the transition probabilities, but did improve considerably in observation probability distance.

It is interesting that the distance of the observation probabilities seems to be optimised first, as indicated by the initial downward trajectory. Even though the distances seem to be minimized in the first couple of training iterations, the trajectory moves away from the desired $(0, 0)$ point later in the training, just like we saw in Figure 2.4. This leads us to question whether the Baum-Welch optimization criterion, i.e., maximizing the log-likelihood, is the most optimal criterion.

5 TRAINING WITH PARTIALLY KNOWN MODELS

In certain applications some information can be easily inferred from the data. As mentioned in the introduction, the transition probabilities for models in gene segmentation can be estimated very reliably by using maximum likelihood estimation (Azad and Borodovsky, 2004). This usually means that the actual transitions are labelled in the training dataset. In other applications such as handwriting and speech recognition this information is not readily available.

In (van Oosten and Schomaker, 2014a), we examined the importance of a correctly estimated transition matrix. In this section, we will examine a related question: How closely can we get to the global optimum if certain properties are fixed to the known properties from the original model.

To answer this question we will copy the probabilities that we want to fix from the original model. This is also called 'clamping', from the idea of a voltage clamp in electrophysiology. We will look at two experiments: in the first experiment the transition probabilities are fixed while the observation probabilities will be trained. The second experiment will clamp the observation probabilities, and while keeping these fixed train the transition probabilities. We will then look at the final distances to the original model.

Performing these experiments, we found that all models converge to a model that has a very small distance to the original model. Table 2.1 shows the distances for the experiments described in

Transition prob.	Observation prob.	$D_{AB\pi}(\lambda_O)$
fixed	trained	0.26
trained	fixed	0.08

Table 2.1: Results of having certain properties of the model already known. The numbers are averages over 10 different models. 'Fixed' means that the model took either the transition or observation probabilities and these were not updated during training. 'Trained' means that the probabilities were randomly initialised and updated during training. The low distance to the original model when keeping the observation probabilities fixed seems to indicate that the observation probabilities are more important than the transition probabilities.

this section. We believe these numbers are indicative of the distances one could expect when reaching a global maximum because we made the task very easy for the algorithm to solve.

For comparison, the average distance of the transition matrices of all 5000 models is $\overline{D_A(\lambda_T, \lambda_O)} = 3.08 \pm 1.37$, and the average distance of the observation probabilities is $\overline{D_B(\lambda_T, \lambda_O)} = 2.01 \pm 0.89$. Please note also that when keeping the observation probabilities fixed, the total distance $D_{AB\pi}(\lambda_T, \lambda_O)$ is even lower than half the distance of models that had a clamped transition matrix. This is interesting because it indicates the importance of the observation probabilities.

Finding 5: Keeping either the transition or observation probabilities fixed makes the Baum-Welch algorithm converge very close to the global optimum.

6 IMPLICATIONS

In the previous sections of this report, we have looked at the distance from a trained model to the original model with the purpose of 'debugging' the Baum-Welch process – to see when it converges to a global optimum. All the experiments seem to indicate that there are configurations of an initial model that do converge to a model close to the data-generating model. However, the distance does not seem to be a very good predictor for either

average log-likelihood or the distance of the final model to the global optimum, the original model λ_O.

It is difficult to gauge from distance alone what the optimal initial position is. It would be useful to have the information on whether a model is moving in the right direction at training time, but –alas– the 'original' model for all practical purposes of HMMs is not available. If we could push a model in the right direction, this might give the Baum-Welch algorithm enough information to find an attractor close to the global optimum. In this section we will look at how much of a push the training algorithm actually needs and what we can do with that.

6.1 How much do we need to 'push' a model in the right direction?

For the purpose of pushing the Baum-Welch algorithm in the right direction, we will modify the updates to the parameters by using the known, original model. This means of course that this method cannot be used in practice, but it will give us some insight into how much the algorithm needs to be shown the correct direction.

We introduce a mix-factor α here. During the update to the parameters of the model, the mix-factor modifies the standard Baum-Welch update to include some knowledge of the original model. The new update rule is formulated as follows:

$$\lambda_{i+1} = \lambda_i + \Delta P$$

where

$$\Delta P = (1 - \alpha)\Delta P_{\mathrm{BW}} + \alpha \Delta P_{\mathrm{T}}$$

with ΔP_{BW} being the update that would be performed by using Baum-Welch alone, and ΔP_{T} is the difference with the true, original model, and thus the delta required to get to the true model in a single step. With the α parameter, we can therefore tweak how much of the true model will be added to the current model, and thus see how much we need to force the update in the right direction to find the global optimum. Setting α to 0, the update rule would be identical to the standard Baum-Welch update rule.

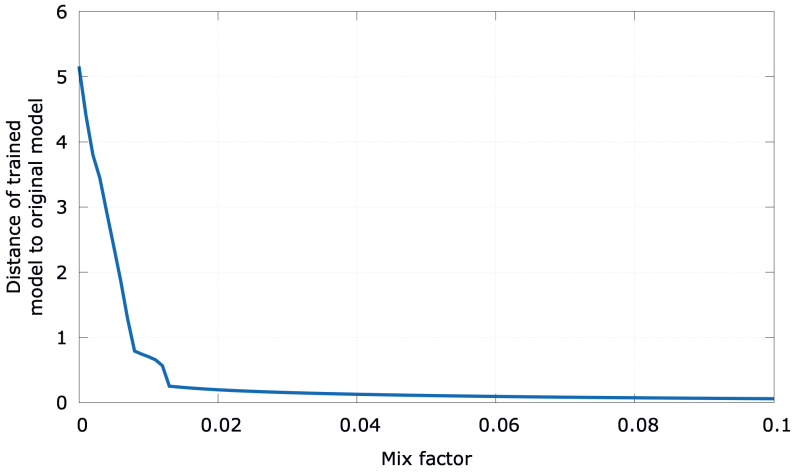


Figure 2.7: Results of the experiments with a mix-factor α between 0 and 0.1 with 0.001 increments, each data point is the average distance to the global optimum of 100 different initialisations. The 100 different initialisations are shared between the different α values, so the initialisation 0 for $\alpha = 0.01$ is the same as initialisation 0 for $\alpha = 0.02$. The graph shows that a mix of 99% Baum-Welch and ~ 1% additional information leads to a desired low distance.

It is important to note that for $\alpha > 0$, the Baum-Welch guarantee $P(O|\lambda_{i+1}) \geq P(O|\lambda_i)$ no longer holds. This is a necessity of our goal: to push Baum-Welch out of the path towards a local optimum. This does not mean that a push out of that path will always result in a worse log-likelihood, but the fundamental guarantee no longer applies.

Starting at $\alpha = 0$, with increments of 0.001, we trained 100 different models per value of α and took the average distance to the global optimum. The results are shown in Figure 2.7.

We can see that at $\alpha \approx 0.015$, the distance has reached a distance of around 0.25, a drop of almost 5 from the default Baum-Welch algorithm. So, with a relatively small push, we can already push the model in the right direction to reach an optimum very close to the original model. This is good news, because we believe that this push can be trained using a meta-learning algorithm.

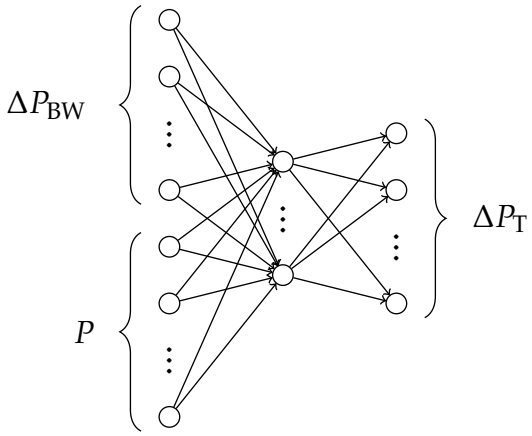


Figure 2.8: Schematic representation of an MLP designed to transform the change desired by the Baum-Welch algorithm to the change that would minimize the distance to the true model.

6.2 Is meta-learning necessary?

We want to use the fact that just a small push in the correct direction has a big impact on the final distance to the global optimum. However, in a realistic setting, the known model is not available, so the correct direction is unknown. This section describes an idea to apply the information that can be obtained when the global knowledge is available, in a setting where only local knowledge can be used.

The basic idea is to train, an external classification system to predict the push that is needed to move in the right direction given only the information available at the time. The data that is necessary for training such a neural network can be obtained by the following procedure: I. Generate data from a random model, II. Train multiple, randomly initialised models on the data, and III. Store the Baum-Welch update, ΔP_{BW} as well as the difference with the true model, ΔP_{T}.

While this approach needs more time to be fleshed out, a possible direction that is being investigated right now is to use a multilayer perceptron (MLP), such as sketched in Figure 2.8. It uses both the current model and the Baum-Welch updates as inputs and ΔP_{T} as the output. The model can be used to transform current

updates (when there is no global knowledge available) to a push in possibly the right direction. We consider this to be the next phase in the research towards finding the global optimum for HMMs. The novelty is that we can now use global knowledge to guide a local search algorithm.

7 DISCUSSION AND CONCLUSION

In this paper, we have examined the inner process of training HMMs. We have looked at the convergence of the Baum-Welch algorithm using tools such as generating models and sequences, comparing models using a χ^2-based metric, and changes in distance over time. We employed these methods to study common assumptions about HMMs and look at the conditions where HMMs converge to a local optimum, rather than a global one. Using a model, λ_O, we can generate sequences to train new, randomly initialised models. We then call λ_O the global optimum for the models trained on the generated sequences.

The common assumption that is tested in this study is that when the model is close to the global optimum (measured by distance), the performance (in terms of likelihood) will be good. In Section 3 we have shown that the assumption does not hold by showing that distance is a poor predictor of likelihood. In the same section, we have also shown that the distance of of a model to the global optimum does not necessarily go down during training with the Baum-Welch algorithm.

In Section 4 we looked at another assumption: if the initial model is close to the global optimum, the final trained model will also be close. Combined with the findings of Section 3, we have found that the distance of the initial model to the global optimum is a bad predictor for the final distance of the trained model to the global optimum. Therefore, there is no guarantee that a small initial distance will yield a good model.

Finally, in Section 5, we have shown that when we have a partially known, and fixed, model, the Baum-Welch algorithm does not have trouble finding the global optimum. This fixing, also known

as clamping, is common in applications in bio-informatics, most notably in gene-segmentation (Azad and Borodovsky, 2004).

These findings mostly contradict most common assumptions about HMMs and the training method that is most frequently employed to train HMMs and lead us to question whether maximizing the log-likelihood is the best optimisation criterion for training an HMM. It is important to note that we have only tackled a small portion of the problem. For instance, we have not looked at classification accuracy as the measure of performance. We have chosen to look at log-likelihood first because this is the measure that is optimised by Baum-Welch. It is commonly assumed that being close to the global optimum leads to a good log-likelihood, which in turn leads to a good classification accuracy ($D_{AB\pi}(\lambda_i, \lambda_O) \mapsto \log\ \mathcal{L} \mapsto$ accuracy (%)). In our experiments, we have only looked at the first step in this chain, but it is still very interesting to look at the second step in future work as well: does a good log-likelihood on the training set lead to a good classification accuracy.

Another problem that is not addressed in this paper is whether handwriting can actually be modelled as a Markovian process. There are studies that use context to model parts of the handwriting outside the current field of view of the model (Bianne-Bernard et al., 2011; Dolfing and Haeb-Umbach, 1997). A different method is applied by Frinken et al. (2014): by using a graph-cut approach to decoding HMMs, non-Markovian constraints can be used to model long-term dependencies in handwriting.

Ultimately, we are interested in the context in which an HMM will be used. We identify two movements currently. The first movement is towards systems that use convolutional neural nets and/or recurrent neural nets such as (B)LSTMs (Graves et al., 2009; Frinken et al., 2012). There seems to be a growing community of researchers and engineers that move in this direction, given by the recent successes achieved by this system.

On the other hand, in systems that still use HMMs, we see a movement towards fairly complex systems, with a lot of human (engineering) effort being spent in designing and optimising

systems that employ HMMs, either in terms of system complexity (Artières et al., 2007; Bideault et al., 2015; Khemiri et al., 2015; Roy et al., 2014; Ahmad et al., 2014), or in the number of hyper-parameters to tune (Britto et al., 2001; Rothacker and Fink, 2015; Puigcerver et al., 2015).

In both neural networks and HMMs research, we believe there is great need to understand the mechanism within the black box (see for other examples Elman, 1990 for multi-layer perceptrons and Schuster-Böckler et al., 2004 for HMMs). In this study, we have shown one particular method: looking at a local process using global information. We believe there is still much to be gained from this approach. For example, we could learn to recognise (un)successful trajectories quickly, or even use meta-learning to modify the inner-loop to 'push' the algorithm towards the global optimum. The experiments with the mix-factor are hopeful: The Baum-Welch algorithm only needs a small push in the right direction to find the global optimum.

To conclude, we have used artificially generated data to examine common assumptions about the convergence property of the Baum-Welch training algorithm. We found that the distance to the global optimum is a poor predictor of the final log-likelihood and that the distance does not necessarily go down during training. We also found that the distance of the initial, random model to the global optimum is a poor predictor for the distance of the final, trained model. Finally, we found that keeping either the transition or observation probabilities fixed makes the Baum-Welch algorithm converge very close to the global optimum.

POSTSCRIPT

In this chapter, we examined the machine learning aspect of the pipeline, as discussed in Chapter 1, by looking at the training process of HMMs. We challenged a number of assumptions on the convergence to the global optimum by creating a model, considering it the global optimum and generating data. This method can be used to study many different machine learning methods that have a tendency to end up in local optima.

Before moving on to other machine learning methods, we will use HMMs again to study another part of the handwriting recognition pipeline: Feature Extraction. This part of the pipeline builds a representation of the handwritten word images, in such a way that the machine learning can compute the class of a word image.

In the next chapter, we will move from the full-model comparisons from this chapter, to studying the relation between the learning algorithm and the feature representation. We will use the same method of generating data with known properties to study whether these properties can be learned and what happens if we remove temporal modelling from the model.

3

A REEVALUATION AND BENCHMARK OF HIDDEN MARKOV MODELS

Abstract

Hidden Markov models are frequently used in handwriting-recognition applications. While a large number of methodological variants have been developed to accommodate different use cases, the core concepts have not been changed much. In this paper, we develop a number of datasets to benchmark our own implementation as well as various other tool kits. We introduce a gradual scale of difficulty that allows comparison of datasets in terms of separability of classes. Two experiments are performed to review the basic HMM functions, especially aimed at evaluating the role of the transition probability matrix. We found that the transition matrix may be far less important than the observation probabilities. Furthermore, the traditional training methods are not always able to find the proper (true) topology of the transition matrix. These findings support the view that the quality of the features may require more attention than the aspect of temporal modelling addressed by HMMs.

1 INTRODUCTION

In 1989, Rabiner published the seminal work (Rabiner, 1989) on hidden Markov models (HMMs), with applications in speech recognition. Since then, HMMs have been used in other domains as well, such as segmenting gene sequences (Eddy, 1998) and handwriting recognition (Bunke et al., 1995; Plötz and Fink, 2009).

In this paper, we will discuss the applications in this last domain and HMMs in general.

There is a large number of variations of the regular HMMs that Rabiner wrote about, ranging from pseudo 2D-HMMs (Kuo and Agazzi, 1993), to truly 2D-HMMs (Markov random fields) (Park and Lee, 1998) and explicit duration modelling (Benouareth et al., 2008), to nested HMMs (Borkar et al., 2001) and many more. In the core, these variations are still HMMs, usually trained using the Baum-Welch algorithm. When the data is already labelled with hidden states, however, the transition probability matrix can be modelled directly, without using the potentially more unpredictable EM-based approach. This is the case in segmenting gene sequences with profile HMMs (Eddy, 1998) for example, using many pattern heuristics to identify state-transitions in the sequence.

The overall HMM architecture (e.g., determining the number of states, transition matrix topology and integrating it into a larger framework) requires a lot of human effort. However, to our knowledge, no real benchmark has been proposed to test algorithm variants of HMM implementations. In section 2, we will discuss how such a benchmark can be constructed. It will not only provide a way to compare results, but also allow one to determine the difficulty of a particular dataset.

The goal of this paper is to investigate the core of HMMs. HMMs consist of three main components: the initial state probability distribution ($\vec{\pi}$), the transition probability matrix ($\mathbf{A}$) and the observation probability functions ($\mathbf{B}$). While the role of the initial state probability distribution is known to be of relatively small importance (especially in left-right topologies such as Bakis, since these models always start in the first state), it is hard to find concrete information on the relative importance of the transition and observation probabilities for optimal performance in the literature.

Artières et al. (2002) mention in passing the importance of the observation probabilities over the transition matrix. However, the study does not provide further information. Therefore, in

section 4 an experiment is presented to gain a better insight in the importance of the transition matrix. It will show that, indeed, the observation probabilities are very important. The implications of this observation and the consequences for using HMMs as a paradigm in handwriting recognition are discussed in the final section.

We will show, using generated data, that it is very difficult for the Baum-Welch algorithm to find the correct topology of the underlying Markovian process. By generating data according to a known Markov process with very specific properties (namely a left-right HMM), we know which properties the ergodic model, initially without any restrictions, should get after training. We can now show that the explicitly coded left-right topology is not found by an ergodic model. See also Figueiredo and Jain (2002) for a discussion of the brittleness of EM algorithms.

Finally, we show that, surprisingly, removing the temporal information from an HMM does not necessarily have a large impact on performance in a real-world problem.

2 BENCHMARK

We will run some experiments using our own implementation as well as other HMM toolkits on a generated data set as a benchmark. It is hard to find a proper HMM benchmark for discrete, one dimensional data that has a gradual scale of increasing difficulty. The dataset that was generated for this purpose has varying degrees of symbol lexicon overlap between classes, making the completely overlapping set most difficult and the dataset with the largest between-class distance least difficult. This is useful for comparing performances between runs on different feature methods, having the ability to attach a 'difficulty index' to each.

The generated data contains 100 classes, each class consists of generated transition and observation matrices. The transition

matrix is a randomly[1] initialised Bakis model with $N_{\text{states}} = 10$ states, which is appropriate for variable duration modeling of left-right symbol sequences. The observation probability functions, with $N_{\text{symbols}} = 20$ symbols, are also instantiated randomly. The topology was chosen as Bakis in this benchmark. Most HMM implementations do not have restrictions on topology, except the dHMM framework (which uses a fixed, hard-coded Bakis structure). See also section 3 for more details on different topologies.

The gradual scale of difficulty is achieved by having multiple data sets with a varying degree of separability in symbol space. Concretely, this means that there is an overlap in lexicons between classes. A separability of δ of a dataset is defined by the following equation:

$$\begin{aligned} L_1 &= \{1 \dots N_s\} \\ L_i &= \{L_{i-1,0} + \delta \dots L_{i-1,0} + \delta + N_s\} \end{aligned} \tag{3.1}$$

Where δ is the separability, L_i is the lexicon, the set of symbols, to be used for class i, $L_{i,j}$ is the j^{th} element of L_i and N_s is the size of the lexicon, i.e., number of symbols per class. A separability of $\delta = 0$ is the most difficult case, because all classes share the same set of symbols: $L_1 = L_2 = \{a, b, c\}$. A separability of $\delta = 1$ means that between classes, one symbol is not re-used in the next class: $L_1 = \{a, b, c\}$ and $L_2 = \{b, c, d\}$, and so on. With more separation than symbols, a gap between the symbols is present: A dataset with $L_1 = \{a, b, c\}$ and $L_2 = \{e, f, g\}$ has a separation of $\delta = 4$.

In this section, we will show the results of running several HMM frameworks on the generated datasets. We test the popular HTK tool kit, which is well known in speech recognition (Young et al., 2006); GHMM, developed mainly for bio-informatics applications (ghm, 2003); a framework developed by Myers and Whitson (Myers and Whitson, 1994), dubbed dHMM here, mainly for discrete Bakis models for automatic speech recognition; and finally our own framework, developed from scratch to review in great detail the algorithmic details of HMMs, dubbed jpHMM.

1 Using the default python module random, which uses a Mersenne twister pseudorandom number generator

Table 3.1: Average classification performance (%) of three randomly initialised runs on the same dataset. Please note that the standard deviation for dHMM and GHMM is 0 due to the use of a static random seed, instead of a random seed. HTK-hinit uses the hinit tool to initialise the model with some estimates from the data, which increases performance only slightly on these datasets. The very small difference between a separability of $\delta = 10$ and $\delta = 20$ is not visible in this table. $N_{\text{states}} = 10$, $N_{\text{symbols}} = 20$

Separability (δ)	jpHMM	dHMM	GHMM	HTK	HTK-hinit
0	1% (± 0.10)	1%	1%	1% (± 0.12)	1% (± 0.06)
1	41% (± 0.46)	40%	37%	41% (± 0.12)	41% (± 0.62)
2	66% (± 0.38)	64%	61%	66% (± 0.10)	66% (± 0.15)
3	81% (± 0.10)	78%	76%	80% (± 0.10)	80% (± 0.10)
5	95% (± 0.25)	93%	92%	94% (± 0.17)	94% (± 0.15)
10	100% (± 0.00)	100%	100%	100% (± 0.00)	100% (± 0.00)
20	100% (± 0.00)	100%	100%	100% (± 0.00)	100% (± 0.00)

We also use the HTK toolkit together with the hinit tool to have a better initialised model, compared to random initialisation.

The benchmark datasets in this paper are all synthesized and discrete. Also, the duration of the sequences is limited. This means that the results of the current study can not directly be compared to all possible applications. However, there is no fundamental limitation on sequence length, or number of states. This can be addressed in future releases of the benchmark. For some applications and features, continuous observation modelling is beneficial (Chen et al., 1995), while for other applications and feature methods, discrete observation modelling is still very relevant (Rigoll et al., 1996). In order to study the core details of HMMs, using discrete observations is interesting, since its modelling is almost trivial. Common techniques to use discrete models on continuous data are vector quantization (Schenk et al., 2008), k-means clustering, or self-organizing maps (see also section 4).

The generated datasets have a separability of $\delta \in \{0, 1, 2, 3, 5, 10, 20\}$. The number of states is $N_{\text{states}} = 10$, $N_{\text{symbols}} = 20$, the length of each sequence is $|\vec{O}| = 10$ observations, yielding effectively an artificial stochastic language with 10-letter words. We have generated 100 classes with 300 se-

quences each. We trained models from each toolkit on all classes, and performed classification based on the most likely model for an instance.

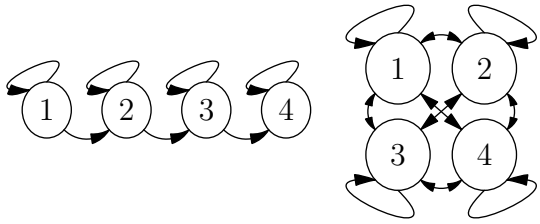
Results The classification performances on the seven datasets are reported in Table 3.1, showing that all implementations perform roughly equally well, which is to be expected. However, we can also see the relation between benchmark difficulty, the separability δ and classification performance for five HMM implementations. From a separability of about $\delta = 5$ onward (for a dataset with $N_{\text{states}} = 10$ and $N_{\text{symbols}} = 20$) classification becomes very accurate.

3 LEARNING THE TOPOLOGY OF A TRANSITION MATRIX

In this section and the next, we describe two experiments to determine the importance of temporal modelling which is effectuated by the transition matrix in the HMM framework. The first experiment is mainly focused on the performance of the Baum-Welch algorithm, while the second shows what happens when the temporal information is removed from an HMM.

The Baum-Welch algorithm, an Expectation-Maximisation (EM) algorithm, works by initialising a model, often by using random probabilities, and then incrementally improving it. The initialisation step is very important due to the possibility of ending up in a local maximum, and the 'random' method is therefore very brittle, requiring human supervision.

As a first experiment to examine the transition matrix, we generate artificial data again. This has the advantage that we explicitly know the properties of the transition matrix. The specific property that we are interested in, currently, is the topology of the model. The topology is the shape of the transition matrix and there are a number of topologies possible. The most well-known is the Bakis topology, which is a left-right model that defines for each state two transition probabilities: to the current state and to the next state. Another topology is the Ergodic topology, which puts no a-priory restrictions on the transition probabilities: every



(a) Bakis (b) Ergodic

Figure 3.1: Illustration of the Bakis and Ergodic topologies. The arrows indicate the possible transitions between the numbered states, without indicating the probability of these transitions.

state has a (possible) transition to every state (including itself). See Fig. 3.1 for an graphical representation of these topologies. A variant of Bakis, that has the ability to skip a state by also having a transition probability from S_i to S_{i+2}, was left out for brevity.

The experiment is set up as follows: a model is created by randomly initialising a Bakis topology with $N = 20$ states ($L = 20$ symbols). After generating 300 instances of 40 observations long with this topology, a fully Ergodic model is trained on these instances. The resulting transition matrix is examined: has it learned the fact that we used a Bakis topology to generate the training data? To be fair, we shuffle the states in the trained model to have the smallest χ^2 distance to the original model. The found hidden state S_1 in the trained model does not have to be state S_1 in the generating model, after all.

Results The original, generated model and the learned ergodic model can be visually inspected in Fig. 3.2. The transition matrix is converted to an image by taking the state-transition probability and coding it into a grey-scale colour: a probability of 0 is rendered as white, while a probability of 1 is rendered as black. From these figures, we can see that the Bakis topology has a diagonal structure: a probability from state S_i to S_i and to state S_{i+1}. The learned, ergodic model does not show a diagonal structure at all, even though we shuffled the matrix to have the smallest χ^2 distance to the generated Bakis model. The learned model is significantly different from the generated Bakis

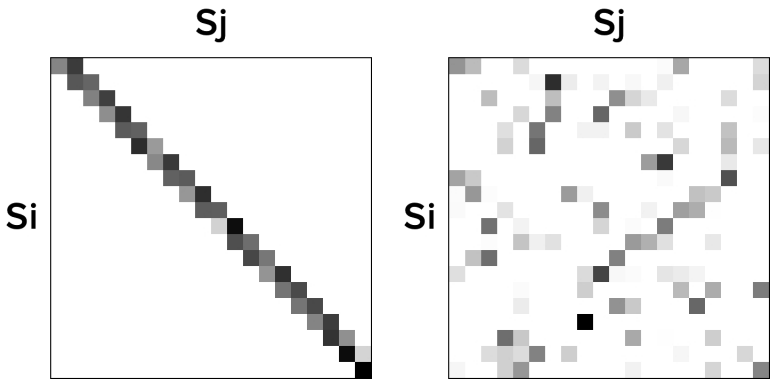
model ($p \ll 0.0001$, $\chi^2 = 27863$, 19 degrees of freedom), using a contingency table test on the transition frequencies[2].

From this observation we could conclude that it is difficult to learn the topology of an underlying Markov process. We performed two similar experiments to verify this finding (this time with $N = 10$ states because of compute time constraints). The first variation was done by averaging over several learned models. This is realised by generating ten Bakis models, generating 300 sequences per model and train an ergodic model on each set of sequences. The models are shuffled and averaged, and visualised in Fig. 3.3. Although we are not aware of this extensive procedure being done in the literature, it appears to be useful to see whether a diagonal pattern can be found, on average, even when it is difficult to see in a single model.

It is well known that one requires a large set of training sequences to estimate the right model. We used this idea in another method of trying to find the underlying Bakis structure using an ergodic model. Instead of averaging over ten models, we now use ten times as much data. This gives the training algorithm more data to learn the structure from. Fig. 3.4(b) shows the results of estimating the topology from 3000 sequences, that were generated using the model shown in Fig. 3.4(a).

Both Fig. 3.3 and 3.4 show that trying really hard to force an ergodic model to find the Bakis structure can result in a slight tendency towards a diagonal structure under highly artificial training conditions. The desired diagonal probabilities are present in the learned ergodic models, but the off-diagonal probabilities are abundant in these models as well. From the diagonals, the self-recurrent state-transition probabilities are most pronounced. This shows that it may be very difficult to find the underlying structure of a Markov process using the Baum-Welch algorithm (given the specific parameters). From this and pilot studies, we conclude that it is less difficult to find a diagonal structure for $N = 10$ states than for $N = 20$ (which is more common). We will verify this in a future study.

2 The Kolmogorov-Smirnov test cannot be used since there is no meaningful univariate axis to integrate the probabilities (Babu and Feigelson, 2006).



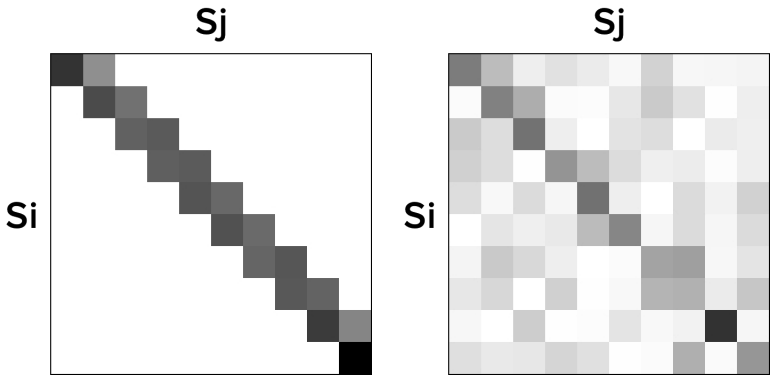
(a) Target (Bakis) model (b) Learned (ergodic) model

Figure 3.2: Transition probability matrices. After generating a model of $N = 20$ states, Fig. 3.2(a), 300 sequences were generated with this model. A new model was trained on this data, and after shuffling the learned model such that it is closest to the original model, we can see that it has not learned the topology, Fig. 3.2(b). A probability of 0 is rendered as white, a probability of 1 as black. χ^2 distance = 48

4 THE IMPORTANCE OF TEMPORAL MODELLING

We are also interested in what happens when we remove the temporal information from the transition matrix. This means that we create a *flat* topology: all transition probabilities are equally probable: $a_{ij} = \frac{1}{N}$, where N is the number of states. During training, the transition matrix will continuously be made uniform (i.e., flat) in each iteration. This is necessary because the observation probabilities may no longer be correct when adjusting the transition probabilities after training. The flat topology can be viewed as an orderless "bag of states". We will now compare how well models with this topology compare to models with a Bakis or ergodic topology.

In this experiment we train an HMM on discrete features, extracted from handwritten word images. The dataset uses a single handwritten book from the collection of the Dutch National Archives (van der Zant et al., 2008a). We use two features: fragmented connected component contours (FCO^3) and a sliding window, both quantized using a Kohonen self-organizing feature map (SOFM, see Kohonen, 1987).

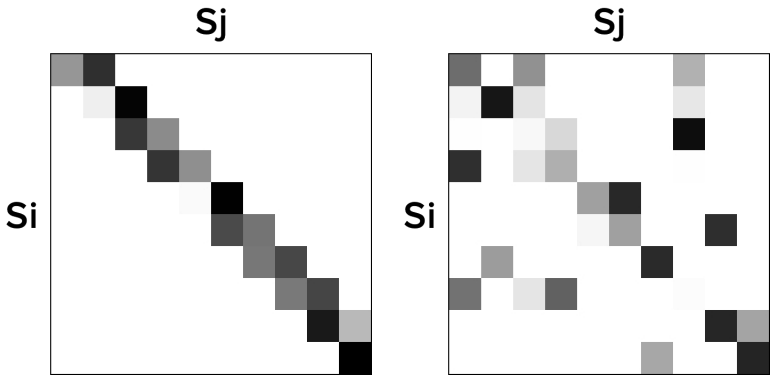


(a) Target (Bakis) model (b) Learned (ergodic) model

Figure 3.3: After generating ten models with the number of states reduced to $N = 10$, per model 300 sequences were generated, otherwise similar to Fig. 3.2. New models were trained on each of these 300 sequences and the models were averaged. Fig. 3.3(a) shows the average model of the generated models, while 3.3(b) shows the average learned model, with a vague tendency towards diagonal state-transitions, mostly the self-recurrent transitions, while the next-state-transitions show a less pronounced pattern. Average χ^2 distance = 16

For the FCO^3 feature, the image is broken up into a sequence of fragmented connected component contours (Schomaker et al., 2007). Each of these contours is then quantized into a discrete index of a SOFM of 70×70 nodes. This means the lexicon consists of 4900 symbols. We have selected 130 classes with at least 51 training instances, with a total of 30 869 labelled instances. Because the average length of the words was 4.4 FCO^3 observations, the number of states was chosen to be 3.

The second feature is extracted using a sliding window of 4 by 51 pixels, centered around the centroid of black pixels over the entire word zone. The SOFM for this feature, with 25×25 nodes, was a lot smaller than the FCO^3 feature map, due to time constraints. Centering around the centroid with a height of 51 pixels means that the outstanding sticks and (partial) loops of ascenders and descenders are still preserved, while reducing the size of the image considerably. We limited the number of classes in the experiments with this feature to 20, with a total of 4 928 labelled instances. The average length of observation sequences



(a) Target (Bakis) model (b) Learned (ergodic) model

Figure 3.4: Instead of averaging over ten models as in Fig. 3.3, we now use ten times as much generated sequences to train a single model (3000 sequences). We see that there is a small tendency towards diagonal (Bakis-like) state transitions, but it is not very strong. χ^2 distance between the two distributions = 14

for the sliding window feature was 65.9 observations, which led us to use $N = 27$ states[3].

For classification, an HMM λ is first trained on the instances of each class, and then classification can be performed using $\text{argmax}_{\lambda \in \Lambda}[\log P(O|\lambda)]$, where Λ is the set of all trained models and O is the test sequence. To investigate the role of the state-transition probabilities, we perform the experiments with three topologies: Bakis, ergodic and flat, which is the topology where all transition probabilities are equally probable. We perform the experiments on both features using 7-fold cross validation, with our own implementation, jpHMM.

Results The results are summarised in Table 3.2 and 3.3. We can see that the results of classification with the FCO^3 feature are very close together (and not statistically significant, with ANOVA, $p > 0.05$). There is a significant difference in the classification performance using the sliding window feature (ANOVA, $p < 0.001$), but the drop in performance is not as dramatic as would

3 The increase in number of states is most likely the reason for the increased time necessary for training

Table 3.2: Results of the FCO^3 experiment. Performances reported are averages over 7 folds, with 130 classes and at least 51 instances per class in the training set. As can be seen, all topologies perform around 60%. Flat models do not perform significantly worse.

Topology	Classification performance
Bakis	59.9% ± 0.9
Ergodic	59.5% ± 0.9
Flat transition probabilities	59.1% ± 0.8

Table 3.3: Results of the sliding window experiment. Performances reported are averages over 7 folds, with 20 classes and at least 51 instances per class in the training set. Differences between the topologies are statistically significant ($p < 0.001$) although the difference between the flat and ergodic topologies is not as dramatic as expected (Please note that in the Flat condition of the transition matrix, temporal information is completely uniform).

Topology	Classification performance
Bakis	75.2% ± 2.0
Ergodic	78.5% ± 1.2
Flat transition probabilities	71.1% ± 1.3

be expected from the removal of temporal information in the Markov paradigm.

Please note that the HMMs were used as a measurement tool to find differences between transition models. They have an average performance, avoiding ceiling effects. Also, the FCO^3 feature is a feature developed for writer identification, not handwriting recognition per-se. The sliding window feature could be fine-tuned further by changing the size of the Kohonen map, the window, the number of states, etc. In this experiment we are interested in evaluating HMM topologies, not in maximising the recognition performance.

5 DISCUSSION

We set out to reevaluate hidden Markov models, by creating a benchmark for discrete HMMs, and running experiments to investigate the importance of the transition matrix.

Using the benchmark, we found that for discrete observations, all common HMM tools have similar performances. Furthermore, we can now measure the difficulty of discrete data, by comparing the performances of discrete HMMs with the performances of the benchmarks with different degrees of difficulty. In the future we want to extend the current study with continuous density HMMs as well.

While it is barely presented in the literature, the fact that the transition matrix is of a smaller importance than the observation probabilities is well known from personal communications at, e.g., conferences. We have done two experiments to establish the importance of the transition matrix, and found that indeed the observation probabilities have a large impact on recognition performance. The results of these experiments showed that (a) it is hard to learn the correct, known topology of the underlying Markov process and (b) that classification with the temporal information removed from the HMMs can also result in reasonably performant classifiers.

Regarding (a), it appears that the Baum-Welch training method is not very reliable to estimate the underlying transition structure in the data. As noted in (Figueiredo and Jain, 2002), EM is brittle and very sensitive to the initialisation process. We have shown that the Baum-Welch method was unable to find the Bakis topology from generated data when initialised as a full Ergodic model. We have previously studied initialisation of models to prevent local maxima (Bhowmik et al., 2011), but this still requires a lot of human modelling effort, specifically for each problem variant.

Regarding our finding (b), that classification with temporal information removed can result in performant classifiers, we believe that the observation probabilities are very important. This sup-

ports our view that the quality of the features may require more attention than the aspect of temporal modelling. From a more scientific point of view, it is still a challenge to adapt the Baum-Welch estimation algorithm to correctly estimate the Markov parameters of an observed sequential process.

Even though these findings expose limitations of HMM and its training procedure, the fact that recognition performance is not degraded dramatically when removing temporal information from HMMs implies that dynamic programming (i.e., the operational stage) is a strong principle. Also, the Markov assumption remains appealing from a theoretical perspective.

Given these considerations, we feel 1) that it may be misleading to stress the *hidden* aspect of HMMs, because of the relatively minor role the hidden states play in achieving good performance, 2) the Baum-Welch algorithm should be replaced with a less brittle method, and 3) although the HMM principles mentioned above are strong, there are many tricks of the trade, that are not treated well in literature (see also the Appendix).

POSTSCRIPT

In the preceding two chapters of this dissertation, we have looked at assumptions in HMMs, from a machine learning and a feature-representation perspective. It appears that properly learning the parameters of a Markovian process is not straightforward. It has been hard to replicate state of the art HMM results with the datasets that are available in a large operational system such as Monk. One of the causes of lower performances on these datasets is bootstrapping: Adding new collections to the system frequently means starting from scratch, since no labels are available and the script is very different from other collections. Therefore, the Monk system uses various feature-extraction techniques and machine-learning methods, each with their own strengths.

While representation and learning methods are core components of a search engine for historical documents, they only work properly when there are enough labelled instances in order to train

them. Furthermore, labels are essential in order to assess the quality of the techniques. Researchers frequently use pre-existing labelled datasets to improve existing techniques or develop new methods. Consider for example the many competitions and benchmark datasets that are referred to in handwriting recognition and machine learning literature (Clausner et al., 2018; Strauß et al., 2018). These are useful to be able to compare different approaches, but it is easy to lose sight of the actual use-cases—in the case of Monk, the use-case is a search engine for historical document collections that are sometimes obscure and scarcely labelled.

Bootstrapping and the need of (a lot of) data are fundamental issues that should be addressed in any form of current (deep) machine learning systems. Because certain methods work better in the bootstrapping phase than when there are plenty of labelled instances across all classes, we believe that it is a good idea to apply the methods in an iterated fashion. Furthermore, we believe it is a good idea to let the machine help in the labelling process such that the most relevant instances are considered first. This is more efficient than labelling images from left to right, top to bottom. We therefore consider the handwriting recognition process to be a loop instead of a pipeline: In Monk, labelling is iterative and in each iteration new images are presented.

The iterative application of labelling can be compared to the "harvesting" of labels: By ordering the unlabelled images in such a way that the process can be performed efficiently, a snowball effect can be created. Such an effect can be marked by large jumps in the amount of labels added to the system. The jumps are created by allowing the annotators to quickly label a lot of images in one go. When a suggestion for the labels is provided and the user interface allows you to acknowledge these labels all in one go, the model improves, which in turn improves the quality of new labels, which finally allows for more labels to be added with a few clicks.

In Monk, the user interface for quickly labelling images is a hit list. The images are arranged in a list, usually presented in a table, that match a certain class. This list is ordered in such a way that

the images at the top are good enough to be acknowledged to be of this class. The interface then has two options: accept the first N labels, or manually select all images that should have their labels be accepted. In the bootstrapping phase, it is frequently observed that the latter method is used by the human labellers, due to the classifier not being able to accurately find enough samples, while in later stages, the "snowball" (a critical mass of $\{image, label\}$ tuples) has gained enough traction to provide enough samples to fill the first page with correctly labelled images.

The Monk web-based labelling system has been invaluable to the collection of labels in a number of rare manuscripts. This allowed the researchers to improve their methods in tandem with the improvements of the labelling of the collections. New performance indicators are needed that allow researchers to see how their methods affect the quality of the hit lists. The default method of a mean average precision (MAP) is not sufficient to get an indication of the quality of highest ranked samples. The next chapter will study hit lists and the functions that need to be optimized in further detail.

4

SEPARABILITY VERSUS PROTOTYPICALITY IN HANDWRITTEN WORD-IMAGE RETRIEVAL

Abstract

Hit lists are at the core of retrieval systems. The top ranks are important, especially if user feedback is used to train the system. Analysis of hit lists revealed counter-intuitive instances in the top ranks for good classifiers. In this study, we propose that two functions need to be optimised: (a) In order to reduce a massive set of instances to a likely subset among ten thousand or more classes, separability is required. However, the results need to be intuitive after ranking, reflecting (b) the prototypicality of instances. By optimising these requirements sequentially, the number of distracting images is strongly reduced, followed by nearest-centroid based instance ranking that retains an intuitive (low-edit distance) ranking. We show that in handwritten word-image retrieval, precision improvements of up to 35 percentage points can be achieved, yielding up to 100% top hit precision and 99% top-7 precision in data sets with 84 000 instances, while maintaining high recall performances. The method is conveniently implemented in a massive scale, continuously trainable retrieval engine, Monk.

1 INTRODUCTION

In handwriting recognition, classification is often performed using statistical methods (Duda et al., 2001; Bunke, H., 2003). The class indexed i with the highest posterior probability given

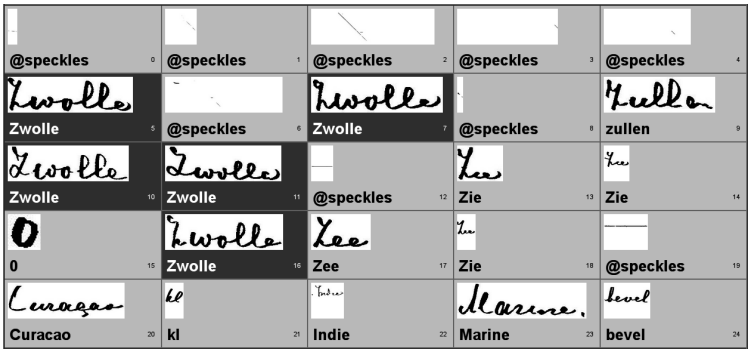


Figure 4.1: First 25 instances in a hit list of the word 'Zwolle'. Original test set performance: Accuracy: 99.2%, precision: 97.6% and recall: 97.6%. Note the faulty instances in the top ranks, upper row. In a realistic test condition with 12k distractors, actual precision is as low as 2.8%.

the sample to be classified is chosen as the result of the classifier:

$$\text{Hypothesis}_X = \underset{i}{\text{argmax}}\, P(C_i|X) \quad \text{where } i \in \{1, N_{\text{classes}}\} \tag{4.1}$$

However, when the goal is word search, rather than automatic text transcription, the user is more interested in retrieval of word instances. Instead of a single classification, the result is a sorted hit list H. Each instance indexed j is ranked with respect to the prototype or class-model corresponding to the search term:

$$H = \underset{j}{\text{sort}}(P(X_j|C)) \quad \text{where } j \in \{1, N_{\text{examples}}\} \tag{4.2}$$

Retrieval is usually performed on a large collection of instances, and only the top of the sorted list, representing the best ranking instances, is considered as interesting. Under such a condition, a large number of classes and a massive data collection can pose a problem, since for each query there is a large number of distractors, i.e., concerning instances from all classes, other than the target class.

This becomes apparent in retrieval engines for handwritten words in historical collections (van der Zant et al., 2008a). In the *Monk*

system, twenty books of ≈1000 pages each contain millions of word zones or word candidates, and the lexicon is in the order of tens of thousand word class models. From the tradition of handwriting-recognition research, it seems reasonable to start with the classification problem (Eq. 4.1), using good shape features and a powerful classifier, such as, e.g., hidden-Markov models (Marti and Bunke, 2000; Artières et al., 2007) or the support-vector machine (Vapnik, 1982; Boser et al., 1992). For a word-mining task, such a classifier may be trained to discriminate a particular word class, and a ranked word list may be constructed, e.g., using the signed SVM discriminant value d_{SVM} for sorting. The basic assumption then is, that the distance from the margin, i.e., from the instances in the distractor classes, will be a good criterion for constructing a ranked hit list for a target class. However, upon applying this approach, we observed an interesting phenomenon in the resulting hit lists. As an example, Figure 4.1 shows the top-25 instances in a hit list for the word 'Zwolle'. The performance for the word classifier on the entire training set was 100% accuracy, with a 97% accuracy on an independent test set (k = 7 folds, σ = ±1%). Following regular testing procedures for SVMs, the training and the test sets were of similar size, each containing a quarter of positive examples (typically 50) and three quarters of negative or distractor examples. However, the resulting hit list contains a number of counter-intuitive samples (e.g., speckle images) in the early ranks, followed by a strand of correct classifications which is followed by a transitional stage of occasional errors.

The impression that a problem exists is confirmed by a larger-scale analysis of the results (Table 4.1), also using a realistic large set containing ≈ $12{\times}10^3$ distracting word instances in the test set. The results for *accuracy* and *recall* on the realistic data set confirm the hopeful expectancies which were raised by the regular training and test sets. However, the *precision* of the output drops abysmally, to about 1% in the worst cases, notably for the classes with a limited number of training examples (Table 4.1, lower right). It should also be noted that a number of 12K distractors (1/1200) is much more realistic than a 1/4 rule which is commonly accepted in academic testing.

Table 4.1: Counter-intuitive, low precision results for good classifiers

		Accuracy		Recall		Precision	
Set	$N_{examples}$	Mean	σ	Mean	σ	Mean	σ
Test	120+	0.98	0.02	0.97	0.05	0.96	0.07
	60-120	0.97	0.03	0.95	0.10	0.91	0.13
	35-60	0.97	0.04	0.93	0.15	0.85	0.19
	7-35	0.96	0.04	0.68	0.42	0.57	0.40
+12K Distractors	120+	0.99	0.01	0.97	0.05	0.26	0.26
	60-120	0.98	0.02	0.95	0.10	0.06	0.12
	35-60	0.97	0.02	0.93	0.15	0.03	0.06
	7-35	0.97	0.04	0.68	0.42	**0.01**	0.05

It is clear that something is needed to improve on the performance. User appreciation of hit lists is of paramount importance in live and continuously trainable systems that rely on user annotation over the internet, such as *Monk* (van der Zant et al., 2008a, 2009). Figure 4.2 shows how hit lists are used in the Monk system. Upon giving the first handful of (bootstrap) examples, a usable machine-learning system should be able to produce an acceptable ranking such that

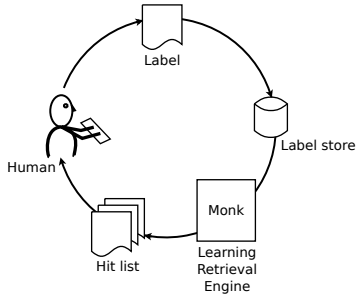


Figure 4.2: Schematic overview of how users utilise the hit lists to label new word images in a continuously learning retrieval engine (Monk). A hit list is presented to the user, who produces a label for an unlabelled word. This label is stored in the label store, which is then processed by the retrieval engine to produce a new hit list. The interface facilitates the quick labelling of a large number of instances that match the query word. See also Figure 1.6 on Page 10

newly found instances of the same class can be easily labelled. The above, concrete observation thus gives rise to a more fundamental question: How is it possible that accuracy is not a good predictor of precision in a retrieval context?

In this study, we will 1) analyse the reason for unexpected, low precision in presumably well-performing classifiers; 2) explore a number of methods to counteract the precision drop and 3) present a convenient approach using nearest-centroid matching, with results in a similar ballpark as the abovementioned SVM approach, at the same time however, avoiding expensive training on the tens of thousands of word classes.

2 SEPARABILITY VERSUS PROTOTYPICALITY

Problem: The SVM is a discriminative classifier, optimised for *classification* (Eq. 4.1). The class of an unknown sample X (Figure 4.3) is decided by determining on which side of the decision boundary β the sample falls. For *retrieval* purposes, it appears reasonable to use the distance to the boundary, $d(X, \beta)$, as a ranking measure: the farther the instance is located from the boundary, the more certain an SVM classifier is of the classification.

Unfortunately, this gives unexpected results, such as shown in Figure 4.1 for the query word 'Zwolle'. Instances that are ranked at the top (@speckles) appear to be counter intuitive to a human user. It seems that there are two problems: 1) the distance to the boundary is not an intuitive measure, and 2) a fairly large number of distractors causes noise in a hit list, and consequently, a lower precision. The implication is that enlarging the dataset increases the probability that incorrect instances occur even before the first correct hit. This has a large impact on the user appreciation and is hard to explain. More informally: Many hits do not appear similar to the user's expected, canonical prototype for the query.

Proposed explanation: In order to give a plausible explanation of this phenomenon, we present a schematic, two-dimensional overview. The position of an instance X in Figure 4.3 has a large distance $d(X, \beta)$ from the boundary β (which is desirable).

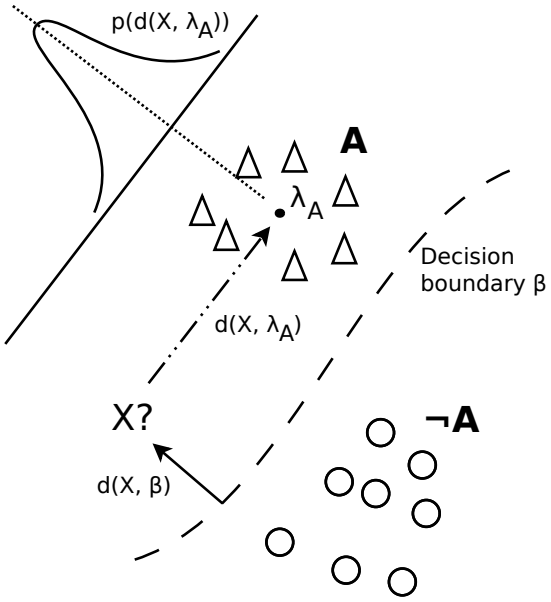


Figure 4.3: Separability vs. Prototypicality: For an unknown instance X, a large distance $d(X, \beta)$ from a margin β does not imply a short distance, $d(X, \lambda_A)$ from the prototype λ_A

However, the instance X is not very prototypical, being located far from the known instances of the target class A. In other words, the distance of the instance X to the prototype, or centroid of class A, $d(X, \lambda_A)$, is large.

The support-vector machine training mechanism has an emphasis on *separability*: the ability to categorise and separate class instances from non-class instances. This ability is usually achieved by evaluating the computed signed distance of an unknown sample to the decision boundary $d(X, \beta)$ which indicates on which side the instance X falls. However, by focusing on separation, an important aspect of pattern recognition is neglected: The phenomenon of *prototypicality* which concerns the similarity of an instance to the canonical class prototype, for instance, measured as the distance to the centroid or prototype of the class $d(X, \lambda_A)$. Quantitatively, prototypicality can be defined as $p(d(X, \lambda_A))$ and

is also the underlying rationale for Bayesian classifiers, exploiting the high density of feature values around the mode of their distribution, as opposed to the SVM. It is important to realise that the prototypicality of instances directly affects the ease with which new training examples can be elicited from users in a continuously learning retrieval system. The degree of prototypicality of the hit list directly affects the gain factor in the feedback loop of the label harvesting system that is presented in Figure 4.2.

For a search and annotation tool of handwritten historical documents, separability and prototypicality need to be optimised simultaneously. It can be argued that similar requirements play a role in general content-based image retrieval, too (Datta et al., 2008; Schomaker et al., 1999). However, most classifier methods optimise for one property, not both. The solution proposed in this study, is to combine classifiers in a two-stage process. The classifier that optimises separability is used in the first stage to divide the instances and produce the most likely class *C* for an unlabelled instance. The goal is to reduce the number of distractors for the second stage. More specifically, the set of distractors of an instance classified as *C* will be a considerable reduction of the set of all instances.

All instances labelled as *C* are then gathered for the second stage, where all instances are re-ranked or re-sorted with a secondary feature or method, one that optimises the ability to rank instances according to prototypicality. This ensures that if an instance is classified as class *C* in the first stage, but is an atypical result (such as the first few results in Figure 4.1, i.e., the speckles), the instance will end up at a later position in the hit list than other, more prototypical examples. Similar problems will occur if reject criteria need to be defined while using the SVM (Mouchère, 2007), or when there are very few negative examples to train from — for example, in a machine diagnostics problem (Tax, 2001). For a schematic overview of the entire re-ranking process, see Figure 4.4.

The results from the SVM experiment in the introduction suggest that a larger number of distractors has a negative effect on retrieval precision. It should be noted that the experiments

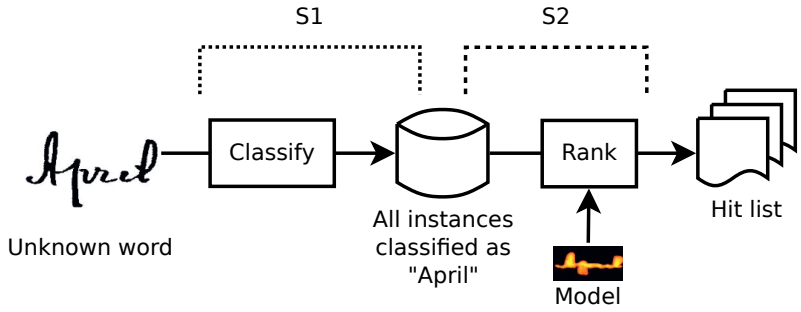


Figure 4.4: Schematic overview of the re-ranking process. The first stage (S1) shows that a word is classified first, and gathered together with other instances that have been classified the same. These instances are then ranked (S2), according to their prototypicality, to produce a ranked hit list.

in this study are conducted in a laboratory setting, using only human labelled instances. In a real-world setting, the problem of distractors will even be worse: the problem space is then heavily populated with non-word images and other noise. For example, in Monk, over all collections there are $22{\times}10^3$ classes, with over $124{\times}10^6$ word images, including rejectable candidates and noise. These numbers indicate the massive size of the current experimental test bed. Instead of pre-cleaning the data, we assume a rigorous, machine-learning approach where as much of the problems are solved by the base classifier and not by the use of overly specific hand-coded preprocessing heuristics. That means that problematic patterns have to be labelled as well. In Monk, there are several classes that are indicated by a label starting with @, and can indicate whether this is, e.g., a table-line, speckles or other noise.

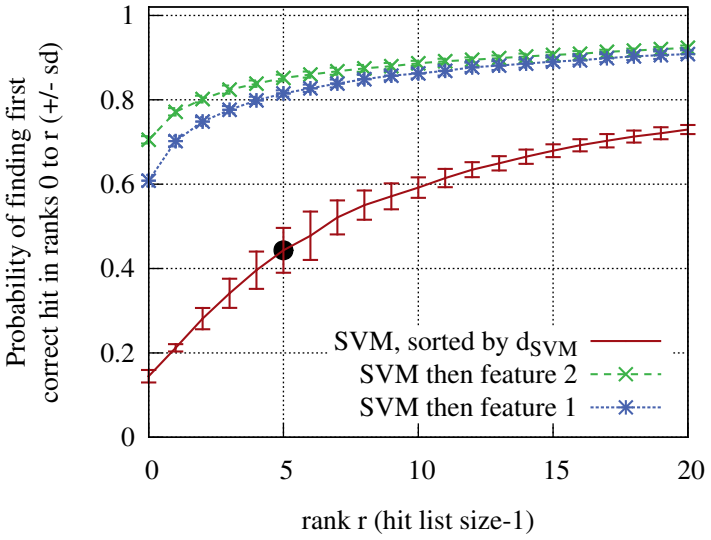


Figure 4.5: Probability of finding the first correct hit in ranks 0 to r for raw and ranked SVM output ($N_{\text{folds}} = 7$). The bars give the standard deviation, which are only clearly visible on the SVM, sorted by d_{SVM} results. Note the strong improvement due to secondary ranking for all ranks but especially for the top hit accuracy at $r = 0$. Feature 2 outperforms Feature 1 significantly. The circle is used as a reference point in the text.

3 METHODS

Figure 4.5 shows the probability of finding the first correct hit in the ranks 0 to r of the hit lists generated in the preliminary study from the introduction. It is apparent that the probability of finding the first correct hit in the first five ranks is roughly 45% (indicated by the circle in Figure 4.5), when using the SVM discriminant value for initial (tier 1) ranking. By reordering the images using a different feature, the performance can be improved, such that the first correct hit is found in the first five ranks 80% of the time (Figure 4.5, upper left). This is hopeful, but this is not enough and the hit list still contains counter intuitive results in the top ranks. There are other ways of improving the tier-1 performance. For example, multiclass SVMs, using decision trees (Takahashi and Abe, 2002), could improve the classification accuracy before ranking, which seems to be beneficial, but it has the downside of requiring a large number of training instances

for each of the more than 10^4 classes. Approaches like Gaussian mixture models (GMMs) or hidden Markov models (HMMs) can also improve the classification accuracy, but also require a large number of training examples. Benefits such as multi-peak distributions can be achieved with more simple techniques, such as (*k*-means) clustering. The *Monk* system is a continuous, '24/7' training system: Labels are continuously added or changed, and it would be too time consuming and require human monitoring to train and retrain SVM classifiers when the system is updated. Nearest-centroid classifiers, on the contrary, can be easily updated with new knowledge by just adding a new feature vector to the set of training samples and averaging the samples to get the centroid. Rather than constituting a simplistic old-fashioned method, nearest-neighbour approaches are at the core of important advances in computational linguistics (Daelemans and van den Bosch, 2005) and image retrieval (Giacinto, 2007; Jégou et al., 2010). The principle of central tendency leads to an intrinsic settling of centroid models as more examples are added. In case of multimodal distributions, occurring for example when there are multiple writing styles per class, clustering can be used to represent the class variants, e.g., by the *k*-means algorithm. Considering these multiple arguments, in this study, we will use a nearest-centroid classifier for the classification stage, instead of SVMs.

The choice of word-based image retrieval instead of character-based approaches is based, firstly, on the observation that in some historical document collections contractions and loops are used to suggest characters in order to speed up writing (see the marked images in Figure 4.6). This makes creating a mapping between letter identity and character shape non-trivial. Secondly, due to the large variety of scripts and languages, most character-based approaches would need to be fine-tuned for each script and language, leading to long projects to process new collections ("each book its PhD project"). Our goal is to collect huge numbers of labelled word images first over several collections and historical periods in order to develop character-based classifiers at a later stage, when necessary.

Figure 4.6: This variety of styles and shapes in a realistic collection illustrates that 'optical character recognition' of handwriting, by some form of sliding window over a word, is only applicable to a small subset. Many patterns are abbreviations, linguistic contractions or suffer from deformed, 'suggested' characters (marked with asterisks). In the absence of character models, the total-word image on the contrary provides a rich and redundant pattern in all cases, and can be labelled easily by volunteers.

As discussed in the introduction, classification is performed by finding the class with the highest probability given the data. Since nearest neighbour classifiers are distance-based, the class with the highest probability is the class with the smallest distance to the instance:

$$\underset{i}{\operatorname{argmax}}\, P(C_i|X) = \underset{i}{\operatorname{argmin}}\, d(C_i|X) \tag{4.3}$$

Similarly, retrieval is performed by ranking all instances based on their distance to a class-model. Two features were experimentally chosen from a set of features to be used in the experiments. The exact implementation of both features is outside the scope of this article; different feature methods could be used instead without changing the actual re-ranking process. The first feature is based

on the biologically inspired features introduced in (van der Zant et al., 2008a), and the second is a more simple feature consisting of the normalised and scaled image. The dimensionality of the former feature is 4358, while the scaled image has a size of 100×50, yielding a comparable dimensionality of 5000. In both feature types, the feature vector consists of probability values, adding up to one.

Two methods of retrieval will be compared: 1) direct retrieval: ranking, in a single step, all instances from the test set with the distance of the image to the centroid of the target class, and 2) the two stage re-ranking method as described in the previous section: do recognition on all instances first, then for each class C rank its candidates. The re-ranking method can be done in four ways using the two features: recognition with either feature and ranking with either feature. All four combinations are used to study the effect of using a different, secondary feature in the re-rank phase.

There are a number of measures to be used for comparing recognition and retrieval: (a) For recognition, we define top-1 recognition accuracy as: The probability that the nearest-centroid is of the correct class. For retrieval, the standard measures (b) precision and (c) recall will be considered, as well as (d) the *average edit distance* in the top-7 of each hit list.

Accuracy (a) is defined as the percentage correctly classified instances:

$$\text{Accuracy} = \frac{N_{correct}}{N_{total}} \tag{4.4}$$

with $N_{correct}$ is the total number of correctly classified instances (in the top-1), and N_{total} is the total number of instances. We are interested in accuracy because it can show which feature is a good choice for the first stage: features and methods with a high accuracy are well suited for classification.

Precision (b) is defined as the proportion of correctly retrieved instances of class C in a fixed hit list H, with target size n, and can be computed with

$$\text{Precision in top-}n = \frac{N_{correct}}{\min(n, |H|)} \tag{4.5}$$

where $N_{correct}$ is the number of instances with the correct label in the top-n and $|H|$ is the number of items in the hit list[1]. The minimum of n and $|H|$ is used because the hit list can be smaller than the target size of n items.

The recall measure (c) is defined as the proportion of instances of class C that can be found in the hit list; formally, it can be defined as

$$\text{Recall for class } C = \frac{N_{obtained}}{N_{targets}} \tag{4.6}$$

where $N_{obtained}$ is the number of instances retrieved with class C, and $N_{targets}$ is the total number of instances with class C in the given test set. The reported precision and recall are accumulated over all classes as proportions.

The concept of prototypicality cannot be seen in isolation from the application context. More specifically, users of a retrieval engine for historical handwritten words will have an evaluation of the quality of a hit list. In other words, $P(X_j|C)$ must reflect an underlying measure of similarity. In information retrieval, relevance feedback is used to estimate user appreciation (Salton and Buckley, 1997). Relevance feedback is outside the scope of this study, but to estimate the user appreciation, we use *average edit distance* as the fourth performance measure. The assumption is that if the text distance (in 'ASCII') between the query and the actual label of an instance is small, the hit list will be *intuitive*, meaning that it reflects the users measure of similarity well. The specific edit distance implemented in this study is the Levenshtein distance (Levenshtein, 1966).

The data set is drawn from the historical document collection from the Dutch Queen's Office (see also van der Zant et al.,

1 According to the Wikipedia article on precision and recall (http://en.wikipedia.org/wiki/Precision_and_recall, last accessed 23 January 2013), this is also called "precision at n" or "P@n"

2008a), or "Kabinet der Koningin" (KdK). The complete data set has over $13{\times}10^3$ classes. However, in order to do a 7-fold cross-validation experiment, only the 1404 classes with seven or more human labelled word instances will be considered. These classes will be divided into four categories, based on the number of instances: 7 up to 35 instances, 35 up to 60 instances, 60 up to 120 instances and 120 or more instances, similar to what has been done in (van der Zant et al., 2008a). This division is useful to compare performances when there are few labelled instances, a lot of labelled instances or in between. In total, there are more than $84{\times}10^3$ instances used. The experiments are performed on a cluster of eight Linux machines with 54 cores in total, connected to a 1.6 petabyte storage, of which the *Monk* system will use roughly 0.5 petabyte.

For each line strip, a number of word candidates are selected, based on the number and size of connected components. This means that the line is usually oversegmented, which leads to overlap between images. To avoid that multiple image renderings belonging to the same word instance end up in both the training and test set, the fold sets are compiled from exclusive page sets: fold $\equiv$ page number $\pmod{N_{\text{folds}}}$, $N_{\text{folds}} = 7$. This has the additional, realistic benefit that trained words, which are written in a consistent style within one page, but inconsistently over the entire collection will not end up in the test set of a fold. Each fold holds 84 288 instances, of which the test set will hold $1/7^{\text{th}} \mathrel{\hat{=}} 12\,041$ instances on average.

4 RESULTS

We look at two types of comparisons: between re-rank methods (choice of features) and between average re-rank performance and direct retrieval (i.e., without re-ranking). Table 4.2 shows the top-1 recognition accuracy, averaged over all seven folds for both features. Feature 1 (f1) outperforms the second feature (f2), especially in the categories of 35-60 and 60-120 examples. Furthermore, the table shows that to accurately classify an instance, the nearest-centroid classifier needs around 35 training instances.

Table 4.2: Top-1 accuracy ($N_{\text{folds}} = 7$)

Feature	N_{examples}							
	7-35		35-60		60-120		120+	
	Mean	σ	Mean	σ	Mean	σ	Mean	σ
f1	**0.62**	±.02	**0.93**	±.01	**0.92**	±.01	**0.94**	±.00
f2	**0.62**	±.01	**0.86**	±.01	**0.87**	±.01	**0.93**	±.00

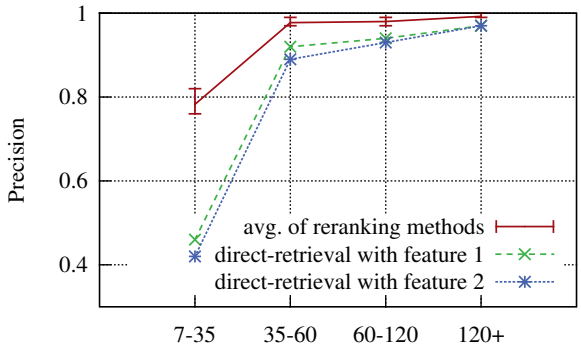
Since feature 1 performs better than feature 2, it seems to be the best candidate for the classification step, as is confirmed below.

Figures 4.7(a), 4.7(b) and 4.7(c) compare the average of the re-rank methods to the direct retrieval methods. The bars on the averages show the minimum and maximum value of the re-rank methods. These results show the gain in performance when using the re-ranking methods instead of direct retrieval. As was expected, reducing the number of distractors has a positive impact on performance.

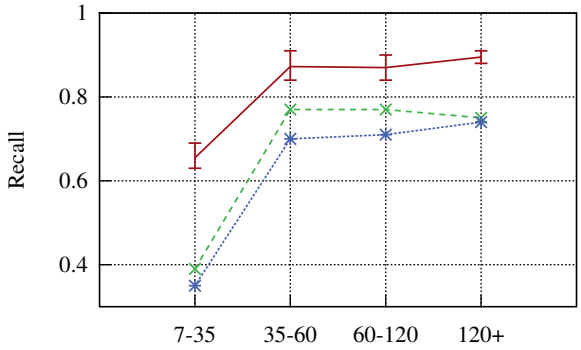
Analogous to Figure 4.5, Figure 4.8 shows the probability of finding the first hit in ranks 0 to r for the re-rank method using feature 1 as the classification feature and feature 2 as the re-rank feature. The re-ranked method shows a considerable improvement from the direct ranking and the ranked SVM output (the best performance as reported in Figure 4.5). The probability of finding the first hit in the first four ranks even approaches 100%.

Table 4.3 and 4.4 show the precision (in the top-1) and recall figures. In general, these results show that re-ranking with a different feature can boost performance. The precision in top-7 for the re-rank methods is even higher than the precision in top-1 for the direct method, especially in the 7-35 category. Using feature 1 as a classification feature and feature 2 for ranking works best for this data collection, even getting a top-1 precision of 1.0 (i.e., 100%) with a standard deviation of 0 in the 120+ category.

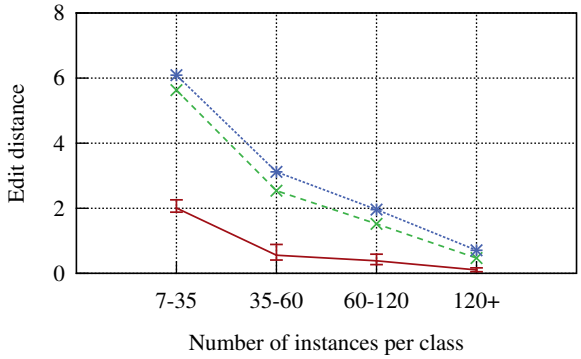
Overall, the results show that all methods perform roughly the same when there are enough labelled samples (i.e., in the 120+ category).



(a) Precision performance in top-1



(b) Recall performance



(c) Average edit distance in top-7

Figure 4.7: Precision and recall performances (at $N \approx 1700$ and $\alpha = 0.01$, confidence is $\pm 3\%$) and average edit distance of re-rank vs. direct retrieval. The bars on the re-rank lines show the minimum and maximum performances of different feature configurations. All measures are averages over 7 folds.

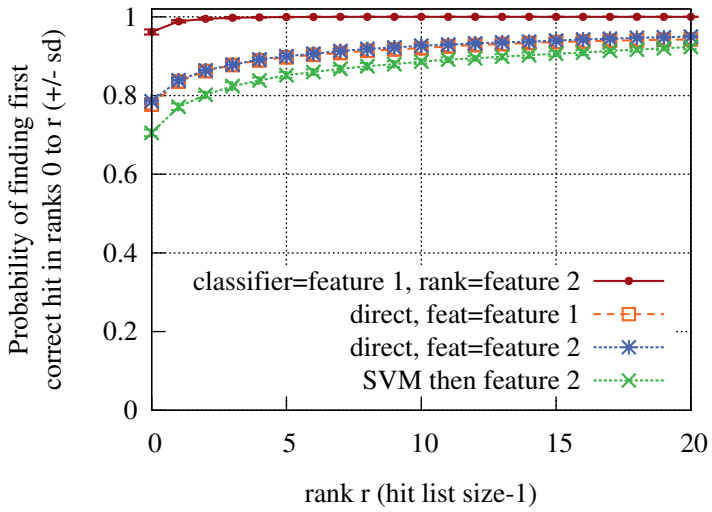


Figure 4.8: Probability of finding the first correct hit in ranks 0 to r for the re-rank method using feature 1 for classification and feature 2 for ranking, and the direct methods ($N_{\text{folds}} = 7$). The bars giving the standard deviations, are barely visible due to the large numbers of test instances in each fold (≈ 1700). The lines for both direct ranking methods are very close together and therefore not distinguishable from each other. The results show a considerable improvement in comparison to the raw, non-reranked results (Figure 4.5), especially for non-ranked SVM: The error at rank 0 is reduced from 29% to 4%, here.

5 CONCLUSIONS

In the design of a large scale retrieval engine for historical handwritten manuscripts it was observed that classifier accuracy is not a good predictor of retrieval precision. Very low precision performances occurred on good classifiers when using a realistic number of distractors. In retrospect, the choice of using the signed distance d_{SVM} from the margin for ranking was evidently suboptimal, but it elucidated two separate functions to be performed: 1) data reduction by optimal *separation* and 2) ranking instances in terms of their *prototypicality* with respect to their class.

The re-ranking method has two main advantages: the focus on both separability and prototypicality increases the probability

Table 4.3: Precision results ($N_{\text{folds}} = 7$, $\sigma \leq 0.03$)

Method	N_{examples}			
	7-35	35-60	60-120	120+
Precision in top-1				
Direct, rank with f2	0.42	0.89	0.93	0.97
Direct, rank with f1	0.46	0.92	0.94	0.97
Re-rank, classify with f2, rank with f2	0.76	0.97	0.98	0.99
Re-rank, classify with f2, rank with f1	0.76	0.97	0.98	0.99
Re-rank, classify with f1, rank with f1	0.79	0.98	0.97	0.99
Re-rank, classify with f1, rank with f2	**0.82**	**0.99**	**0.99**	**1.00**
Precision in top-7				
Direct, rank with f2	0.14	0.52	0.71	0.90
Direct, rank with f1	0.15	0.57	0.75	0.91
Re-rank, classify with f2, rank with f2	0.64	0.87	0.91	0.97
Re-rank, classify with f2, rank with f1	0.68	0.91	0.94	0.98
Re-rank, classify with f1, rank with f1	0.69	0.93	0.94	0.97
Re-rank, classify with f1, rank with f2	0.69	0.93	**0.95**	**0.99**

Table 4.4: Recall results ($N_{\text{folds}} = 7$, $\sigma \leq 0.03$)

Method	N_{examples}			
	7-35	35-60	60-120	120+
Direct, rank with f2	0.35	0.70	0.71	0.74
Direct, rank with f1	0.39	0.77	0.77	0.75
Re-rank, classify with f2, rank with f2	0.63	0.84	0.84	0.88
Re-rank, classify with f2, rank with f1	0.63	0.84	0.85	0.89
Re-rank, classify with f1, rank with f1	0.67	0.90	0.89	0.90
Re-rank, classify with f1, rank with f2	**0.69**	**0.91**	**0.90**	**0.91**

that the top of a hit list is more similar to the user's expectation than otherwise. Secondly, the reduction of distractors lowers the number of noisy instances in a hit list and is advantageous in terms of processing demands. As the results presented in the previous section show, reducing the number of distractors in a retrieval experiment improves precision and decreases average edit distance in the hit list, which we assume will increase the user appreciation of hit lists. We think that a simultaneous solution of separability and prototypicality will suffer from a performance reduction that is typical of Pareto curves in multi-objective optimisation, but this is a matter of future research. To investigate whether we can optimise both separability and prototypicality in the SVM paradigm, we performed some preliminary tests. These

tests show that weighing the discriminant value d_{SVM} with the distance to the centroid of positive examples $e^{-d(\lambda,X)}$ does not have positive effects on precision. Future research will look into other multi-objective approaches involving both separability and prototypicality.

It appeared to be beneficial for retrieval performance to use different features in the separate stages. While the processing order is fixed — separation first, ranking second — the selection of optimal features and machine learning algorithms will depend on the material. In the KdK data set, precision benefited the most by using a strong, robust feature for recognition first, and a secondary feature with a strong image-based component that works well on collections where words are written fairly consistently. On data sets where the writing varies a lot within a class, other features or classifier methods may prove to be more advantageous, including (k-means) clustering to capture the different writing styles. A system like Monk will have several tool libraries and approaches for diverse material. The optimality of the parameters for a complete processing pipeline depends on the ink deposition process, writing style and physical material. Improving the recognition accuracy using linguistic models and contextual information is difficult due to the nature of the material. While linguistic models offer improved transcription performances for contemporary texts, previous efforts of using contextual information (Ritsema van Eck and Schomaker, 2012; Zinger et al., 2009) proved not to be robust enough for use in our system because there are no useful corpora available for the document collections we deal with. This is due to the abundance of abbreviations, contractions and named entities that are not found in corpora of contemporary text. Furthermore, in certain document collections, several languages are used, sometimes even in the same paragraph. Corpora for transcription systems for contemporary texts usually contain millions of words gathered from various sources (Zimmermann and Bunke, 2004; Devlin et al., 2012), which we can not provide for the bootstrapping of handwriting recognition for the document collections in Monk.

When a class has enough instances (i.e., the 120+ category), choice of feature does not seem to have much effect on retrieval performance. On the other hand, reducing the number of distractors by a two-step approach is still beneficial. In the bootstrapping phase of a retrieval system (i.e., the category of 7-35 training examples), the choice of feature does have a big impact. Even small accuracy performance increases have large consequences in this stage, helping the user to label new instances with little effort (since *Monk* presents hit lists in its web-based labelling interface).

The methods presented in this paper can use all kinds of classifiers. Currently, nearest-centroid classifiers are used due to the nature of '24/7' learning, where new labels are being added frequently. It would be cumbersome to retrain classifiers such as SVMs every time a new label was added. The SVM has one benefit in the bootstrap phase: its recognition accuracy is better than the performance of a nearest neighbour classifier. However, the 7-35 category in this experiment has the most classes by far, which would be very inconvenient for the training of tens of thousands multi-class SVMs. This touches on the fundamental difference between SVMs and Bayesian classifiers. While Bayesian classifiers, including nearest centroid classification, will incorporate the retention of the degree of prototypicality in the "1 out of N" choice itself (i.e., $p(d(X, \lambda))$), a tree of SVMs capitalizes on separability, only.

The *Monk* project has a large number of collections with different script types: 15^{th} (mixed languages, frequent use of word contractions) and late 19^{th} century texts (cursive with a lot of abbreviations and variation), Qumran scrolls (isolated characters), captain's logs (cursive) and even Thai (Surinta et al., 2012) and Bangla (Bhowmik et al., 2011) texts. The different shapes and writing styles have different requirements of the features; For each script, features will be selected to optimise both separability and prototypicality.

Summarising, we found that the assumption that a good recognizer will also be good at ranking is not intrinsically tenable. Two requirements need to be fulfilled. First, a method (feature

and classifier) is selected based on its ability to separate class instances from non-class instances. Subsequently, a method (feature and classifier) is selected on the basis of its ability to rank instances according to prototypicality, such that the final ranking is similar to the users expectation. This stepwise approach yielded very substantial improvements in precision, substantial improvements in recall as well as a substantial reduction of the edit distance, a measure of word-match intuitiveness. Finally, the insight that separation and ranking of instances both need to be optimised may have a broad applicability beyond handwriting recognition.

5

GENERAL DISCUSSION

1 MACHINE LEARNING AND REPRESENTATION

The subject of this dissertation is the development of search engines for historical handwritten document collections. Techniques such as machine learning and image representation are often being studied in the field of handwriting recognition to improve the accuracy of handwriting recognition systems. Usually, in order to have fair comparisons, standard datasets are used for the performance evaluation. While most techniques work very well on datasets such as MNIST or the letters by president George Washington, their application is much less straight-forward on collections with more difficult material such as those of the alderman scrolls of the city of Leuven, as discussed in Chapter 1.

The Monk system (Schomaker, 2016) is a search engine and data mining tool for historical document collections. Many different collections are available in Monk, ranging from rather neat collections, written by a single writer, to very difficult-to-read collections from medieval times. These collections are usually interesting to humanities researchers or even the general public. Due to the nature of these collections, we have observed some notable differences between applying machine learning techniques on the neat, academic datasets, and the raw datasets that are being made public by archives worldwide (see for example Figure 1.2 on page 3 and Figure 4.6 on page 67).

One of the techniques we observed with different results between academic benchmarks and historical material is hidden Markov modelling (Chapters 2 and 3), which has a long history in handwriting recognition systems as well as other applications. HMMs model time series using a hidden state transition probability matrix and per state a model of the observation probabilities. The application of HMMs on the difficult datasets in Monk has not resulted in accuracy scores that are expected based on the literature. We performed two studies to understand the differences in performance. The first study deals with the training method while the second study concerns the understanding of the different components of these models.

1.1 Baum-Welch training of HMMs

The process of training HMMs with Baum-Welch is well known for its tendency to get stuck in local optima, resulting in less-than optimal models. The idea that sparked the study in Chapter 2 is that the initial model, which is randomly selected to start the training, has a big impact on the final performance. We therefore studied whether models converge to the global, rather than a local, optimum if the starting point is already closer than other, non-converging models. To study the training procedure, we generated an artificial data set: A randomly chosen model with known properties generated data sequences on which new HMMs can be trained. The model that is used to create the dataset is then considered the global optimum for this particular set and can be compared to the trained models.

The main conclusion from the experiments in Chapter 2 is that the χ^2 distance of a trained model to the global optimum is not a good predictor of likelihood. At first glance, this is surprising because one would expect that models that are similar to the global optimum would also show a performance similar to the model at the global optimum. Furthermore, the experiments show that it is very hard for the Baum-Welch training algorithm to converge to a point close to the global optimum without guidance. However, when we know either the state transition probabilities or the observation probabilities beforehand, the

resulting models are very close to the global optimum. This is relevant because it shines some light on the conditions in which HMMs will perform well.

1.2 The relative importance of transition vs. observation probabilities

When we know either the state transition probabilities or the observation probabilities, we can "clamp" these parameters, which means that they are fixed while still training the other properties. This greatly reduces the number of parameters to be learned. The fact that clamping has a big positive effect on the ability of the training algorithm to find the global optimum, raises an interesting question: What is the most critical design element of an HMM? The transition probability matrix A or the observation probabilities B? This is the main question in Chapter 3, and is also studied by using generated data from a known model. This time, the model has a very specific topology: The Bakis topology dictates a diagonal structure of the transition matrix, leaving most probabilities in the matrix $P_{ij} = 0$. It is shown that the Baum-Welch training algorithm has difficulty finding this clear diagonal structure. Furthermore, using "real" data, extracted from handwritten word images, it has been shown that removing all temporal information (and clamping the transition matrix to a uniform distribution) from the models, does not result in as drastic a drop in performance as one might expect.

The main conclusion from Chapter 3, that the observation probabilities seem to have a larger impact on the model performance than the transition probabilities, has some interesting implications for a search engine for handwritten documents. It means that special attention should be given to the representation of word images. One of the main challenges is the bootstrapping phase of a collection, where there are very few labelled instances. In these cases, it is very hard to automatically learn the best representation of a word image, let alone train both representation and classification at the same time like in deep learning. It makes sense to create the feature representation and selection methods by hand in cases where there are very few labelled images.

2 LABELS

The discussion in the first two chapters has been around machine learning and representation. We have discussed why it is not straightforward to learn the parameters of a Markovian process and that the representation of handwritten word images is important for HMMs. However, in order to learn from known examples and to generalize to unseen examples, it is important to have a large amount of labelled images, i.e., tuples of $(\textit{class label}, \textit{word image})$. Collecting these tuples can be quite labour-intensive. Frequently, machine learning experts presume the existence of labelled data and only focus on either the (deep) machine learning or the feature representation of the word images. Labelled data has not been a big problem in the studies on HMMs in Chapters 2 and 3 of this dissertation because the data was mostly generated artificially, and the classes were therefore known. However, the extensive datasets of handwritten word images had to be labelled by hand to create a proper ground truth.

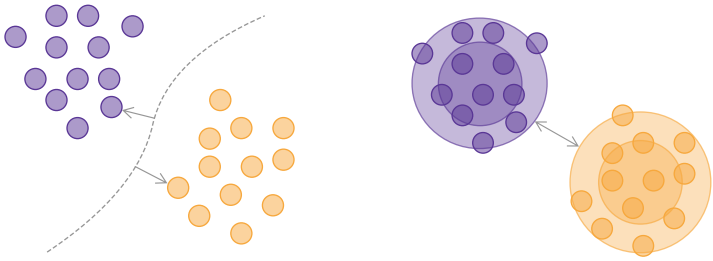
In a system such as Monk, new datasets are added on a regular basis. These new datasets do not contain labels yet and are very diverse in script-type and picture quality, making it difficult to build of off other document collections. Transfer learning (Pan et al., 2010) would generally be useful in these cases, as long as the source and target domains are *related*, i.e., the feature spaces are not too different. However, the differences between collections in the Monk system are quite large. This means that it is required to 'bootstrap' new collections: starting with a small number of labels and quickly building the necessary body of knowledge about a collection. By aiding the human annotator, it should be possible to gain momentum in the labelling process. Quick accumulation of new labels, often in a group of related classes, also known as a snowball effect, is something that is implemented in the core processes of the Monk system. The snowball effect is usually marked by sudden jumps in the number of added labels (see Figure 1.8, page 14.)

In Chapter 4, a solution for the problem of bootstrapping and quickly building a large database of labelled data is explored

that uses hit lists. This works by generating a list of images a classifier determines to belong to a particular class. The system then ensures that the images that are most likely to be correct are ranked at the top. The annotator uses this fact in the hit-list interface to label yet unknown images quickly by marking the top n images to be correct. This method has been shown to allow the sudden jumps in number of added labels. Correcting a label provides a large amount of new knowledge to the system as well: Correcting mistakes either suggests new classes or exposes a confusion between two existing classes.

Another method for selecting instances that should be labelled for the largest impact on the classification performance is active learning (Settles, 2009; Baum and Lang, 1992). The selection process in active learning is based on the idea of finding the images for which the classifier has the least evidence of belonging to a certain class (i.e., has the most confusion). This works well for discriminative learning, because at the decision boundaries, new knowledge has the biggest impact. However, a different approach might be better suited for Bayesian or generative methods, where each class is represented by its own model and the classification is solved by a 'winner takes all' principle (usually by applying the argmax function on the probabilities per class). In this case, the better a single model represents samples of that class, the higher this model will end up in the final ranking. See also Figure 5.1. This is related to the concept of density weighting in active learning (Settles and Craven, 2008) where the instances to be queried are weighted by their distances to all other known instances, but this does not necessarily take the class into consideration.

Hit lists have been effective in labelling new word images. A few guidelines can be established for building an efficient snowball effect. First, the images should be ranked from most likely to least likely belonging to the class of the hit list. This ensures that the top n instances can quickly be assessed on their correctness. Secondly, the interface should allow for quickly labelling a large number of images at once, preferably by selecting the first n images and accepting the label the classifier has assigned to these



(a) Discriminative models (b) Generative models

Figure 5.1: Discriminative models (a) are separated by a decision boundary. The objective is to get this boundary as accurate as possible. Active learning works well here because it samples around the current decision boundary. Generative models (b) consist of multiple models for each class, in this case represented by a Gaussian. Applying active learning here would mean sampling around the area between the classes, skewing the distribution per class.

images. Thirdly the images in the hit list that are not of the correct class, should be 'intuitive', such that they are not totally different from the correct instances of that class. Apart from the difficulty in explaining very obvious mistakes to the user, labelling mistakes have a much smaller impact when the differences are small. Furthermore, classes that are related to each other can then be spotted and labelled as such more easily. Using a hit list-based approach is preferable to a linear process, where word images are annotated from left to right, top to bottom, because a hit list allows the user to inspect many occurrences of a word in a single screen. This aids the recognition of misclassifications and allows the annotator to scan instead of typing in every label by hand.

In order to get hit lists with quality results in the top ranks, a two-step process is needed. This process alternates between classification and ranking. Each phase can use their own machine learning and representation methods. The two-step process, together with an increasing amount of labelled data, will improve the hit lists over time and yield even more useful labels. This way, we can ensure that the right method for the right job is used, but also that once one method does not yield enough

new labels any more, a different method can be used instead. This interplay between different feature extraction methods and iterated application of building hit lists can be compared to the Fahrkunst elevator[1] in ancient mines, which uses an alternation of steps to get at a higher level.

3 LOOPS AND SNOWBALLS, NOT PIPELINES

The observation of the snowball effect, marked by sudden jumps in number of added labels, leads to the idea that handwriting recognition should not be a simple, one-directional pipeline, but a loop. This loop has been described in Chapter 1 (see Figure 1.6 on page 10). By defining the handwriting recognition process as a pipeline, the larger feedback loop is ignored, and it will be harder to bootstrap new, challenging manuscripts with a new script type or different vocabulary. This means that even though Machine Learning and the representation of the word images is very important, all elements, including the ground truth, are important and should be considered in any system that uses handwriting recognition techniques for retrieval or classification.

In this dissertation, we have studied the different aspects of the handwriting recognition loop. We can consider the effect of human feedback on these aspects. The first aspect we studied is the algorithmic level in Machine Learning. Human feedback here improves the algorithm, which can be applied on many different problems. However, the improvement is usually tested on specific use-cases or on standard academic datasets (e.g., the letters by President Washington or the MNIST handwritten digit dataset). Furthermore, the results in Chapter 3 suggest that an improvement to the overall system performance may be achieved more efficiently by focussing on representation, rather than learning the underlying process parameters. Currently, a lot of studies are focussing on deep learning methods, that are especially interesting because these methods train both classifiers as well as representation. These methods will be discussed in Section 4 below.

1 Also known as the *man engine*.

The representation of the data is the second aspect that we studied where human feedback can be applied. Improvements in this area are often tightly coupled to the dataset that they are developed for. This frequently leads to the phenomenon of 'One PhD, One book' where the efforts of, e.g., a graduate student leads to accuracy improvements for only a single book or at best a single collection. However, human effort on this aspect does have a large impact on model performance, especially because it is sensitive to the book or collection that it is applied to. We believe there is still a need of handcrafted features: The two-step process discussed in Chapter 4 allows changing the feature extraction method to apply the best method for a collection or even a class, in order to 'harvest' the most new labels. Transfer learning (Pan et al., 2010) is a very interesting technique that might be very relevant for achieving high accuracy on new datasets, however the datasets in Monk can be very different from one another, in handwriting style as well as the material (such as the paper, ink deposition method, etc.), and may warrant a different representation method.

The third and final aspect of human feedback that we discussed is that of providing labels. Labels are what drives the handwriting recognition loop. Any effort to improve labels, improves the knowledge of the system as a whole. The more knowledge is embedded in a system, even more and better algorithms can be applied. This means that different techniques, both at machine learning as well as the representation level, should be used for new collections than for collections that have many more labels, classes and writing styles available. This knowledge is even more tied to the collection than representation is, even though some efforts have been made to use the knowledge of one collection for bootstrapping another. The general principle, as well as any techniques to apply these ideas, can be applied to any collection.

4 DEEP LEARNING

The topic of Deep Learning, or techniques related to it, was outside the initial scope of this dissertation. However, since a couple of years Deep Learning has gathered much support in the

community of AI and Machine Learning. Rightfully so, given the breakthroughs and number of competitions that have been won by Deep Learning methods. For example, in (Sanchez et al., 2016) all the participants used a deep learning method in one shape or form. The first major competition where Deep Learning made a significant jump in performance was in the ImageNet competition (Krizhevsky et al., 2012). Given the popularity of Deep Learning, it is a subject that should be addressed in this dissertation as well.

The core topic of this dissertation is the general feedback loop in a search engine for handwritten document collections. In Chapter 4 and this chapter, we have made the case for a method that is relatively independent of the specific machine learning or representation method. Deep learning seems like it would fit in this framework, but it also seems that it is an opposing framework: Deep Learning trains both classifier and representation simultaneously.

There is much excitement surrounding deep learning since the results and quality of the models are very impressive. However, there are also reasons to be cautious. The downside is that you need a large amount of labelled data. Furthermore, the fact that you do not have to create a separate representation method does not imply that there is no "manual labour" needed any more. Designing the network architecture and tuning all the hyperparameters is still largely done by hand and can be very labour intensive. Also, training a character-based classifier is susceptible to the same pitfalls as those mentioned in Chapter 4 (see for example Figure 4.6 on Page 67).

The increasing interest in Deep Learning models however seems to underscore the conclusions from Chapters 2 and 3, that using HMMs for handwriting recognition purposes is not straightforward and may have fundamental issues. Figure 5.2(a) shows the relative interest in HMMs versus LSTMs on Google: Since 2016 the number of search queries for LSTMs has overtaken the number of queries for HMMs. Both are methods for modelling time series, where the main advantage of LSTMs is that they allow for a longer history to be used in calculating the probability of the

next observation. Furthermore, the Deep Learning community is very effective in transferring new ideas in one field of study to others. For example, the attention mechanism for LSTM networks (Bahdanau et al., 2014; Doetsch et al., 2016) allows even more precision in aligning the inputs to each part of the output sequence, even though it is not necessarily developed for the handwriting recognition community.

When there is enough training data to train a combined classifier and representation method, the results tend to exceed the more traditional methods where representation is hand-crafted and classifiers such as SVMs or HMMs are used. However, the feedback loop will be a lot slower: The large training set and large number of model parameters make the training of an LSTM or Convolutional Neural Network for example, computationally expensive. The flexibility of creating separate representation and classification methods is especially helpful to gain "phase transitions" in the feedback loop.

Finally, the introspection methods described in Chapters 2 and 3 can, with some adjustments, also be applied to Deep Learning methods. Opening black boxes of classifiers is gaining popularity, especially to build trust in the methods. The LIME method (Ribeiro et al., 2016) aims to be a classifier-agnostic method. There are a number of introspection methods for neural networks as well (e.g., Olah et al., 2018; Barratt, 2017). It still seems interesting to look at deep learning methods from a global perspective, similar to the way we have looked at the HMMs. This would help with a better understanding of strengths and weaknesses of the backpropagation training method.

5 CONCLUSION

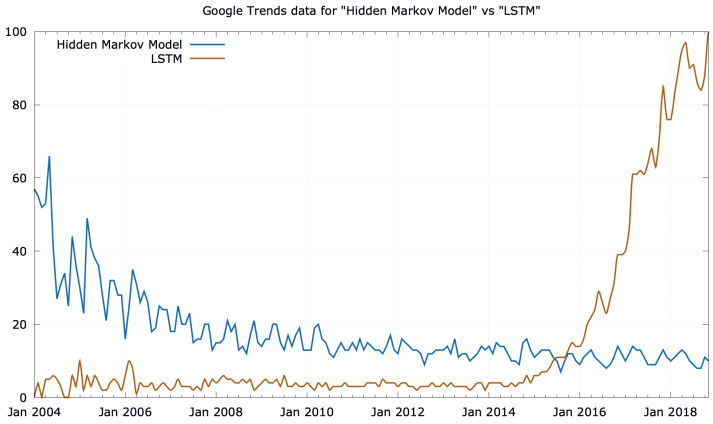
The discussions in this chapter on the different aspects of handwriting recognition do not necessarily only apply to the handwriting recognition field; The conclusions can have implications across disciplines and methods. A focus of this dissertation is the inclusion of the labelling of images in the entire process, and to consider this process to be a loop instead of a fixed pipeline.

Labelled data is important—for both scientific research as well as industry—because it drives the feedback loop and therefore allows for improvements in both accuracy and the methods, either machine learning or representation.

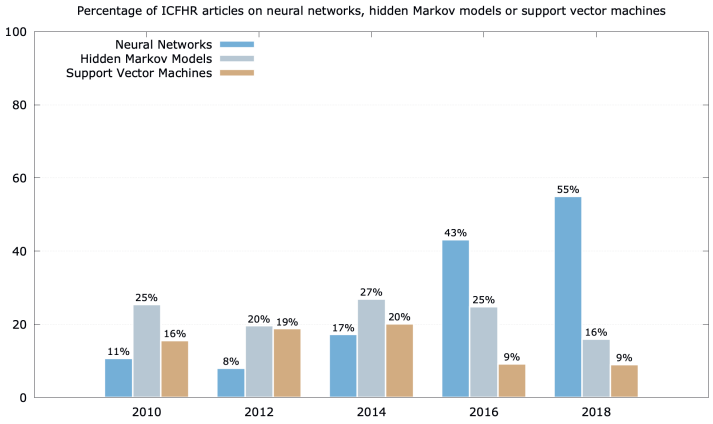
One thing that we have noticed while working on the Monk system is that exploration is very useful. The Monk system uses this idea to trigger phase transitions by switching ranking and classification methods. This is especially useful when a certain combination of these methods is no longer effective in getting new labels from annotators. At the same time, we have seen that the diversity in scientific research has declined: Deep Learning, or neural networks in general, has been gaining popularity. A quick study of abstracts and keywords of the 2018 edition of the ICFHR conference shows that 55% of the papers mention something related to deep learning or LSTM, up from 17% in 2014, see Figure 5.2(b). However, we believe that the field in general benefits from exploration as well.

Exploration can take many forms. From the Monk system, we have seen the idea of exploring different classification and ranking methods and Chapter 4 shows that it is useful to separate these two functions. Another parameter to explore is the representation of the images. A deep learning method is usually not ideal for these separate stages since the representation and classification method are usually deeply interconnected, and because retraining takes a lot of time. However, the Monk system still allows a user to select different methods to use in the sorting of hit lists. Finally, inspiration can also be taken from biology. While the idea of a neural network is biologically inspired, the implementation is usually very engineering oriented.

An interesting biologically inspired approach can be found in Hawkins and Blakeslee (2007); Hawkins et al. (2018). Hawkins is working on a dual mission to both "reverse engineer" the neocortex, as well as creating software based on the current level of understanding of the neocortex. The approach is interesting because it is an exploration of unsupervised learning of hierarchical representations and predictions. It would be very interesting to see these theories applied to handwriting recognition, as a way



(a) Google Trends data for HMMs and LSTMs



(b) Percentage of ICFHR abstracts on several topics

Figure 5.2: (a) Comparison of the number of searches on Google on "Hidden Markov Model" versus "LSTM". The *y*-axis represents the number of searches relative to the highest number of searches in a given month—in this case: November 2018 with the highest number of searches for "LSTM". Data by Google Trends. (b) Percentage of abstracts and keywords in ICFHR articles that mention either a) LSTM, CNN, Deep Learning or Neural Networks, b) HMMs, or c) SVMs.

of exploring a different direction from gradient descent-based methods.

Besides a focus on methods and tools, there can be feedback on other aspects of the writing as well. An interesting future research area would be to focus on giving feedback on a sentence, page or even book level. Feedback on these levels would help with identifying relations between words and their position. Feedback on semantics would help in understanding rather than just classification.

The main ideas of this dissertation are a) that human feedback can be injected in several aspects of the pipeline, b) that we should consider handwriting recognition to be a loop instead of a pipeline and finally, c) that by taking advantage of the loop, a snowball effect can be achieved. Therefore, the advice is to invest in a system of getting more and better labels in order to increase the gain in the feedback loop, which has a positive effect on both machine learning and (learned) representations as well.

While this dissertation focussed on techniques such as HMMs and SVMs, using the described framework, it is possible to collect enough labels to further study "label-hungry" methods such as deep learning because this framework is method agnostic. We even believe that this advice is not limited to the handwriting recognition field. Having proper, labelled data is crucial to many machine learning and pattern recognition applications across many fields and industries.

APPENDIX

Implementing an HMM framework from scratch is not trivial. The canonical paper by Rabiner (1989) contains all the theory necessary, but it may require some extra considerations to make implementation easier. We will give some of these considerations here. The approach used for implementing jpHMM is taken in part from A. Rahimi (2000)[1].

Scaling forward and backward variables

The first issue to address is scaling the forward and backward variables $\alpha_t(j)$ and $\beta_t(j)$. The forward variable is the probability of the partial observation sequence up to time t and being in state S_j at time t, given the model λ: $\alpha_t(j) = P(O_1 O_2 \cdots O_t, q_t = S_j|\lambda)$. The backward variable is the probability of the partial observation sequence from time $t+1$ to time T, given state S_j at time t and the model λ: $\beta_t(j) = P(O_{t+1} O_{t+2} \cdots O_T|q_t = S_j, \lambda)$.

These variables need to be scaled to avoid problems with floating point representations in code. Since the forward variable $\alpha_t(j)$ usually consists of many products of transition and observation probabilities, they tend to approach 0 quickly. On a computer, these variables are bound by a finite precision floating point representation.

A scaling can be applied to both $\alpha_t(j)$ and $\beta_t(j)$, to keep the calculations in range of a floating point representation. Rabiner proposes to use the scaling factor $c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}$, which is independent of state. This means that $\sum_{i=1}^{N} \hat{\alpha}_t(i) = 1$. Both $\alpha_t(j)$ and $\beta_t(j)$ are scaled with the same factor, c_t.

1 Please find Rahimi's solution at http://alumni.media.mit.edu/~rahimi/rabiner/rabiner-errata/rabiner-errata.html, accessed January 23, 2014.

The recursion formulae defined by Rabiner are theoretically correct, but hard to use for implementation because it is unclear that one needs to use the scaled $\hat{\alpha}_t(i)$ in the computation for c_{t+1}. Rahimi therefore proposes the following computation steps:

$$\begin{aligned}
\overline{\alpha}_1(i) &= \alpha_1(i) \\
\overline{\alpha}_{t+1}(j) &= \sum_{i=1}^{N} \hat{\alpha}_t(i) a_{ij} b_j(O_{t+1}) \\
c_{t+1} &= \frac{1}{\sum_{i=1}^{N} \overline{\alpha}_{t+1}(i)} \\
\hat{\alpha}_{t+1}(i) &= c_{t+1} \overline{\alpha}_{t+1}(i)
\end{aligned}$$

Rabiner leaves out the full steps to compute $\hat{\beta}_t(i)$. We can use the following (also from Rahimi):

$$\begin{aligned}
\overline{\beta}_T(i) &= \beta_T(i) \\
\overline{\beta}_t(j) &= \sum_{i=1}^{N} a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(i) \\
\hat{\beta}_t(i) &= c_t \overline{\beta}_t(i)
\end{aligned}$$

We can express the probability of a sequence given a model using $P(\mathbf{O}|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t}$, but since this is also a product of probabilities, we are better off using the sum of log probabilities: $\log[P(\mathbf{O}|\lambda)] = -\sum_{t=1}^{T} \log c_t$.

Multiple observation sequences of variable duration

While implementing the reestimation formulae for multiple observation sequences of variable duration, we ran into the problem of requiring $P(\mathbf{O}^{(k)}|\lambda)$, where $\mathbf{O}^{(k)}$ is the kth observation sequence. We can no longer compute this, because we now use log-probabilities. However, we can rewrite these formula to no longer use $P(\mathbf{O}^{(k)}|\lambda)$. The full derivations are left out, but are essentially the same as those by Rahimi. We will also show the reestimation formula for π, because both Rabiner and Rahimi do not mention it. They assume a strict left-right model, such as Bakis, where $\pi_1 = 1$ and $\pi_i = 0$ for $i \neq 1$.

We will use the following equalities:

$$\prod_{s=1}^{t} c_s^k = C_t^k$$
$$\prod_{s=t+1}^{T_k} c_s^k = D_{t+1}^k$$
$$\prod_{s=1}^{T_k} c_s^k = C_t^k D_{t+1}^k = C_{T_k}^k$$
$$\frac{1}{\prod_{t=1}^{T_k} c_t^k} = \frac{1}{C_{T_k}^k} = P(\mathbf{O}^{(k)}|\lambda)$$

where c_t^k is the scaling factor $\frac{1}{\sum_{j=1}^{N} \overline{\alpha}_t^k(j)}$. Because we now have a new way of representing $P(\mathbf{O}^{(k)}|\lambda)$ as $\frac{1}{C_{T_k}^k}$, we can substitute that into the reestimation equations, leading to the following equations after some rewriting:

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i) a_{ij} b_j(O_{t+1}^{(k)}) \hat{\beta}_{t+1}^k(j)}{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i) \hat{\beta}_t^k(i) \frac{1}{c_t^k}}$$
$$\overline{b}_j(\ell) = \frac{\sum_{k=1}^{K} \sum_{t \in [1,T_k-1] \wedge O_t = v_\ell} \hat{\alpha}_t^k(i) \hat{\beta}_t^k(i) \frac{1}{c_t^k}}{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(j) \hat{\beta}_t^k(j) \frac{1}{c_t^k}}$$
$$\overline{\pi}_i = \frac{\sum_{k=1}^{K} \hat{\alpha}_1^k(i) \hat{\beta}_1^k(i) \frac{1}{c_1^k}}{\sum_{j=1}^{N} \sum_{k=1}^{K} \hat{\alpha}_1^k(j) \hat{\beta}_1^k(j) \frac{1}{c_1^k}}$$

For the full details and derivations of the reestimation equations, please see the explication by Rahimi or contact the authors of this study. The documented code for jpHMM will be published on-line soon.

SUMMARY

Handwriting recognition is an active field of research, even though today our writing is mostly done digitally. There is a large number of archives that contain vast collections of handwritten documents, often in script styles that are hard to read for most people. Searching, and finding relevant pages, is a manual and tedious process.

In the field of handwriting recognition, many researchers use standardized benchmarking data sets to develop the machine learning and pattern recognition techniques and to compare their results to others. However, these datasets are usually very clean and are not comparable to the noisy quality of the historical handwritten document collections in archives and national libraries. When applying the techniques from research to such problematic collections, a number of hidden assumptions that are usually entertained by researchers in Machine Learning become apparent. Such assumptions are discussed in this thesis, along with a number of issues that were encountered in the application of machine learning techniques within a large-scale search engine for historical documents: Monk.

One of the assumptions is that the handwriting recognition process is usually considered a linear pipeline consisting of feature extraction and machine learning[1]. A ground truth for the data is generally presumed by researchers, and therefore not part of these pipelines. When designing a search engine, the challenging part was integrating the labelling into the process. Line-by-line editors of annotations are not ideal: A text line is not a natural object for search and starting from the first page moving down line by line does not use the full potential an integrated model has to offer. Therefore, we propose a more data-mining oriented

1 Segmentation and other pre-processing steps are outside the scope of this thesis

approach that uses a hit-list interface to gather labels per word image.

A data-mining approach to labelling images allows a human user to have a direct impact on the performance of the entire handwriting recognition process (as depicted in Figure 1.6, on page 10). There are several aspects where humans have an impact on the process: (a) On the machine learning methods, (b) on the feature engineering and (c) on the labelling. These aspects were studied in more depth in each of the chapters of this thesis. For each aspect, we looked at the issues that come up during the design of a search engine in order to answer the main question: Where can one have the most impact on the quality of the results of a search engine for historical handwritten documents: By improving the machine learning methods, the feature engineering methods or the labelling methods?

Chapter 2

Chapter based on
van Oosten, J.-P. and Schomaker, L. (Submitted). Examining common assumptions about the convergence of the Baum-Welch training algorithm for hidden Markov models. *Journal of Machine Learning Research*

In Chapter 2, we examined a number of assumptions related to the machine learning aspect of the handwriting recognition process. We were especially interested in assumptions about convergence in the training algorithm for Hidden Markov Models (HMMs), since HMMs have played such an important role in handwriting recognition. The main assumptions that were studied in this thesis are related to the fact that the Baum-Welch training method converges to a local optimum.

The first assumption is that the closer a model is to the global optimum, the better the method will perform. This was studied by generating data with a known global optimum, and training many models on this generated data. We could then measure

the distance between the models and the global optimum and measure the performance in terms of log-likelihood. Surprisingly, this experiment showed us that the (χ^2) distance to the global optimum is not a good predictor of likelihood of a trained model. One would expect that models closer to the global optimum would also have a better performance.

The other main assumption that we tested in Chapter 2, is that models that are already close to the global optimum (Baum-Welch starts by picking a random starting point and trains from there) will most likely end up close to the global optimum. However, we found that it is hard for models to converge to a point close to the global optimum without guidance.

Chapter 3

Chapter based on
van Oosten, J.-P. and Schomaker, L. (2014a). A reevaluation and benchmark of hidden Markov models. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 531–536. IEEE

In Chapter 3, we continued studying HMMs, but zoomed in on the essential elements of the models. We were mostly interested in the relation between the state-transition probabilities, that model the temporal structure of the data, and the observation probabilities, that model the feature representation of the data per state. The main assumption that is tested in this chapter is that the temporal structure is as important as the feature representation.

We studied the relation between the two parts by generating data with a particular temporal structure and trained a model on this data. This structure should be present in the state-transition probabilities in a trained model. However, the experiments in Chapter 3 showed that there is no clear indication that the original structure could be found. Another experiment that was performed in Chapter 3 is removing the temporal relation between

states and observe whether or not classification performance dropped in these models. Surprisingly, the performance did not drop as drastically as expected.

The main conclusion that we can draw from these experiments is that the observation probabilities seem to have a larger impact on the model performance than the transition probabilities. This means, related to the general research question, that special attention should be given to the feature representation.

Chapter 4

Chapter based on
van Oosten, J.-P. and Schomaker, L. (2014b). Separability versus prototypicality in handwritten word-image retrieval. *Pattern Recognition*, 47(3):1031–1038

Finally, in Chapter 4 we turned our attention to the labelling part of the handwriting recognition process. We consider it an essential part of the process and explicitly integrate it into a continuous loop. The hit-list interface is introduced in this chapter and it helps gather a large amount of training data by achieving a snow-ball effect (i.e., an initially small number of labels can accumulate more and more labels over time). A hit-list is constructed by classifying words into the different lists and then ranking each list. We found that one cannot assume that a good recognizer will also be good at ranking.

Related to labelling, we found that it is important to consider the construction of your dataset as part of the process and integrate human annotators in a continuous learning cycle. An implication from Chapter 4 is that one should alternate between classification and ranking and use different methods that are optimized for each subtask. Also, the specific classification and ranking methods should not be considered as fixed. It is necessary to alternate between these, e.g., if the current method does not yield enough new labels anymore to keep the momentum going. The final conclusion from this is that the handwriting recognition process

is not a static process, a single training event, but needs constant maintenance.

Discussion

To conclude, this thesis discusses human involvement in the handwriting process from three different angles: In the design of machine learning methods, design of feature extraction methods and representations, and labelling. Chapters 2 and 3 mainly deal with assumptions around the machine learning and feature extraction methods, while the main concern in Chapter 4 is about how to deal with a changing dataset, especially with continuous additions of labels.

The main method of examining the assumptions in the use of HMMs is the generation of data from known models, and to study what happens in the models during training. Taking a global perspective to study what happens in local (gradient descent) processes is a method that can be used to study other machine learning methods as well, such as neural networks. If the models themselves can be used to generate data, it is relatively easy to compare the trained model versus the global optimum.

The bigger theme in the thesis is about the idea that we need to regard the handwriting recognition process as a dynamic process. In the Monk system, this is expressed in a flexible hit list interface. While the hit list method is only applied to handwritten words in this thesis, we believe that this form of active learning is relevant for machine learning in general. Ideally, the beneficial effect of a labelling action is experienced as soon as possible by the user. This creates a snow-ball effect in the feedback loop and leads to broadly labelled datasets. Another benefit of the hit list interface is that it allows exploration. Using different classification and ranking methods is useful when the addition of labels is stagnating.

It is crucial for all machine learning methods to have proper, labelled data. Therefore, the framework described in this thesis is relevant for all applications of machine learning across fields

and industries. The advice in this thesis is therefore to invest in a system of getting more and better labels and to incorporate this framework into any application of machine learning.

SAMENVATTING

Handschriftherkenning is een actief onderzoeksgebied, ondanks het feit dat tegenwoordig de meeste tekst digitaal geproduceerd wordt. Grote hoeveelheden handgeschreven teksten zijn opgeslagen in archieven zoals het Kabinet van de Koning in het Nationaal Archief. Vaak zijn dit manuscripten die in ouderwetse, lastig te lezen, handschriften zijn geschreven. Het zoeken naar, en vinden van, relevante pagina's is een handmatig en tijdrovend proces.

In het vakgebied van handschriftherkenning gebruiken veel onderzoekers standaard datasets voor het ontwikkelen en vergelijken van hun patroonherkennings- en machine learning-technieken. Deze datasets zijn echter doorgaans voorbewerkt, en niet te vergelijken met de kwaliteit van de historische, handgeschreven collecties in de archieven en nationaal bibliotheken. Wanneer de technieken uit het vakgebied worden toegepast op dit soort moeilijk materiaal, komen een aantal verborgen aannames naar voren die vaak door de onderzoekers gedaan zijn. Deze aannames worden onderzocht in dit proefschrift, samen met een aantal uitdagingen die naar voren kwamen bij het toepassen van machine learning-technieken in een grootschalige zoekmachine voor historische documenten: Monk.

Een van de aannames is dat het proces van handschriftherkenning vaak als lineair wordt beschouwd, bestaande uit het extraheren van kenmerken en machine learning[1]. Onderzoekers gaan er over het algemeen van uit dat er al een "grondwaarheid" (*Ground Truth*) beschikbaar is. Het maken en beheren van zo'n dataset maakt daarom vaak geen deel uit van het proces. Een onderdeel van dat maken van een dataset is het labelen (plaatjes voorzien van labels die aangeven welk woord er geschreven staat). Het

1 Segmentatie en andere voorbewerkingsstappen worden in dit proefschrift niet expliciet behandeld

was een uitdaging om het labelingsproces te integreren in de bouw van een zoekmachine. Het regel voor regel annoteren is niet ideaal: men zoekt doorgaans niet op een volledige tekstregel. Bovendien wordt het volledige potentieel van een geïntegreerd model niet benut als er pagina voor pagina, regel voor regel geannoteerd wordt. Daarom stellen we een aanpak voor die op datamining is gebaseerd en gebruik maakt van een "hit list" interface om labels per woordbeeld te verzamelen.

Een datamining-benadering voor het labelen van afbeeldingen maakt het voor mensen mogelijk om direct impact te hebben op de prestaties van het volledige handschriftherkenningsproces (zoals afgebeeld in Figuur 1.6 op pagina 10). er zijn verschillende aspecten waar mensen een impact kunnen hebben op het proces: (a) Op de machine learning-methoden, (b) op de feature engineering en (c) op het verzamelen van labels. Deze aspecten worden in elk van de hoofdstukken van dit proefschrift onderzocht. Voor elk aspect hebben we gekeken naar de problemen die naar voren kwamen bij het bouwen van een zoekmachine. Het doel was om de hoofdvraag te beantwoorden: Waar kan men de meeste impact hebben op de kwaliteit van de resultaten van een zoekmachine voor historische handgeschreven documenten: Door het verbeteren van de machine learning-methoden, de feature engineering-methoden of de label collectie-methoden?

Hoofdstuk 2

> Dit hoofdstuk is gebaseerd op
> van Oosten, J.-P. and Schomaker, L. (Submitted). Examining common assumptions about the convergence of the Baum-Welch training algorithm for hidden Markov models. *Journal of Machine Learning Research*

In Hoofdstuk 2 hebben we een aantal aannames bestudeerd die betrekking hebben op het Machine Learning-aspect van het handschriftherkenningsproces. We waren in het bijzonder geïnteresseerd in de aannames over convergentie in de trainingsalgoritmen voor Hidden Markov Models (HMMs), aan-

gezien HMMs zo'n belangrijke rol hebben gespeeld in handschriftherkenning. De belangrijkste aannames die onderzocht zijn in dit proefschrift hebben betrekking op het feit dat de Baum-Welch-trainingsmethode naar een lokaal optimum convergeert.

De eerste onderzochte aanname is dat hoe dichter het model bij een globaal optimum ligt, hoe beter het zal presteren. Dit was bestudeerd door data te genereren met een bekend globaal optimum, en veel verschillende modellen te laten trainen op deze gegenereerde data. Zo konden we de afstand tussen de getrainde modellen en het globale optimum meten, evenals de prestatie in termen van log-likelihood. Verrassend genoeg heeft dit experiment ons laten zien dat de (χ^2) afstand tot het globale optimum geen goede voorspeller is van de likelihood van een getraind model. Men zou verwachten dat modellen die dichterbij het globale optimum komen ook een betere prestatie zouden hebben.

De andere aanname die in Hoofdstuk 2 is getoetst, is dat modellen die al dicht bij het globale optimum liggen (Baum-Welch start met een willekeurig gekozen startpunt en optimaliseert vanaf daar), ook dicht bij het globale optimum zullen komen te liggen. We vonden echter dat het moeilijk is voor modellen om zonder hulp te convergeren naar een punt dichtbij het globale optimum.

Hoofdstuk 3

> Dit hoofdstuk is gebaseerd op
> van Oosten, J.-P. and Schomaker, L. (2014a). A reevaluation and benchmark of hidden Markov models. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 531–536. IEEE

Hoofdstuk 3 gaat verder met onderzoek naar HMMs, maar met een focus op de essentiële elementen van de modellen. We waren vooral geïnteresseerd in de relatie tussen de *state transition*-kansen die de temporale aspecten modelleren, en de *observation*-kansen die de vormkenmerken per state modelleren. De belangrijkste

aanname die in dit hoofdstuk aan bod komt is dat de temporale structuur ongeveer even belangrijk is als de feature-representatie.

We hebben de relatie tussen de twee delen van de modellen bestudeerd door wederom data te genereren, dit keer met een bepaalde temporele structuur, en modellen te trainen op deze data. Deze structuur zou in de state transition-kansen van het getrainde model aanwezig moeten zijn. Echter, de experimenten in Hoofdstuk 3 laten zien dat de structuur niet overduidelijk teruggevonden kan worden. Een ander experiment in Hoofdstuk 3 verwijdert de temporele relatie tussen states om te kijken of de prestaties voor een classificate-taak achteruit gingen. Opmerkelijk genoeg gingen de prestaties niet zo drastisch omlaag als verwacht.

De belangrijkste conclusie die we uit deze experimenten kunnen trekken is dat de observatie-kansen een grotere impact op de modelprestaties hebben dan de transitie-kansen. Dit betekent, in relatie tot onze globale onderzoeksvraag, dat feature representatie in het bijzonder aandacht verdient.

Hoofdstuk 4

> Dit hoofdstuk is gebaseerd op
> van Oosten, J.-P. and Schomaker, L. (2014b). Separability versus prototypicality in handwritten word-image retrieval. *Pattern Recognition*, 47(3):1031–1038

In Hoofdstuk 4 tenslotte richten we onze aandacht op het labelingsaspect van het handschriftherkenningsproces. We beschouwen het als een essentieel onderdeel van het proces en integreren het expliciet in een iteratief proces. De hitlist-interface wordt in dit hoofdstuk geïntroduceerd. Deze helpt door een sneeuwbaleffect bij het verzamelen van een grote hoeveelheid trainingsdata (dat wil zeggen, een aanvankelijk klein aantal labels kan op den duur steeds meer labels verzamelen). Een hitlijst wordt samengesteld door woorden in verschillende lijsten te classificeren en vervolgens elke lijst te rangschikken (*ranking*). We kwamen er-

achter dat je niet kunt aannemen dat een goede classifier ook een goede rangschikking kan maken.

Met betrekking tot labeling ontdekten we dat het belangrijk is om de constructie van een dataset als onderdeel van het proces te beschouwen en menselijke annotatoren te integreren in een continue leercyclus. Een implicatie van Hoofdstuk 4 is dat men moet afwisselen tussen classificatie en rangschikking, en verschillende methoden moet gebruiken die zijn geoptimaliseerd voor elke subtaak. Beschouw bovendien de specifieke classificatie- en rangschikkingsmethoden niet als vaststaand. Het is nodig om deze af te wisselen, bijvoorbeeld als de huidige methode niet genoeg nieuwe labels oplevert om het momentum vast te houden. De uiteindelijke conclusie hieruit is dat het handschriftherkenningsproces geen statisch proces is, of een enkel trainingsmoment, maar constant onderhoud nodig heeft.

Discussie

Tot slot bespreekt dit proefschrift de menselijke betrokkenheid bij het handschriftherkenningsproces vanuit drie verschillende invalshoeken: In het ontwerp van de machine learning-methoden, het ontwerp van feature extraction-methoden en representaties, en het labelen. De Hoofdstukken 2 en 3 gaan voornamelijk over aannames rond machine learning- en feature extraction-methoden, terwijl het belangrijkste onderwerp in Hoofdstuk 4 het omgaan met een veranderende dataset is, vooral als er continu labels worden toegevoegd.

De belangrijkste methode om de aannames bij het gebruik van HMMs te onderzoeken is het genereren van data uit bekende modellen, en om te bestuderen wat er tijdens de training in de modellen gebeurt. Vanuit een globaal perspectief kijken wat er gebeurt in lokale (*gradient descent*) processen is een methode die kan worden gebruikt om ook andere machine learning-methoden te bestuderen, zoals neurale netwerken. Als de modellen zelf kunnen worden gebruikt om data te genereren, is het relatief eenvoudig om het getrainde model te vergelijken met het globale optimum.

Het grotere thema in dit proefschrift gaat over het idee dat we het handschriftherkenningsproces als een dynamisch proces moeten beschouwen. In het Monk-systeem komt dit tot uiting in een flexibele hitlijst-interface. Hoewel de hitlijstmethode in dit proefschrift alleen wordt toegepast op handgeschreven woorden, zijn we van mening dat deze vorm van actief leren relevant is voor machine learning in het algemeen. Idealiter wordt het gunstige effect van een labelingshandeling zo snel mogelijk door de gebruiker ervaren. Dit creëert een sneeuwbaleffect in de feedbackloop en leidt tot een breed gelabelde dataset. Een ander voordeel van de hitlijst-interface is dat het exploratie mogelijk maakt. Het gebruik van verschillende classificatie- en rangschikkingsmethoden is nuttig wanneer het toevoegen van labels stagneert.

Het is van cruciaal belang voor alle machine learning-methoden om over de juiste gelabelde data te beschikken. Daarom is het framework dat beschreven is in dit proefschrift relevant voor alle toepassingen van machine learning, zowel in de academische wereld als ook in de industrie. Het advies van dit proefschrift is daarom om te investeren in een systeem om meer en betere labels te krijgen en om dit framework op te nemen in elke toepassing van machine learning.

PUBLICATIONS BY THE AUTHOR

Bhowmik, T. K., van Oosten, J.-P., and Schomaker, L. (2011). Segmental K-means learning with mixture distribution for HMM based handwriting recognition. *Pattern Recognition and Machine Intelligence*, pages 432–439

van Oosten, J.-P. and Schomaker, L. (2012). Separability versus prototypicality in handwritten word retrieval. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 8–13. IEEE

van Oosten, J.-P. and Schomaker, L. (2014b). Separability versus prototypicality in handwritten word-image retrieval. *Pattern Recognition*, 47(3):1031–1038

van Oosten, J.-P. and Schomaker, L. (2014a). A reevaluation and benchmark of hidden Markov models. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 531–536. IEEE

Surinta, O., Holtkamp, M., Karabaa, F., van Oosten, J.-P., Schomaker, L., and Wiering, M. (2014). A* path planning for line segmentation of handwritten documents. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 175–180. IEEE

Niitsuma, M., Schomaker, L., van Oosten, J.-P., Tomita, Y., and Bell, D. (2016). Musicologist-driven writer identification in early music manuscripts. *Multimedia Tools and Applications*, 75(11):6463–6479

van Oosten, J.-P. and Schomaker, L. (Submitted). Examining common assumptions about the convergence of the Baum-Welch training algorithm for hidden Markov models. *Journal of Machine Learning Research*

BIBLIOGRAPHY

(2003). The General Hidden Markov Model Library (GHMM). http://ghmm.org. Accessed February 22, 2014.

Ahmad, I., Fink, G. A., and Mahmoud, S. A. (2014). Improvements in sub-character HMM model based Arabic text recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 537–542. IEEE.

Almazán, J., Gordo, A., Fornés, A., and Valveny, E. (2014). Segmentation-free word spotting with exemplar SVMs. *Pattern Recognition*, 47(12):3967–3978.

Artières, T., Gallinari, P., Li, H., Marukatat, S., and Dorizzi, B. (2002). From character to sentences: A hybrid Neuro-Markovian system for on-line handwriting recognition. *Series in Machine Perception and Artificial Intelligence*, 47:145–170.

Artières, T., Marukatat, S., and Gallinari, P. (2007). Online handwritten shape recognition using segmental hidden Markov models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):205–217.

Azad, R. K. and Borodovsky, M. (2004). Probabilistic methods of identifying genes in prokaryotic genomes: connections to the HMM theory. *Briefings in bioinformatics*, 5(2):118–130.

Babu, G. and Feigelson, E. D. (2006). Astrostatistics: Goodness-of-fit and all that! In *Astronomical Data Analysis Software and Systems XV*, volume 351, page 127.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Barratt, S. (2017). Interpnet: Neural introspection for interpretable deep learning. *arXiv preprint arXiv:1710.09511*.

Baum, E. B. and Lang, K. (1992). Query learning can work poorly when a human oracle is used. In *International joint conference on neural networks*, volume 8, page 8.

Benouareth, A., Ennaji, A., and Sellami, M. (2008). Semi-continuous HMMs with explicit state duration for unconstrained Arabic word modeling and recognition. *Pattern Recognition Letters*, 29(12):1742–1752.

Bhowmik, T. K., van Oosten, J.-P., and Schomaker, L. (2011). Segmental K-means learning with mixture distribution for HMM based handwriting recognition. *Pattern Recognition and Machine Intelligence*, pages 432–439.

Bianne-Bernard, A.-L., Menasri, F., Mohamad, R. A.-H., Mokbel, C., Kermorvant, C., and Likforman-Sulem, L. (2011). Dynamic and contextual information in HMM modeling for handwritten word recognition. *IEEE transactions on pattern analysis and machine intelligence*, 33(10):2066–2080.

Bideault, G., Mioulet, L., Chatelain, C., and Paquet, T. (2015). Benchmarking discriminative approaches for word spotting in handwritten documents. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 201–205. IEEE.

Bluche, T., Ney, H., Louradour, J., and Kermorvant, C. (2015). Framewise and CTC Training of Neural Networks for Handwriting Recognition. In *2015 13th international conference on document analysis and recognition (ICDAR)*, pages 81–85. IEEE.

Borkar, V., Deshmukh, K., and Sarawagi, S. (2001). Automatic segmentation of text into structured records. In *ACM SIGMOD Record*, volume 30, pages 175–186. ACM.

Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

Britto, A., Sabourin, R., Bortolozzi, F., and Suen, C. Y. (2001). A two-stage HMM-based system for recognizing handwritten numeral strings. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 396–400. IEEE.

Bunke, H., Roth, M., and Schukat-Talamazzini, E. G. (1995). Off-line cursive handwriting recognition using hidden Markov models. *Pattern recognition*, 28(9):1399–1413.

Bunke, H. (2003). Recognition of cursive Roman handwriting: past, present and future. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 448–459. IEEE.

Chen, F. R., Wilcox, L. D., and Bloomberg, D. S. (1995). A comparison of discrete and continuous hidden Markov models for phrase spotting in text images. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 398–402. IEEE.

Clausner, C., Antonacopoulos, A., Mcgregor, N., and Wilson-Nunn, D. (2018). ICFHR 2018 Competition on Recognition of Historical Arabic Scientific Manuscripts–RASM2018. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 471–476. IEEE.

Collobert, R., Sinz, F., Weston, J., and Bottou, L. (2006). Large scale transductive SVMs. *Journal of Machine Learning Research*, 7(Aug):1687–1712.

Daelemans, W. and van den Bosch, A. (2005). *Memory-based language processing*. Cambridge Univ Pr.

Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):5:1–5:60.

Devlin, J., Kamali, M., Subramanian, K., Prasad, R., and Natarajan, P. (2012). Statistical machine translation as a language model for handwriting recognition. In *Frontiers in Handwrit-*

ing Recognition (ICFHR), 2012 International Conference on, pages 291–296. IEEE.

Doetsch, P., Zeyer, A., and Ney, H. (2016). Bidirectional decoder networks for attention-based end-to-end offline handwriting recognition. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 361–366. IEEE.

Dolfing, J. and Haeb-Umbach, R. (1997). Signal representations for hidden Markov model based online handwriting recognition. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 4, pages 3385–3388. IEEE.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern classification*.

Eddy, S. R. (1998). Profile hidden Markov models. *Bioinformatics*, 14(9):755–763.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Farago, A. and Lugosi, G. (1989). An algorithm to find the global optimum of left-to-right hidden Markov model parameters. *Problems Of Control And Information Theory-Problemy Upravleniya I Teorii Informatsii*, 18(6):435–444.

Figueiredo, M. A. T. and Jain, A. K. (2002). Unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):381–396.

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

Frinken, V., Fischer, A., Manmatha, R., and Bunke, H. (2012). A novel word spotting method based on recurrent neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 34(2):211–224.

Frinken, V., Kakisako, R., and Uchida, S. (2014). A novel HMM decoding algorithm permitting long-term dependencies and its application to handwritten word recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 128–133. IEEE.

Giacinto, G. (2007). A nearest-neighbor approach to relevance feedback in content based image retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 456–463. ACM.

Gil, S. and Williams, B. (2009). Beyond local optimality: An improved approach to hybrid model learning. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 3938–3945. IEEE.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868.

Hannun, A. (2017). Sequence Modeling with CTC. *Distill*. https://distill.pub/2017/ctc.

Hawkins, J. and Blakeslee, S. (2007). *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines*. Macmillan.

Hawkins, J., Lewis, M., Klukas, M., Purdy, S., and Ahmad, S. (2018). A framework for intelligence and cortical function based on grid cells in the neocortex. *bioRxiv*.

Hsu, D., Kakade, S. M., and Zhang, T. (2012). A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.

Jégou, H., Douze, M., and Schmid, C. (2010). Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336.

Khemiri, A., Kacem Echi, A., Belaid, A., and Elloumi, M. (2015). Arabic handwritten words off-line recognition based on HMMs and DBNs. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 51–55. IEEE.

Kohonen, T. (1987). Adaptive, associative, and self-organizing functions in neural computing. *Applied Optics*, 26(23):4910–4918.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kuo, S.-s. and Agazzi, O. E. (1993). Machine vision for keyword spotting using pseudo 2D hidden Markov models. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 5, pages 81–84. IEEE.

Lee, J.-S. and Park, C. H. (2006). Training hidden Markov models by hybrid simulated annealing for visual speech recognition. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 1, pages 198–202. IEEE.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Lorigo, L. M. and Govindaraju, V. (2006). Offline Arabic handwriting recognition: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 28(5):712–724.

Marti, U. and Bunke, H. (2000). Handwritten sentence recognition. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 3, pages 463–466. IEEE.

Mouchère, H. (2007). *Étude des mécanismes d'adaptation et de rejet pour l'optimisation de classifieurs: Application à la reconnaissance de l'écriture manuscrite en-ligne*. PhD thesis, l'Institut National des Sciences Appliquées de Rennes.

Myers, R. and Whitson, J. (1994). HMM: Hidden Markov Model software for automatic speech recognition. https://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/speech/systems/hmm/0.html. Accessed February 22, 2014.

Niitsuma, M., Schomaker, L., van Oosten, J.-P., Tomita, Y., and Bell, D. (2016). Musicologist-driven writer identification in early music manuscripts. *Multimedia Tools and Applications*, 75(11):6463–6479.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. (2018). The building blocks of interpretability. *Distill*. https://distill.pub/2018/building-blocks.

Pan, S. J., Yang, Q., et al. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Park, H.-S. and Lee, S.-W. (1998). A truly 2-D hidden Markov model for off-line handwritten character recognition. *Pattern Recognition*, 31(12):1849–1864.

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

Puigcerver, J., Toselli, A. H., and Vidal, E. (2015). Probabilistic interpretation and improvements to the HMM-filler for handwritten keyword spotting. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 731–735. IEEE.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rabiner, L. R., Wilpon, J. G., and Juang, B.-H. (1986). A segmental k-means training procedure for connected word recognition. *AT&T technical journal*, 65(3):21–31.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

Rigoll, G., Kosmala, A., Rattland, J., and Neukirchen, C. (1996). A comparison between continuous and discrete density hidden Markov models for cursive handwriting recognition. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 2, pages 205–209. IEEE.

Ritsema van Eck, M. P. and Schomaker, L. (2012). Formal semantic modeling for human and machine-based decoding of medieval manuscripts. In *Digital Humanities, Hamburg*.

Rothacker, L. and Fink, G. A. (2015). Segmentation-free query-by-string word spotting with Bag-of-Features HMMs. In *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pages 661–665. IEEE.

Roy, P. P., Dey, P., Roy, S., Pal, U., and Kimura, F. (2014). A novel approach of Bangla handwritten text recognition using HMM. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 661–666. IEEE.

Salton, G. and Buckley, C. (1997). Improving retrieval performance by relevance feedback. *Readings in information retrieval*, 24:5.

Sanchez, J. A., Romero, V., Toselli, A. H., and Vidal, E. (2016). ICFHR2016 competition on handwritten text recognition on the READ dataset. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, pages 630–635. IEEE.

Schenk, J., Schwärzler, S., Ruske, G., and Rigoll, G. (2008). Novel VQ designs for discrete HMM on-line handwritten whiteboard note recognition. In *Pattern Recognition*, pages 234–243. Springer.

Schomaker, L. (2007). Retrieval of handwritten lines in historical documents. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 594–598. IEEE.

Schomaker, L. (2016). Design considerations for a large-scale image-based text search engine in historical manuscript collections. *it-Information Technology*, 58(2):80–88.

Schomaker, L., de Leau, E., and Vuurpijl, L. (1999). Using pen-based outlines for object-based annotation and image-based queries. In *Proceedings of the Third International Conference on Visual Information and Information Systems*, VISUAL '99, pages 585–592, London, UK, UK. Springer-Verlag.

Schomaker, L., Franke, K., and Bulacu, M. (2007). Using codebooks of fragmented connected-component contours in forensic and historic writer identification. *Pattern Recognition Letters*, 28(6):719–727.

Schuster-Böckler, B., Schultz, J., and Rahmann, S. (2004). HMM Logos for visualization of protein families. *BMC bioinformatics*, 5(1):1.

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Settles, B. and Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Siddiqi, S. M., Gordon, G. J., and Moore, A. W. (2007). Fast state discovery for HMM model selection and learning. In *International Conference on Artificial Intelligence and Statistics*, pages 492–499.

Strauß, T., Leifert, G., Labahn, R., Hodel, T., and Mühlberger, G. (2018). ICFHR2018 Competition on Automated Text Recognition on a Read Dataset. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 477–482.

IEEE.

Surinta, O., Holtkamp, M., Karabaa, F., van Oosten, J.-P., Schomaker, L., and Wiering, M. (2014). A* path planning for line segmentation of handwritten documents. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 175–180. IEEE.

Surinta, O., Schomaker, L., and Wiering, M. (2012). Handwritten character classification using the hotspot feature extraction technique. In *Proceedings of the First International Conference on Pattern Recognition Applications and Methods, 2012*, pages 261–264.

Takahashi, F. and Abe, S. (2002). Decision-tree-based multiclass support vector machines. In *Proceedings of the 9th International Conference on Neural Information Processing*, volume 3, pages 1418–1422. IEEE.

Tax, D. (2001). *One-class classification*. PhD thesis, Technische Universiteit Delft.

van der Zant, T., Schomaker, L., and Haak, K. (2008a). Handwritten-word spotting using biologically inspired features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):1945–1957.

van der Zant, T., Schomaker, L., and Valentijn, E. (2008b). Large scale parallel document image processing. In *Electronic Imaging 2008*, pages 68150S–68150S. International Society for Optics and Photonics.

van der Zant, T., Schomaker, L., Zinger, S., and van Schie, H. (2009). Where are the search engines for handwritten documents? *Interdisciplinary Science Reviews, 34*, 2(3):224–235.

van Oosten, J.-P. and Schomaker, L. (2012). Separability versus prototypicality in handwritten word retrieval. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 8–13. IEEE.

van Oosten, J.-P. and Schomaker, L. (2014a). A reevaluation and benchmark of hidden Markov models. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 531–536. IEEE.

van Oosten, J.-P. and Schomaker, L. (2014b). Separability versus prototypicality in handwritten word-image retrieval. *Pattern Recognition*, 47(3):1031–1038.

van Oosten, J.-P. and Schomaker, L. (Submitted). Examining common assumptions about the convergence of the Baum-Welch training algorithm for hidden Markov models. *Journal of Machine Learning Research*.

Vapnik, V. (1982). *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag, New York.

Wei, H., Baechler, M., Slimane, F., and Ingold, R. (2013). Evaluation of SVM, MLP and GMM classifiers for layout analysis of historical documents. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1220–1224. IEEE.

Young, S. J., Evermann, G., Gales, M. J. F., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. C. (2006). *The HTK Book, version 3.4*. Cambridge University Engineering Department, Cambridge, UK.

Zhang, Z., Dai, B. T., and Tung, A. K. (2008). Estimating local optimums in EM algorithm over Gaussian mixture model. In *Proceedings of the 25th international conference on Machine learning*, pages 1240–1247. ACM.

Zimmermann, M. and Bunke, H. (2002). Hidden Markov model length optimization for handwriting recognition systems. In *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 369–374. IEEE.

Zimmermann, M. and Bunke, H. (2004). N-gram language models for offline handwritten text recognition. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 203–208. IEEE.

Zinger, S., Nerbonne, J., and Schomaker, L. (2009). Text-image alignment for historical handwritten documents. In *IS&T/SPIE Electronic Imaging*, pages 724703–724703. International Society for Optics and Photonics.

ACKNOWLEDGEMENTS

When I started my PhD research, almost 10 years ago, I didn't really know what to expect. I know I didn't expect it to take so long, or that I'd finish a good portion of the work while working a full-time job (for anyone that's reading this, and considering to do a part-time PhD while working: please reconsider!). However, there were quite a few marvellous people along the way that helped me on this interesting journey.

Of course, I want to thank my supervisor and promoter Lambert Schomaker. I'm very grateful for the opportunity to continue the research that we started with the Master's project. The many hours that we've spent always seemed filled with interesting thoughts and even philosophical debates. It seemed that the more we got into the subject, the longer and deeper our discussions became. I'd also like to extend my thanks to the reading committee, Antal van den Bosch, Michael Biehl and Rolf Ingold.

I'm also deeply thankful for the opportunity to travel to Japan and visit the Kyushu university. For that, I cannot thank Seiichi Uchida and Volkmar Frinken enough, for the invitation and the interesting scientific discussions, the amazing trips we took and the friendship. This is an experience that will remain a fond memory for a long time.

There are also the many colleagues at the ALICE institute, some of which I shared an office with. It is probably impossible to name everyone, but in particular, I'd like to thank Amir, Aswin, Bea, Ben, Charlotte, Faik, Harmen, Hedde, Jolie, Jort, Mahya, Michiel, Olarik, Robert, Sheng, Tapan, Tijn, and Trudy. A special thanks goes out to Elina for all her excellent help all these years!

Then there are the many colleagues at Target Holding / Slimmer AI. I will not name everyone, but I'd be remiss if I didn't thank

Gert-Jan, for all the support and occasionally the kick in the butt, and JC for taking over the supporting role and kicks-in-the-butt these last two years.

Over the course of 10 years, there were a lot of friends that took an interest in my progress and are hopefully also excited for me to finally finish. Some people I'd like to thank in random order: Frank (for your programming wisdom, the great collaboration on Flexc++ and generally being a good friend), Robert and Marieke (all the support, the board games, being there all these years!), Richard and Marloes (also lots of board games, the support and the many hours on the phone while we're physically distant), Bram and Carine (all the discussions deep into the night, all the support and love), Michiel and Kyuhee (for the hours of coding together), Ben and Charlotte (also lots of board games). My friends from high school and their loved ones: Maurits and Swi Yin, Maarten and Celine (hopefully we'll be able to get back to escaping rooms soon!). Jilles kept not only my mind sane (with amazing movie nights!) but also my body (I really hope I can get back to kicking your ass!). At Target / Slimmer, I also met some amazing friends: Harma (for all the support and long talks about both our work and life), Marc, Mark and Tom (Passie = Bassie!) and Rolf (for your wisdom, grey hairs, amazing Linux bash-skills and computer history knowledge).

Thanks to Bente for the amazing drawing on the cover of the thesis.

Of course, I want to thank my amazing parents, Paul and Harda. They have always been there when things were a bit rough. Thanks for always believing in me! Thanks also to Reinier and Daphne (and Mees!), Oma, and everyone else in the family for the support and love!

Last but certainly not least: Mitsai. Without her support, this thesis would not have been finished. I would not only have lost my mind, but probably also gone hungry. The unconditional support and love is amazing, and I am forever in your debt for sticking it out with me through all these highs and lows. Thanks love, we made it!