Offline Handritting Word Recognition

Thijs Kooi, Davide Modolo

January 27, 2011

Table of contents

1. Overview
   - General
   - Dataset
2. Implementation Details
   - Pipeline
   - Pre-Processing
   - Feature Extraction
   - Hidden-Markov Model
3. Results
4. Conclusions
5. Section no. 4
   - blocs

Overview of the Project

Off-line handwriting recognition

- It involves the automatic conversion of text in an image into letter codes which are usable within computer and text-processing applications
- Off-line handwriting recognition is comparatively difficult, as different people have different handwriting styles

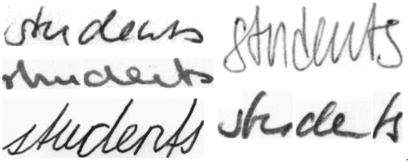


Figure: 'Students' written by different authors

Our AI Project

A lot of research has been done over the past years.

We explored the topic and implemented a full pipeline for the task. The research touched different fields:

- Data Collection
- Image Processing
- Features extraction
- Machine Learning
- Word Recognition

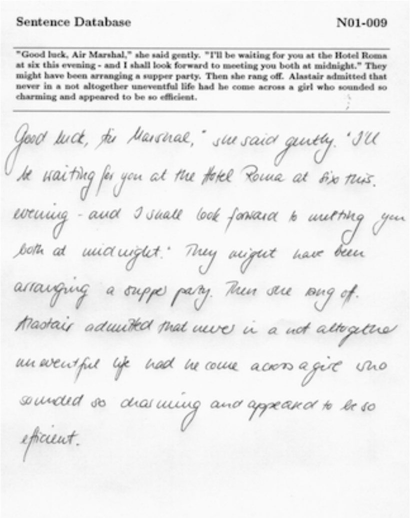
Dataset

The IAM Handwriting Database 3.0[1]

- Unconstrained handwritten text (scanned at a resolution of 300dpi and saved as PNG images with 256 gray levels)
- 1'539 pages of scanned text of 657 writers
- 13'353 isolated and labeled text lines
- 115'320 isolated and labeled words

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

Example of a page of scanned text



Sentence Database N01-009

"Good luck, Air Marshal," she said gently. "I'll be waiting for you at the Hotel Roma at six this evening - and I shall look forward to meeting you both at midnight." They might have been arranging a supper party. Then she rang off. Alastair admitted that never in a not altogether uneventful life had he come across a girl who sounded so charming and appeared to be so efficient.

"Good luck, Air Marshal," she said gently. "I'll
be waiting for you at the Hotel Roma at six this
evening - and I shall look forward to meeting you
both at midnight." They might have been
arranging a supper party. Then she rang off.
Alastair admitted that never in a not altogether
uneventful life had he come across a girl who
sounded so charming and appeared to be so
efficient.

Implementation Details

Pipeline

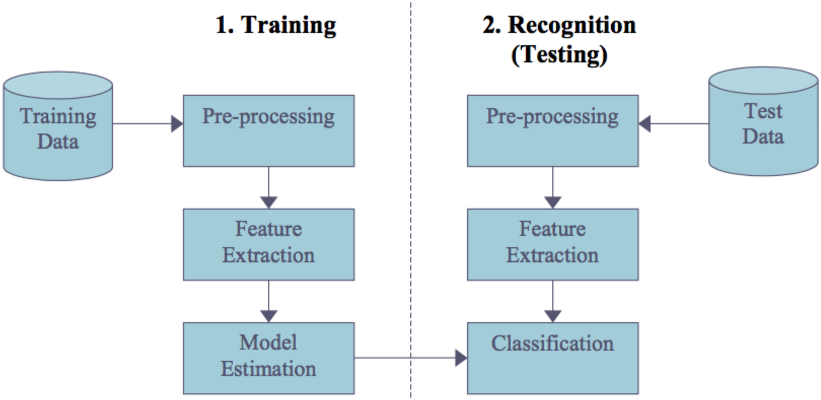


Figure: Pipeline of a word recognition system

Implementation Details

Pre-Processing

Pre-processing

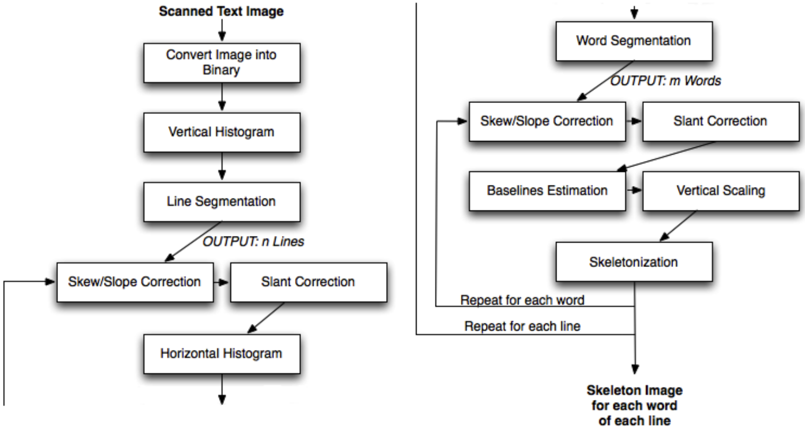
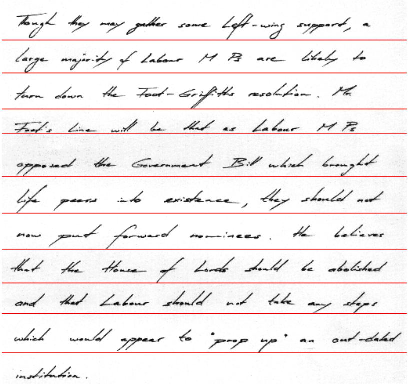
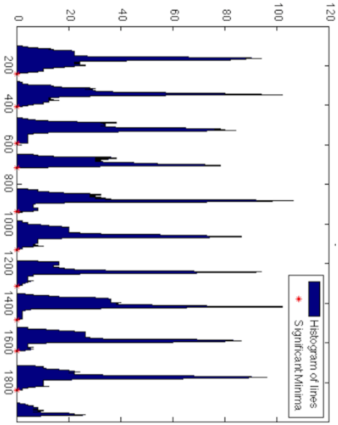


Figure: Pipeline for the pre-processing/normalization step

Line Segmentation



Original image segmented in lines



Vertical histograms and significant minima

Figure: Example of line segmentation

Skew and Slope Correction

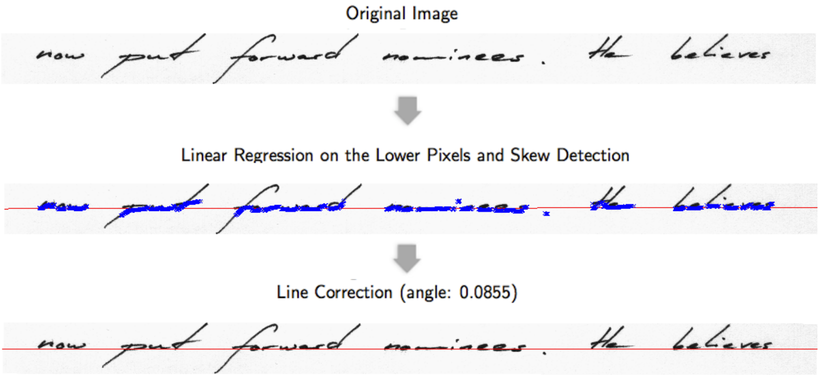


Figure: Skew detection and correction pipeline

Slant Correction

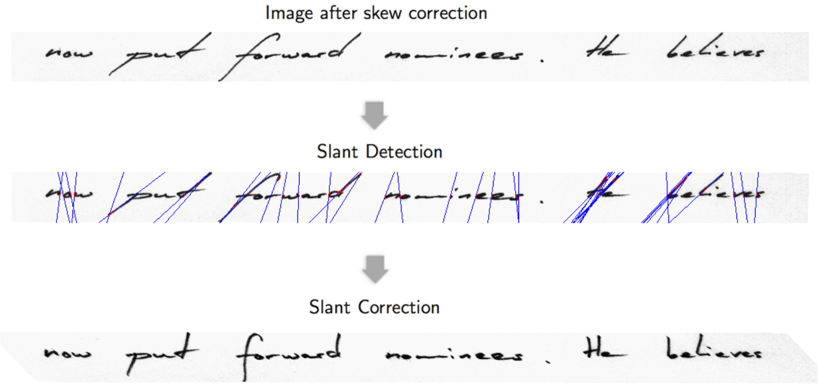
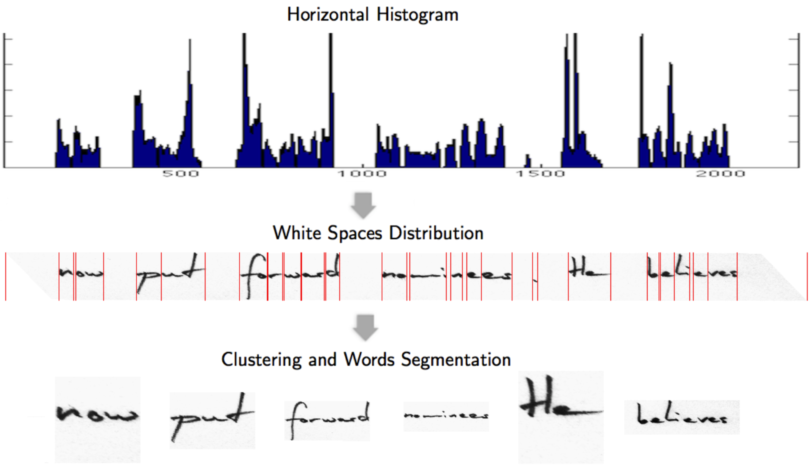
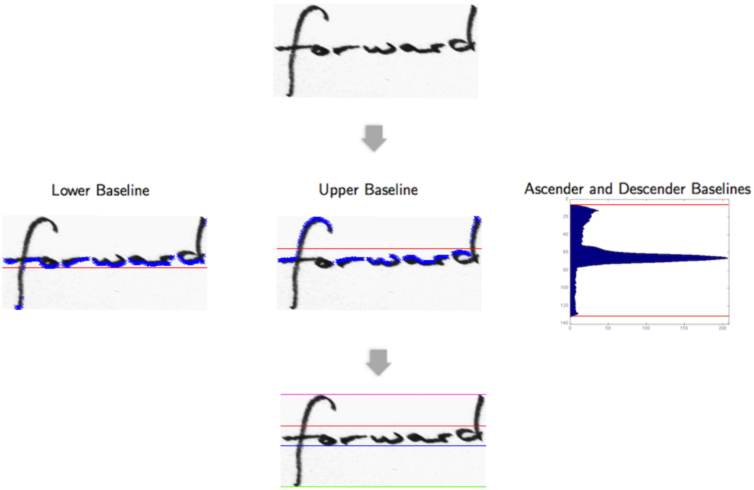


Figure: Slant detection and correction pipeline

Word segmentation



Baseline Estimation



Vertical Scaling

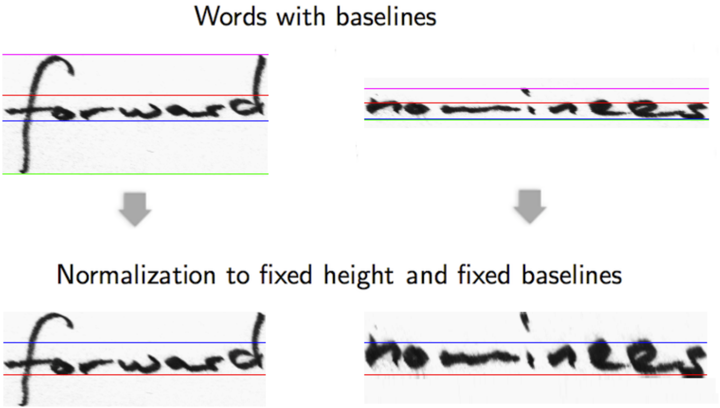


Figure: Examples of vertical scaling process

Skeletonization

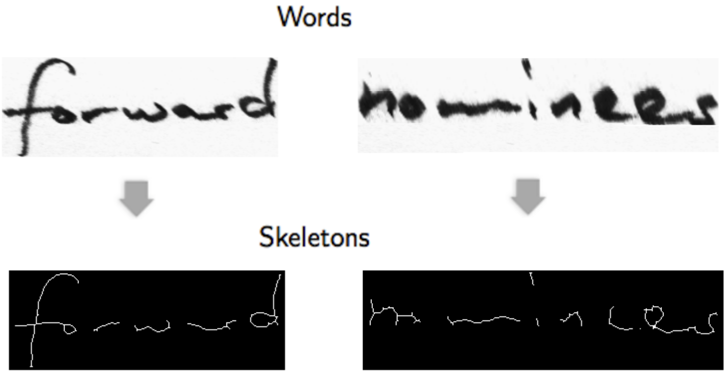
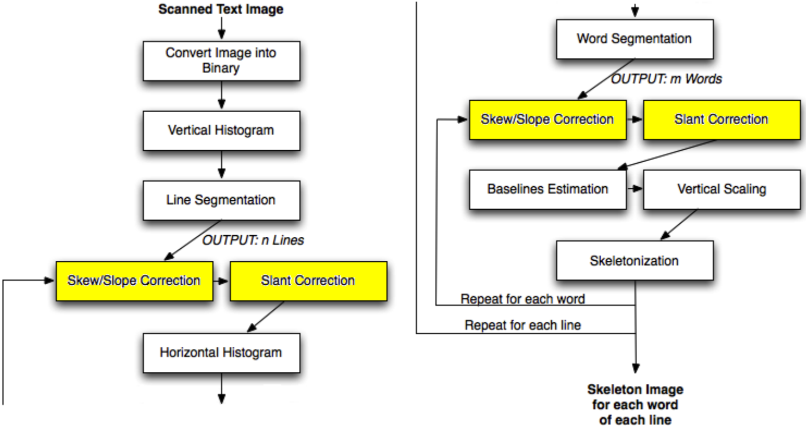


Figure: Skeletonization process

Remembering the entire pipeline....

Why repeating skew and slant correction twice?



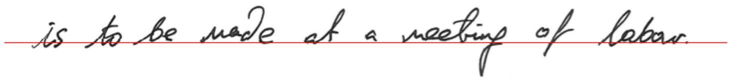
The normalizations are necessary for:

First

Words segmentation

Second

Not all the words have the same slope, slant and lower baseline



Implementation Details

Feature Extraction

Features

Extracted from the skeleton of the words.

Mainly 2 types:

- Statistical
- Morphological

Statistical Features

Percentage of white pixels in the 3 zones of the word:

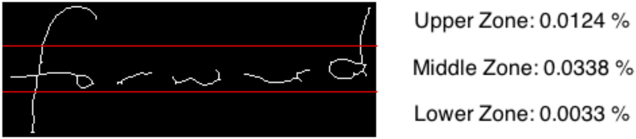


Figure: Example

Morphological Features

Obtained by **connected component analysis**.

- A connected component it is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices.

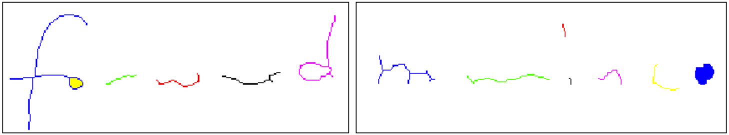
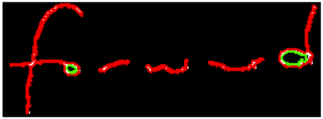


Figure: Example of words divided in different components (colors)

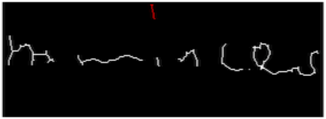
Morphological Features

We extract:

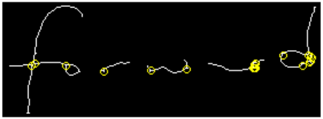
Loops



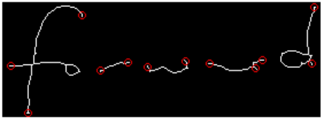
Dots



Junctions



Endpoints



HMM

- A set of N states $S = (s_1, s_2, \ldots, s_N)$, where the state of the system at time t is denoted q_t
- A set of priors $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$, providing the probability $P(q_1 = s_i)$.
- A transition function $\mathbf{A}$, where $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$.
- An observation function $\mathbf{B}$, mapping each observation at every state to a probability $b_i(\mathbf{o}_t) = P(\mathbf{o}_t | q_t = s_i, \lambda)$, where λ denotes the model parameters.

The model is trained to estimate the posterior probability $P(\mathbf{O}|\lambda)$ of an observation sequence $\mathbf{O}$, with D-dimensional observation vectors $\mathbf{o}_t = (o_1, o_2, \ldots, o_D)$.

HMM

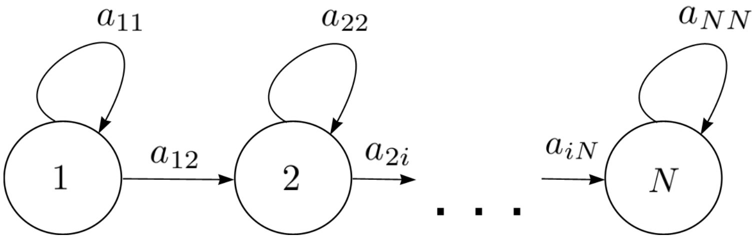


Figure: Left-to-right HMM with N states

Main problems in an HMM

1. The probability of an observation sequence, given the model, $P(\mathbf{O}|\lambda)$.
2. The most likely parameters of the model $\lambda^* = \max P(X|\lambda)$, given a training set of M observation sequences $X = (\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_M)$.
3. The most likely state sequence, underlying a given observation sequence and the model, $Q^* = \max P(Q|\mathbf{O}, \lambda)$.

Main problems in an HMM

1. The probability of an observation sequence, given the model, $P(\mathbf{O}|\lambda)$.

   Sum-product algorithm: forward-backward algorithm

2. The most likely parameters of the model $\lambda^* = \max P(X|\lambda)$, given a training set of M observation sequences $X = (\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_M)$.

   EM-algorithm: Baum-Welch reestimation

3. The most likely state sequence, underlying a given observation sequence and the model, $Q^* = \max P(Q|\mathbf{O}, \lambda)$.

   Dynamic programming: Viterbi algorithm

Forward probability

$$\alpha_t(i) \equiv P(o_1, o_2, \ldots, o_t | q_t = s_i, \lambda) = \left[\sum_{j=1}^{N} \alpha_{t-1}(j) a_{ij}\right] b_j(o_t) \quad (1)$$

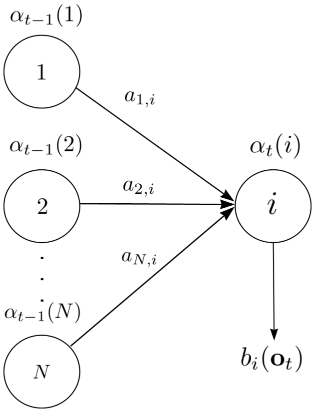


Figure: Computation of forward probability

Forward probability

$$\alpha_t(i) \equiv$$
$$P(o_1, o_2, \ldots, o_t | q_t = s_i, \lambda) =$$
$$\left[\sum_{j=1}^{N} \alpha_{t-1}(j) a_{ij} \right] b_j(o_t) \tag{2}$$

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{3}$$

Problem 1 solved.

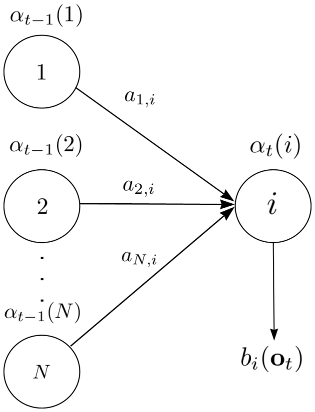


Figure: Computation of forward probability

Learning the parameters

The forward-backward algorithm also commes with a backward probability:

$$\beta_t(j) \equiv P(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = s_j, \lambda) = \sum_{i=1}^{N} a_{ij} b_i(o_{t+1}) \beta_i(o_{t+1}) \tag{4}$$

With this we can define the probability of being in a state at a timestep as:

$$\gamma_t(i) \equiv (P(\mathbf{O}|\lambda))^{-1} \alpha_t(i) \beta_t(i) \tag{5}$$

Where normalisation constant
$P(\mathbf{O}|\lambda) = \sum_{j=1}^{N} \alpha_t(i)\beta_t(i) = \sum_{j=1}^{N} \alpha_T(j)$.

Learning the parameters

$\hat{a}_{ij} = frac$Prob. of being in i and transfering to jProb. of begin in i $=$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{6}$$

The priors remain fixed, for the left-to-right model.

Updating the parameters

Comparison with GMM:

Model:	**GMM**	**HMM**
Model parameters:	$\lambda = \pi, \mu, \Sigma$	$\lambda = \pi, \mathbf{A}, \mathbf{B}$
Hyper parameters:	Number of components	Topology (states, transitions), observation function
Observed variables:	Data points	Observations
Latent variables:	Priors of a component	State sequence

Updating parameters

Model:	**GMM**	**HMM**
E-step:	Estimate the probability of a component, given the data and current parameters.	Estimate the probability of being in a state at a timestep and the probability of transfering from a state to another state.
M-step:	Maximise π, μ and Σ.	Maximise π, **A** and **B**

Problems we ran into

Consider the following observation:

$$\mathbf{O} = \begin{pmatrix} -1 & 0 \\ 2 & 0 \\ 1 & 0 \\ -2 & 0 \end{pmatrix}$$

Problems we ran into

Singularities. Consider the following observation:

$$\mathbf{O} = \begin{pmatrix} -1 & 0 \\ 2 & 0 \\ 1 & 0 \\ -2 & 0 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 3\frac{1}{3} & 0 \\ 0 & 0 \end{pmatrix}$$

Problems we ran into

Singularities

$$\mathbf{O} = \begin{pmatrix} -1 & 0 \\ 2 & 0 \\ 1 & 0 \\ -2 & 0 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 3\frac{1}{3} & 0 \\ 0 & 0 \end{pmatrix}$$

$$|\Sigma| = 0$$

Possible solution: Add some random noise.

Problems we ran into

Singularities

$$\mathbf{O} = \begin{pmatrix} -1 & 0 \\ 2 & 0 \\ 1 & 0 \\ -2 & 0 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 3\frac{1}{3} & 0 \\ 0 & 0 \end{pmatrix}$$

-¿ Add some random noise, also to prevent variance from collapsing.

Problems we ran into

Short words have less likelihood.
Harder to recognise, more subject to writer variations.
tried to solve this by using MOG.

Results

blabla

Conclusions

blabla

blocs

title of the bloc

bloc text

title of the bloc

bloc text

title of the bloc

bloc text