Master thesis Data Science
Computing Science

Radboud University

Extracting Entities from Handwritten Civil Records using Handwritten Text Recognition and Regular Expressions

Historical Database of Suriname and the Caribbean
slavenregisters@let.ru.nl

Author:
Lisa Hoek

First supervisor/assessor:
prof. dr. Martha A. Larson

Second assessor:
dr. Iris H.E. Hendrickx

In coorporation with:
Radboud Group for Historical
Demography and Family History.

Daily supervisor:
Björn Quanjer

Nijmegen, August 2023

Abstract

The Historical Database of Suriname and the Caribbean (HDSC) is a foundation that collects and manages digital data collections of historical documents from former Dutch colonies in South America. After publishing the slave registers of Suriname and Curaçao dating from 1830 to 1863, their goal is now to expand this with more documents, e.g., civil records, and make these accessible to everyone. This to promote scientific research in historical demographics, but also to remove barriers for people to look up their ancestors and learn more about their family history.

Crowdsourcing in the digital humanities is an effective way to transcribe images of handwritten historical documents, and such projects are currently active in the HDSC. However, when data is large or the number of volunteers small, the transcription process can be too slow. A way to improve efficiency is to make use of computer models. Handwritten Text Recognition (HTR) obtains machine transcribed texts from image data, and Entity Extraction can transform these un-/semi-structured transcriptions into structured data.

In this thesis, we build an HTR+RegEx model that extracts entities from semi-structured handwritten death certificates from Curaçao. We use Transkribus for segmentation of the image data and HTR (CER=5.01), and Regular Expressions (RegExes) to collect the entities of the transcription texts.

We show that RegExes built on a small set of data translate to majority of the certificates. Unfortunately, the HTR-component in Transkribus has difficulty transcribing names (CER=16.06/WER=39.48). This propagates errors to our RegEx-component and makes linking persons in a final database of various historical population registers more difficult. So, we argue improvements should be made for HTR, but do see possibilities for this model to be integrated into the current crowdsourcing process of the HDSC to increase efficiency.

Acknowledgements

I would like to thank all colleagues of the Historical Database of Suriname and the Caribbean. I enjoyed spending time in the office with Björn and Matthias. Thank you, Coen and Thunnis, for my opportunities here at HDSC. I would also like to express gratitude to Erik from eScience Center who I could level with and discuss all Computer Science related topics to. Finally, I would like to thank my supervisor, Martha, who has given me extensive feedback and directions when needed. She gave me the support needed to finish my thesis.

Additional achievements

This thesis was presented at the ESHD Conference (2023, August 30-September 2) at Radboud University Nijmegen:

- Lisa Hoek, Thunnis van Oort, Björn Quanjer, Matthias Rosenbaum-Feldbrügge, Coen van Galen, and Jan Kok. (2023) *Defeating the Haystack. Combining HTR and entity recognition to reconstruct populations of the past: the case of Curaçao.* 5th Conference Of The European Society of Historical Demography, Nijmegen, the Netherlands.

Contents

Chapter 1

Introduction

This chapter introduces the Historical Database of Suriname and the Caribbean and how our thesis is part of it. We will provide a brief overview of our data, and the AI powered tool we will be using to digitise the civil records with Handwritten Text Recognition. We end the chapter with our research questions and outline of this thesis.

1.1 Historical Database of Suriname and the Caribbean

The Historical Database of Suriname and the Caribbean (HDSC) started in 2017 as a collaboration between the National Archives of Suriname and the Netherlands, the Radboud University, Nijmegen, and the Anton de Kom university, Paramaribo. They shared one goal: making the slave registers of Suriname 1830-1863 accessible for both researchers and the public. This enables historical demographic research about life in slavery. Moreover, people can look up their ancestors online and learn more about their family history.

After an earlier HDSC-project that transcribed and published the contracts of indentured servants in the late 1990s, digitisation of more documents hit a wall: they did not have the resources (funding and manpower) to facilitate this. The slave registers of Suriname are much larger with its 43 books (30.000 pages). To make these slave registers accessible, HDSC started a crowdfunding. With its wide attention and interest from the public, they collected enough money for the resources needed, and a community formed around the project. This automatically led to many active volunteers that contributed in the crowdsourcing that followed. In an astonishing six months, all slave registers were transcribed with the help of 600 volunteers. This returned a searchable database accessible for both researchers and the public[1].

Once HDSC published the slave registers of Suriname in 2018-2019, the National Archive of Curaçao contacted the HDSC that they also possessed many slave registers of their island. In 2020, a similar project enabled them to transcribe these documents too and publish these online[2]. The ambition of the HDSC is to expand this with the free populations of Suriname and Curaçao, but also to add all other former Dutch colonial possessions in South America. Their largest source of information are over 300,000 handwritten civil records: birth, death,

[1]Nationaal Archief Suriname. Slavenregisters. `https://nationaalarchief.sr/onderzoeken/alle-genealogie/genealogie-slavenregister/persons`

[2]Nationaal Archief Curaçao. Slavenregister. `https://www.nationaalarchief.cw/api/picturae/slavenregister/persons`

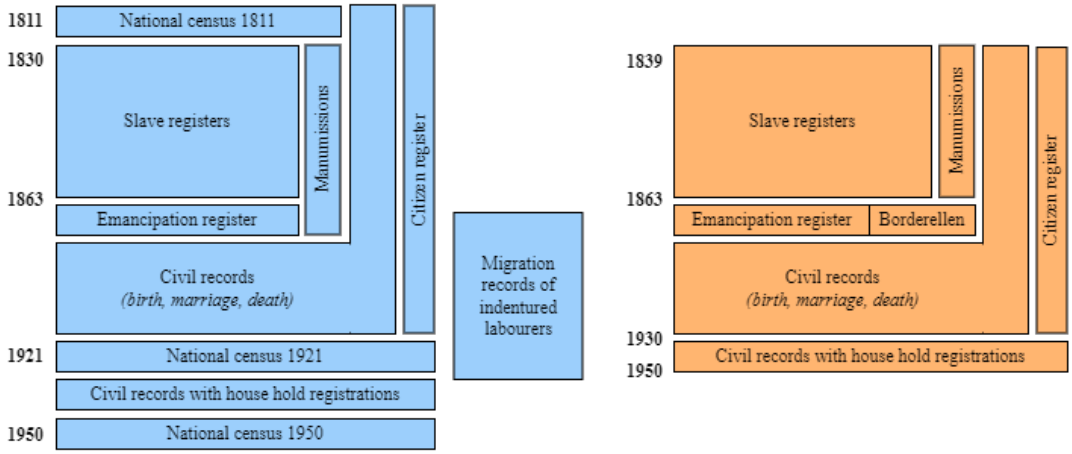


Figure 1.1: Reconstruction of registers and records of Suriname (left) and Curaçao (right).

and marriage certificates. Figure 1.1 shows an example of how the different types of population records of Suriname and Curaçao are connected in time. The Caribbean is a highly interconnected web of islands and people. Various historical population registers can be combined into one linked research database that will allow for the reconstruction of life courses in the Dutch colonies.

1.1.1 Scientific and public importance

The civil records contain data of when people were born, who their parents were, who they married, where people lived and when they died. With this civil data, we can investigate historical demographics of Suriname and other former colonies. For example, life after slavery, where did people go after being enslaved? Linking slave registers to manumissions to civil records can follow whole life courses. We can learn more about life on plantations by connecting pieces of data. Analysing transfers between slaves, mortality rates, etc. Research questions such as 'were enslaved children more likely to die when they were separated from their mother?'[3] or 'were mortality rates on certain plantations higher than on others?'[4] arise. It is of great importance that we learn more about the Dutch domination in its colonies, to be aware of the consequences that we still see today and understand how it shaped our current society, in both Suriname, the Caribbean islands and the Netherlands.

Besides the scientific opportunities to research this data, and the importance to learn more about Dutch colonial and slavery history, there are also multiple reasons for making the historical records freely accessible to 'the public': 1) it would only be possible with the help of volunteers, so they are eager to see their contribution, 2) the documents are for society because they are about society, and 3) for equality between people, they should have equal access; data (e.g., about one's own ancestors) should not be hidden behind a paywall. So, to create equality and giving back to society, it is important citizens can easily use the data too.

[3]So far, this has not been seen in the data [Thompson et al., 2023], arising other questions that there might have been a strong sense of community among the slaves taking care of the young ones regardless the absence of their mothers.

[4]This might tell us something about the amount of work slaves had to do on the plantation, and how well slave owners took care of their enslaved.

1.1.2 Privacy

In the Netherlands, archives are much more developed and many more historical documents are available online. One can trace back one's own ancestors in the online database WieWasWie[5] consisting of 90 million historical documents.Surinamese documents such as the slave registers or the national census of 1921 are also connected to this database.

Dutch Archival law (and comparable laws from former Dutch colonies) state that an archive can make civil records public after a certain time has passed. For birth records, this is 100 years, for marriage certificates after 75 years, and for death certificates after 50 years. The HDSC adheres to these standards and does not process or publish records of recent years. If desired, people can report infringing material at archives, but their reasoning or evidence must be well supported. The slave registers are even less privacy sensitive compared to civil records, because persons are very difficult to trace unless one knows what specifics to look for (enslaved persons did not have a last name).

Relating this to HDSC's objective to make the databases accessible to everyone, and its importance to do so, the HDSC feels great responsibility in making these records available. It is also well supported by the crowd: archives could not have accomplished their goals without HDSC's community of volunteers.

1.1.3 Crowdsourcing

One of the key components in citizen science projects is the continued motivation of the people. According to the HDSC, their projects are more than simply outsourcing their transcription tasks. The community is strong and the sense of belonging and contributing is what keep it going. Therefore, it is the volunteers that influence the chances of successful completion of the project. Over the last couple of years, Dutch slavery past has received increasing attention from the public. Unfortunately, as years progress, the daily average of transcribed certificates returned by volunteers is decreasing over time. Continuing the crowdsourcing as of now would take many years before all certificates are entered into the database [van Oort et al., 2022].

Currently, crowdsourcing in the HDSC is done via Het Volk[6]. This Dutch platform facilitates transcription projects. One of its current projects are death certificates from Suriname that are being transcribed. Two randomly selected volunteers view a certificate and enter the key information in a form with pre-chosen fields. Then, each certificate is checked by a third citizen scientist in the control project. The final decision is made by looking at the scan and comparing the two outputs from the volunteers. Having multiple citizen scientists look at the same certificate reduces mistakes, but is not very time efficient.

1.2 Motivation

Ways to increase productivity could make use of computer models. In particular, HTR (Handwritten text recognition) to obtain the plain text from the images, and Entity Extraction through RegExes (Regular Expressions) to label the desired information to put in a database. Maybe in some future, human labelling would not be needed anymore, and data extraction happens instantly. However, we are not close to reaching this point due to the difficulty of HTR and the variety of our data.

[5]WieWasWie. Iedereen heeft een geschiedenis. `https://wiewaswie.nl`

[6]Het Volk. Surinaamse Overlijdensakten III (districten 1846-1880). `https://hetvolk.org/projects/`

In this thesis, we assess whether computer models would perform sufficiently to be even useful at all, and if so, in what ways the computer models could be implemented to reduce the workload of the citizen scientists in the crowdsourcing process. Integrating the computer models in our crowdsourcing workflow is not straightforward; it can influence the performance and/or motivation of the citizen scientists. For example, we will see in this thesis that transcription quality reduces once annotators see pre-filled in text (they fail to correct the computer model). Another aspect are the incentives of the citizen scientists: if they only need to correct model output, will this lessen their feeling of contributing? We decided to mainly focus on building the HTR+RegEx model, but we will also briefly discuss our thesis' implications for the crowdsourcing process (how this would affect the citizen scientists) in future work.

1.2.1 Death certificates Curaçao

Now that the Surinamese civil records are being transcribed, HDSC also started examining the civil records of Curaçao. They acquired three Excel files from the National Archive Curaçao, along with scans of the birth, marriage, and death certificates. The tables consist of entities from the certificates that were transcribed by multiple people over the years. Quality is probably varying over the records; the archive is also unaware of the crowdsourcing process that took place. For the birth certificates, the Excel table is rather complete. The marriage certificates have some gaps in the table, but these images are also absent, so is rather complete too.

The death register database consists of some gaps. There exist 79.000 entries for the 70.000 death certificates of Curaçao dating from 1831 to 1950. However, this is incomplete data, consists of many duplicates, and the archive is unaware of its crowdsourcing process. Former work by the HDSC investigated that gaps in the data are mainly years 1879-1895, 1905-1909, 1930-1939 and 1945-1949. The HDSC would like to fill in these gaps, so these years are the focus point of this thesis.

Figure 1.2 shows two examples of death certificates (from 1891 and 1949). Figure A.1g shows another certificate format that was used before 1869 which consisted of three columns. This thesis will only focus on the two-column format, since our years of interest only have this type of layout.

What is interesting about this data is that the sentences are semi-structured and a mix between printed and handwritten words. The printed sentences change a little over the years, but make the certificate relatively well structured, e.g., the name of the deceased is always written in the same position on the certificate. This also holds for some other entities. Entities such as the names of the parents have more variety, these could have been written at the end of the certificate (in the empty space), or left out. Because this is more handwritten, it is less structured than the beginning of the certificate text (which is bound to the printed text). There do exist exceptions where printed text is crossed out and custom sentences are written down, e.g., for stillborns.

Because the data is semi-structured, we believe this makes regular expressions a suited method for Entity Extraction. Because historians from the HDSC are wondering this as well, this is one important research question for this thesis. After applying our HTR+RegEx model, we wish to obtain output in the form of a table containing the entity values of each certificate.

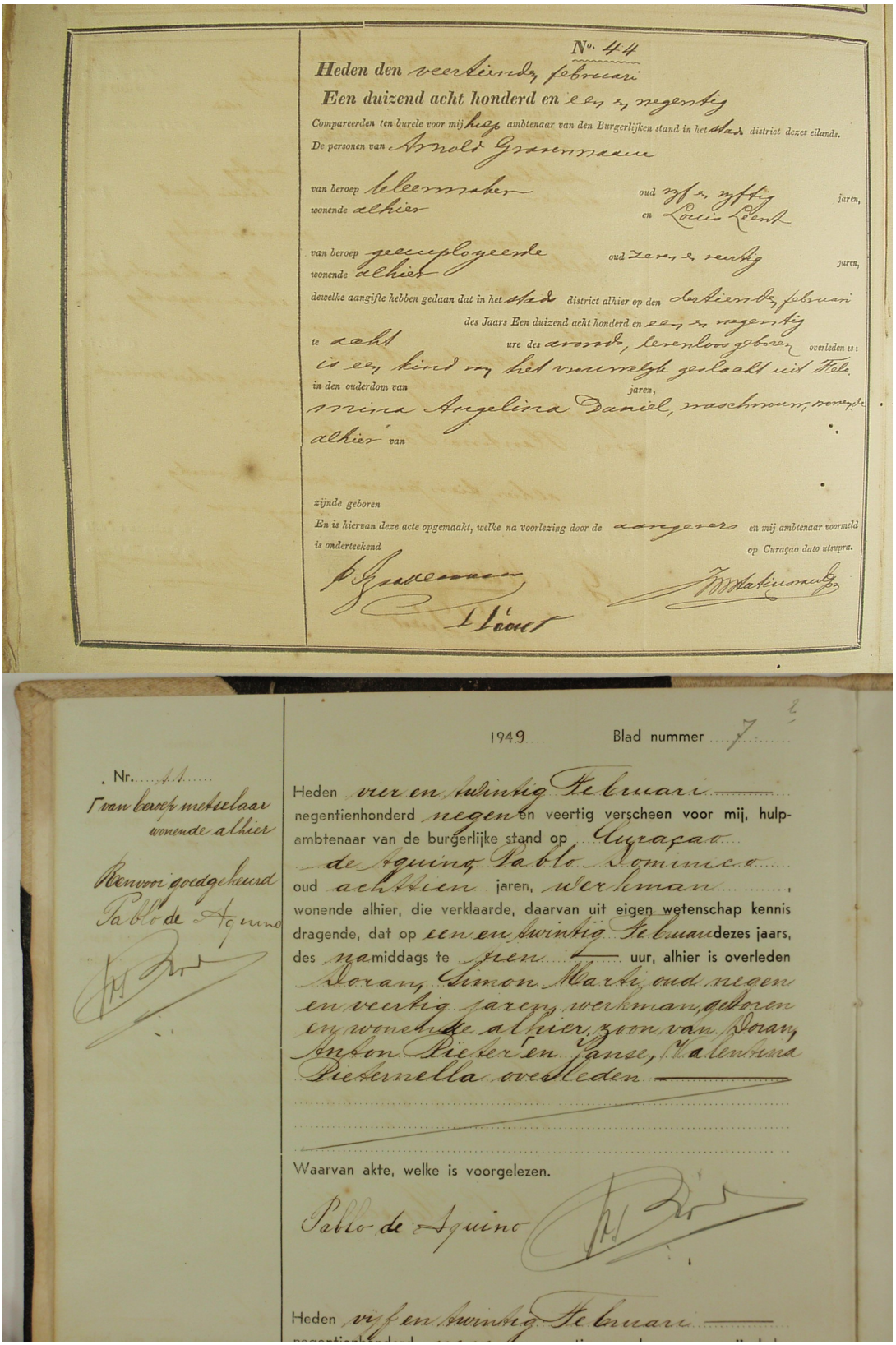


Figure 1.2: Two examples scans of death certificates from Curaçao 1891 and 1949.

1.2.2 Transkribus

Transkribus is a tool in which users, collaboratively or individually, transcribe collections of documents. These documents are scans (images) of printed or handwritten text, and their transcriptions can be made automatically using HTR models or manually by users. Transkribus is mainly used by researchers/professors and archivists [Terras, 2022]. We have not seen it being used outside the scope of historical texts (e.g., manuscripts, old letters / diaries, civil records). Transkribus is maintained and further expanded by READ-COOP[7], the cooperation successor from the Recognition and Enrichment of Archival Documents (READ) project[8].

Transkribus is suited for our research due to its many advantages. There are many functionalities all located in one platform; it is possible to collaborate in the same dataset, add labels to text, and HTR training can be done with a few clicks of a button. This comes in handy when users do not have a Machine Learning background, like the historians from the HDSC. Another benefit is that Transkribus is also suited for our annotation and no additional annotation tool, such as Doccano, is needed. Our annotators are already familiar with the platform.

Former experiments with Transkribus in a project of the HDSC sounded promising: already with a training set of 30 certificates, the tool is able to read the certificate with a 95% accuracy rate. Unfortunately but expected, the majority of the mistakes lay within the entities we wish to extract, the names of persons, city names, etc. We will investigate what kind of errors are made by the model and how we can improve on this.

There are also several disadvantages. Building HTR-models in Transkribus binds them to their platform; one can share models with one another within Transkribus, but these models cannot be used outside Transkribus. Another problem that occurred around November 2022 was that one of the two HTR-engines in Transkribus expired. This meant all models trained with this engine needed to be retrained with the other one. This shows durability problems and disadvantages of using Transkribus long-term. Hopefully, READ-COOP can avoid these problems in future. In addition, we encountered several bugs that we had to create some workarounds for.

Despite the disadvantages, Transkribus has a nice environment to obtain transcriptions of our death certificate data. Because we would like to focus more on the creation of RegExes, we think we can save time by using Transkribus than building our own custom HTR-model.

1.3 Research questions

During this thesis, we try to answer the following main research question:

RQ How can we increase efficiency of large-scale Entity Extraction from handwritten civil records while maintaining quality similar to crowdsourcing in the HDSC?

We limit ourselves to the death certificates of Curaçao for simplicity, and because this dataset is most wanted by the HDSC to be transcribed. It is important to retain low error

[7]READ-COOP. We revolutionise Access to Historical Documents. `https://readcoop.eu`

[8]European Commission - CORDIS EU research results. Recognition and Enrichment of Archival Documents. `https://cordis.europa.eu/project/id/674943`

rates, especially for names, to be able to link same persons but also their family members together in later projects of the HDSC. The need for a higher efficiency is high, as the current crowdsourcing project is still taking over three years to complete. To answer this research question, we build and test an HTR+RegEx model that can automatically extract the entities from image data. As future work, we describe possible ways how this model can be integrated into the crowdsourcing projects of the HDSC. We created multiple sub questions that help us answer the main research question:

RQ-1.1 What do the death certificates of Curaçao look like?
RQ-1.2 How do we construct useful subsets of data to use in research questions 2-4?

Chapter 4 starts by conducting a small exploration on the full set of Curaçao death certificates. We do this by manually skimming through many of the data. We describe variations in the data we encounter and decide what to include in our training and evaluation data. Because transcribing documents is time-expensive, we limit ourselves to transcribing 315 certificates in total during this thesis.

RQ-2.1 How can Transkribus be used to obtain the best HTR text possible within reasonable time limits?
RQ-2.2 How well does it perform?

Chapter 5 consists of the first component of our model. We decided to create our model in multiple steps, first obtaining the digital text through HTR (Chapter 5), then Entity Extraction through regular expressions (Chapter 7). The first component, obtaining the digital text, is done in Transkribus. In this chapter, we train multiple models in Transkribus and evaluate which is best. We also act upon some serious errors that significantly increase performances of our final pipeline.

RQ-3 What is the quality of the available death register database?

Chapter 6 analyses the death register database the HDSC is in possession of. They are unaware of its transcription process, so the exact quality is unknown. This chapter gives some insight into its quality, by comparing 100 samples from the database with our own ground truth data. This is also used to judge our annotators' transcription quality. We find alignments between our entity labels and the database entries, however, we also stumble upon many interpretation and paraphrasing issues.

RQ-4.1 How well can regular expressions be built to retrieve the entities from the certificates?
RQ-4.2 How much performance of the regular expressions is lost when executed on imperfect machine output?

Chapter 7 describes our pipeline once (imperfect) transcriptions of the certificates are obtained. The method we focus on in this research is Entity Extraction through regular expressions. We build these in Python on a sample of 100 certificates and test their performance on two other sets of 100 certificates. A third set contains certificates with our machine output from Transkribus, which gives us useful insight into the performance when used in real setting. This thesis chooses regular expressions over any other (more complex) Entity Extraction method, because of its simplicity and explainability. If it had turned out to be unfeasible or

performance had been low, we would have explored other options, but this was not necessary, because we do see regular expressions being used for the semi-structured type of data we are dealing with.

In short, this thesis 1) performs a small analysis on the Curaçao death certificates, 2) experiments with multiple different models in Transkribus, 3) investigates whether the existing death register database can be useful for our model, and 4) shows potential using regular expressions for large-scale Entity Extraction for semi-structured handwritten civil data.

1.4 Outline

The remainder of this thesis is structured as follows. Chapter 2 gives a basic understanding on HTR, regular expressions, and Transkribus. Related work for these three topics along with crowdsourcing are written in Chapter 3 to get a better understanding how the HDSC and our thesis relate to other projects. Then, Chapter 4 describes our data and Chapter 5 creates and evaluates our HTR model in Transkribus. In Chapter 6, we investigate the quality of the death register database from the HDSC. In Chapter 7, we continue with the obtained digital text from the certificates and conduct our experiment with regular expressions. The conclusions of this thesis are stated in Chapter 8. Finally, we summarise open research questions and improvements that can be made in future in Chapter 9. Appendices A, B and C contain more examples of death certificates, instructions on how to work with our models in Transkribus, and the named groups used in the RegEx-component, respectively.

Chapter 2

Preliminaries

This chapter introduces Handwritten Text Recognition and its primary evaluation method. Then, we give a short overview of some functionalities of Regular Expressions. The chapter ends with a general explanation of the Transkribus components that are useful for this thesis.

2.1 Handwritten Text Recognition

Handwritten Text Recognition (HTR) is the task of extracting text from an image to its digital text format. It differs from Optical Character Recognition (OCR) which handles machine printed characters, e.g., from a typewriter. The recognition of isolated handwritten digits is actually one of the famous benchmarks in Machine Learning (MNIST database [Lecun et al., 1998]).

HTR is a very active research area and more difficult to solve than most OCR problems. It is because of the difference between recognising individual characters and cursive text. How cursive text should be segmented is not straightforward; it is even a known paradox in literature (Sayre's Paradox) stating one cannot perform a good recognition without the text being segmented, but in order to obtain a good segmentation, one should recognise the text first.

To overcome this paradox, current Machine Learning methods are segmentation free; the recognition and segmentation are done at the same time. State-of-the-art uses extensions of LSTM-models [Puigcerver, 2017, Fornés et al., 2017, Pedersen et al., 2022], although attention-based models are becoming more popular as well [Dahl et al., 2023, Kang et al., 2022]. This is the advancement needed to go beyond OCR and enables the model to learn language (words and sentences) too. So, current HTR models incorporate language.

The neural network that is used by the Transkribus engine is Pylaia [Puigcerver and Mocholí, 2018] which is an LSTM. Section 3.2.3 discusses this network in more detail.

2.1.1 WER and CER

A popular way to evaluate HTR performance is to calculate the Word and Character Error Rate (WER and CER). The Character Error Rate (CER) indicates the number of transformations that are needed to go from the predicted text to the ground truth data. These transformations can be substitutions, deletions, or insertions. Dividing the number of transformations by the number of characters in the ground truth, we obtain a measure how many

characters are incorrectly predicted. This means CER should be as low as possible, with CER=0 being best (i.e., predicted text is fully correct). The Word Error Rate (WER) indicates how many words are incorrectly transcribed, so it works on word-level. Every word containing an error already increases the WER. That is why WER is expected to be higher than the CER-value, because character errors are often spread over the multiple words. Again, WER=0 means the predicted text aligns perfectly with the ground truth data. In general, a CER-value around 5% (=0.05) is seen as more than sufficient and similar to a human error rate, e.g., in transcriptions.

2.2 Entity Extraction

Entity Extraction is a text analysis technique that pulls out specific data from semi- or unstructured texts. In this thesis, we wish to extract entities from semi-structured death certificates and put them in a structured database.

Nowadays, state-of-the-art often uses attention-based transformer models [Hamdi et al., 2021, Ehrmann et al., 2023]. These are big language models with parameters tweaked to understand a particular language. These models are particularly used for Named Entity Recognition (NER) which classifies Named Entities such as persons, organisations and locations in unstructured texts. Transformer models can also be fine-tuned for other language tasks, e.g., sentiment analysis or text generation. A very well known transformer model is GPT(-3/4)[1] used in the popular chatbot Chat-GPT[2]. Although these AI models become better and better, its inner workings remain a black box; one does not know why the model made a certain decision. This means we also cannot directly see why the model made a false prediction.

Another method in Entity Extraction is using Regular Expressions (RegExes). These can obtain better precision, but often at the cost of a lower recall. Building the RegExes can also be very time-expensive, though, creating ground truth data for transformer models also takes a considerate amount of time. A benefit of the RegExes is that they have more explainability, i.e., we can trace back why certain false positives or negatives arise. Because our certificate data is semi-structured, this thesis analyses whether a simpler method such as RegExes would suffice our Entity Extraction.

2.2.1 Regular Expressions

A Regular Expression (RegEx) is a pattern that matches a sequence of characters in a text. One can choose from several options such as finding all occurrences, finding the first/last occurrence, or finding the 'best' match if one allows for some variation.

There are a couple of metacharacters and operators that can be used to make the pattern smarter. The following list gives some examples of possibilities within RegExes useful for our thesis:

- '\d' for digits 0-9.
- '\w' for word characters, these include the letters a-z, but also A-Z, the nine digits 0-9 and the underscore '_'.
- '\s' for whitespace characters.

[1] OpenAI. GPT-4. `https://openai.com/gpt-4`

[2] OpenAI. Introducing ChatGPT. `https://openai.com/blog/chatgpt`

- '\b' is a boundary character, this can be used to only find matches of exactly that word. For example, '\band\b' will only match individual occurrences of the word 'and' and will not match the end of 'sand' or 'land'.
- '.' for any character except newline.
- '\.' for an actual dot in text.
- '$*$' to say zero or more of the pattern in front of it is allowed.
- '+' to say the pattern in front of it should occur at least once.
- '?' to say the pattern in front of it can occur once or not at all.
- '()' can be used to create a group and a pattern can be placed inside and around it.
- '|' for giving options.
- '[...]' to match any character inside the brackets, so 'n[or]' matches both 'no' and 'nr' (abbreviations of the word 'number'). It is similar to RegEx patterns 'n(o|r)' and '(no|nr)'.
- '(?P<name>pattern)' is a named group. This can be used to extract that specific part of the match if used in a larger pattern.
- '+?' this combination will find the shortest possible match due to '?' behind the operator '+'.
- '\W{0,3}' means zero to three non-word characters are allowed, for example, when between words can be a single space (one non-word character) or a comma and a space (two non-word characters), this pattern allows for more variation.

This list is not extensive, more documentation can be found widely on the internet.

2.3 Transkribus functionalities

The Transkribus tool has two different versions. Transkribus Lite is the online web application of Transkribus. It has the benefit of not having to download Java and Transkribus itself, however, it does not contain all functionalities yet. Transkribus Expert is the full version, but will not receive any further updates as Transkribus is transitioning to full use of the web app. In this thesis, our annotators will be using the online version, and we will conduct our experiments in the Expert version.

When using Transkribus, the first step is to upload the images one wants to transcribe to Transkribus. Then, one can immediately use existing HTR models to transcribe the texts, but it is also possible to train one's own models when 1) Transkribus' default models do not perform sufficiently, or 2) one wishes to enhance accuracy further. To train one's own models, it is necessary to create ground truth data in Transkribus.

Most components of Transkribus are free, but the one and most important component of Transkribus (using HTR to create transcriptions) costs credits. One credit is used per document (image) for transcription. If the document does not contain any handwritten characters and only printed text, costs can be reduced to 0.2 credit as printed characters are more easily recognised. One credit costs between 5 and 30 cents, depending on price packages. For institutions, there is custom pricing. With the Transkribus Scholarship Programme, it is possible to receive free credit packages. Every user also receives 500 credits on sign-up. For this research, we have access to credits from Radboud University. All training of models is free, so we have only used credits when applying our best HTR model on our three sets of 100 certificates, before they were manually revisioned.

In the glossary below, we will explain Transkribus' main components and go more deeply over its different functionalities. Note that we focus on the parts that are useful for this re-

search, so it is not a complete description of Transkribus' functionalities. For this, we refer to the documentation on the Transkribus web page (How-to Guides, Glossary and Documentation for Developers)[3]. A more in-depth explanation of Transkribus' Machine Learning frameworks can be found in Sections 3.2.2 and 3.2.3.

The following terms are elements in the transcript of a document:

- **Baseline**. Underneath each sentence (one or more words) a baseline is automatically or manually drawn to indicate this is a part we want to transcribe. This segmentation happens automatically under the hood when transcribing, but it can also be executed as an individual step to first inspect the layout created by Transkribus and be manually adjusted (e.g, to create ground truth data). This step is necessary because HTR-models do not take images of full documents as input, we first want to split them into separate lines.
- **Line region (polygon)**. Based on the baseline underneath each sentence, a polygon is created which perimeter wraps around each sentence. These polygon-shaped image snippets are then fed to the HTR model. These polygons are automatically created; Transkribus does not provide any tools for us. The baselines, however, can (and should be) manually adjusted. For example, when the last character of a sentence falls outside the polygon snippet, one can draw the baseline a bit further so it does include this final character.
- **Text region**. This is a square or polygon that contains one or more baselines together. It is automatically created after creation of the baselines, depending on how much space is between them.
- **Structure tags**. The structure of one's texts can be defined using tags like 'heading', 'paragraph', 'signature-mark', etc. These tags can be applied on text regions, lines or on separate words. These tags can be used in P2PaLA training (see P2PaLA). In Figure 2.1, one can see a default tag 'marginalia' and a newly created tag 'certificate'.
- **Textual tags**. These tags are used on specific words in the transcribed texts and can be learnt during HTR training, because they are encoded in the text. Some tags can contain properties, e.g., for abbreviations their full meaning. There is also the tag 'textStyle' with properties like 'bold', 'italic' or 'strikethrough' (which are visually displayed as such), and there are two default tags 'gap' and 'unclear' that can be used if the text is illegible. Lines with these two tags can be omitted during training[4].
  Other examples of tags are 'person' or 'place'. However, their benefit is limited, as the model can only learn to recognise tags that were seen during training (no new persons or places). Transkribus will provide Named Entity Recognition (NER) as a separate tool in the future[5]. Furthermore, Transkribus advises to limit the number of properties to only one or two per tag, and they say that text styles (i.e., bold, italic, strikethrough) can be learnt quite well[5]. Unfortunately, we do not agree with Transkribus on the latter, as we encountered quite some annoying parsing errors: "strikethrough:true" occurred partly in the transcribed text, instead of visually displaying it as such. Section 5.2.1 describes this in more detail.

[3]READ-COOP. Transkribus Resource center. `https://readcoop.eu/transkribus/resources/`

[4]We found 'gap' to be unusable in Transkribus Lite as the tag does not work *on* characters, but on a position *between* characters where the text is unreadable. The tag 'unclear' can be used if one is uncertain the particular characters in the transcription are correct, or one has a reason to omit this line from training.

[5]READ-COOP. Transkribus Tag Training. `https://readcoop.eu/glossary/tag-training/`

Figure 2.1: Part of an image in Transkribus that contains text regions with structure tags 'marginalia' and 'certificate', Line regions (light blue polygons) and Baselines (dark blue lines).

The following terms are models that can be trained and applied on documents:

- **P2PaLA**. This model creates regions or regions and baselines with structure tags. One can train their own model if one uses structure tags and apply it on an image to create a layout for it. This model can recognise regions that are visually or positionally distinguishable in an image. So, its benefit is that it focuses on the regions first rather than recognising the baselines.
- **Layout Analysis (LA)**. This model creates baselines and, if desired, merges them into regions. In general, the default LA model from Transkribus works quite well for most sorts of documents, but it can be desirable to train one's own baseline model to enhance performance.
  - **Baseline model** can be trained on one's own data to better recognise baselines in the images. The term 'baseline' should not be misunderstood. In Machine Learning, it refers to a simple reference model to compare one's own models to. In Transkribus, it is a model that is trained to detect baselines in the layout.

  It is different from P2PaLA as P2PaLA recognises the structure (regions and possibly baselines) and enriches them with structure tags, where a baseline model detects only baselines (and can merge them into regions afterwards).
- **Text recognition (HTR)**. The HTR model transcribes the text from the images. It does this per baseline. The ordering of the baselines is like most languages (left to right, top to bottom). One can choose to automatically create baselines with the default Transkribus LA, or use the baselines obtained from a Baseline model or P2PaLA model. One can choose an HTR model from the Transkribus community or use an own-trained HTR model. This HTR step costs credits.
  - **Pylaia** is the underlaying neural network structure that every HTR model is trained on. There used to be a second model (HTR+ from CITlab), but these licenses expired. Unfortunately, all HTR+ models need to be retrained with Pylaia.

For instructions how to use the Transkribus models used in this thesis, we refer to Appendix B.

Chapter 3

Related Work

This chapter investigates the concept of crowdsourcing in the digital humanities and its open questions we should be aware of. We also discuss related work on Transkribus: a user study conducted, and the inner workings of the Transkribus' framework. Then, the chapter discusses two relevant papers. Both show what a full pipeline could look like and have interesting similarities and differences to our work. Transkribus is also experimented with, so these results tell us something about its usability. Then, we quickly visit related work from 'Het Utrechts Archief' working with Chat-GPT. The chapter ends with other work in the HTR domain that applied several different NER techniques, under which regular expressions.

3.1 Crowdsourcing

Crowdsourcing stems from 'outsourcing' to the 'crowd' and emerged when internet shifted in the early 2000's from static websites displaying information to more interactivity. Together, one could co-create knowledge and online communities were built. Especially in culture and heritage, crowdsourcing usually happens with a small number of superusers who are dedicated to the project, instead of mass crowdsourcing happening in commercial sectors. "Heritage crowdsourcing projects are about inviting participation from those who are interested and engaged," according to [Terras, 2016, p. 423-424], "The work is not 'labour' ... it is often highly motivated and skilled individuals that offer to help".

According to [Andro and Imad, 2017], motivations of volunteers that participate in these projects, are mainly intrinsic. To the individual, this can be for entertainment, self-esteem, interest in the subject, but also for collectivist reasons such as feeling useful to the community, promoting heritage and meeting people. The extrinsic motivations are often bonuses that come with it such as small renumerations, or improving e-reputation. [Clotworthy, 2019] even talks about health-related benefits in combination with the social, collective and community-based rewards: fulfilling social involvement can increase quality of life and thereby delay mortality.

A review of studies [Burnett, 2021] shows various reasons why this type of engaged crowdsourcing is dominant in digital humanities. First of all, goals of researchers and institutions can be achieved quicker when working with an engaged crowd. They can build and connect with new groups and communities, and gain insights into user opinions. It also shows relevance of and builds trust to the institution, and it leads to higher levels of public interest and public ownership. Lastly, people become enthusiastic about transcribing historic documents and feel author of the content, which contributes to their happiness and keeps them engaged.

This type of crowdsourcing in digital humanities is more closely related to citizen science than to commercial crowdsourcing. Many definitions of 'citizen science' exist, but all agree it is the 'participation of the general public in scientific processes'. Citizen science can go further than crowdsourcing when the public helps identifying questions or can steer the research into certain directions, opposed to merely producing the data (transcriptions).

Although there are benefits to multiple parties, some researchers also express concerns regarding data accuracy [van Hyning, 2019] [Terras, 2016]. One problem that arises when dealing with multiple transcriptions per document, is that it is not possible to simply take the 'average'. One would need robust methodologies to identify the most accurate transcription without manually having to look at each document. A solution from [Deines et al., 2018] is derived from signal processing: given a set of transcriptions, which one has the most information in it that is also in most fellow transcriptions? This method would be useful if more than two transcriptions per document are available.

There are many online transcription platforms that make use of crowdsourcing, most known are Zooniverse[1] and FromThePage[2]. Their platforms are backed by thousands of users, so multiple transcriptions are obtained easily and fast (mostly English). Smaller projects like in the HDSC need to think of different solutions.

There does not seem to be a one-solution-fits-all for crowdsourcing in digital humanities. When institutions involve the public to build cultural resources, complex issues arise: "How can we integrate the contribution of the crowd with institutional collections? How can we facilitate convergence of professional and amateur knowledge? How do we assure the quality of the crowd-contributed content? How can we design a system that supports the combination of crowd-contributed content and institutional content?" [Carletti et al., 2013, p. 13].

Also, a recent development: AI into the archives. With HTR models improving, not only questions arise about crowdsourcing, but also how to integrate AI into the process. So far, there has been next to no research what best practices are in storing, sharing, and explaining HTR generated content [Terras, 2022]. We see this also holds for combining AI and crowdsourcing in science / citizen science.

We found one paper [Ponti and Seredko, 2022] that studies citizen science and its task allocation between experts, citizens, and AI. Their framework compares different types of tasks in multiple stages of scientific projects. They conducted a literature review of 50 papers in which they characterised citizen science projects based on the nature of the task for the citizens, and the skill needed. Also tasks performed by experts and AI were included. One of the findings they had is that some tasks formerly executed by citizen scientists, are very suited for AI. So, they argue that the combination of AI and citizen science may disincentivize certain volunteer groups. "If the only thing you are good at is replaced by AI/ML that can make you feel left out and useless" [Ponti et al., 2021, p. 8]. Integrating AI in citizen science might leave only tasks for volunteers that are either too simple or too complex, disengaging the crowd. With that, the living community and the connection between the institute and its volunteers could potentially be lost. Therefore, it is important to remain allocating tasks to citizen scientists alongside experts and AI in a meaningful way.

[1] Zooniverse. `https://www.zooniverse.org/`

[2] From The Page. `https://fromthepage.com/`

3.2 Transkribus

Transkribus is a widely used platform. There are many individuals using it for private reasons, but it is also used by institutions and archives. In particular, Europeana[3] is a project in collaboration with Transkribus enriching Europe's digital cultural heritage. Another example is Read.Hanse.Sources![4], a citizen science project transcribing manuscripts from the Hanseatic period. Also, the Dutch Archives ('het Nationaal Archief') is working on new AI models using Transkribus. They created the IJsberg model together with 'Noord-Hollands Archief'[5]. The model is trained on 6,444 pages (1,7 million words) on old Dutch texts from the VOC from the 17th and 18th century and from notarial archives from the 19th century. Due to its large size, it is used as base model in our HTR training. In the final month of this thesis, a new Dutch base model was publicly released which contains the data from the IJsberg model plus three other big models (Amsterdam Notarial Super Model, Dutch_XVII_Century, and Dutch Mountains 18th Century). We retrained our HTR using this base model in Future work 9.2 to see its performance.

The next section tells us something about the user satisfaction of Transkribus. This is followed by two sections explaining the inner workings of Transkribus, which provides us a better understanding of what happens 'under the hood' once working with Transkribus' interface.

3.2.1 User study

A survey [Terras, 2022] conducted among 155 Transkribus users (approx. 800 active Transkribus accounts in the survey period) gives insights into the Transkribus platform. In this survey, there were only 4% of the respondents who said the generated results were very accurate and required little correction, 34% said results were quite accurate, 16% said results were disappointing, and 8% found their results unusable. Also, a significant part (21%) acknowledged that the results were very variable and dependent on individual texts. The remaining 17% could not comment yet on Transkribus' transcription quality on historical texts.

The survey also gives some insights into the efficiency. 21% of the respondents noticed a significant increase in efficiency to their projects when using Transkribus, 23% stated it was a useful increase. Many were still training and trialing the software (36%). Unfortunately, 12% said that it had not sped up the processes of generating transcripts from historical texts. Respondents who commented on the time efficiency saw a reduction in transcription time by a factor of 3 or 4, a decline of 10% in time, or, 80% save of time with small loss of accuracy (3-5% CER). A last person noted that they are hoping to save time as their current process takes two rounds of transcriptions from scratch. This compares to the crowdsourcing in the HDSC projects.

3.2.2 Baseline and Polygon creation

The algorithm Transkribus uses for baseline detection is an ARU-Net described in [Grüning et al., 2019]. It produces 'mask-images' that indicate with small vertical lines the beginning and ends of each baseline. The post-processing (from these mask-images to creation of the

[3]Europeana. `https://europeana.transcribathon.eu/`

[4]Read.Hanse.Sources! `https://fgho.eu/en/projects/hanse-quellen-lesen`

[5]Noord-Hollands Archief. IJsberg model. `https://noord-hollandsarchief.nl/ontdekken/nhalab/project-transkribus-2`

baselines) is an ongoing process by Transkribus itself. Due to licensing issues, they are not able to use the post-processing from the paper (described stage 2, baseline estimation).

Between baseline detection and HTR, there is one additional Baseline2Polygon model which transforms the baselines into polygons containing the sentences. These are snippets including the ascender/descenders of the characters. With some extra padding, these are the final line images that are fed into the HTR model. According to Transkribus, it does not necessarily hurt if not all ascender/descenders are cut properly, the HTR model can also learn the characters without it, also its language model can compensate such effects. This was explained to us in email contact with the Transkribus support team after noticing that the Baseline2Polygon did not give perfect results, e.g., baselines only starting halfway the first character. The additional padding prevents too early cut-offs in the polygon segments.

3.2.3 Pylaia

For the HTR, Transkribus uses neural network Pylaia. It is a PyTorch-based deep learning toolkit for handwritten document analysis. Its source code is publicly available [Puigcerver and Mocholí, 2018]. Pylaia is created by Joan Puigcerver who has written the paper "Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?" [Puigcerver, 2017] on which Pylaia's architecture is based. Puigcerver argues that Multidimensional Long Short-Term Memory (MDLSTM) networks might not be strictly needed to achieve state-of-the-art performances and that one-dimensional blocks also suffice. Pylaia's network consists of five convolutional and five recurrent blocks, with one final linear layer at the end. Figure 3.1 gives an overview of the architecture. We refer to the paper for a detailed explanation of each block.

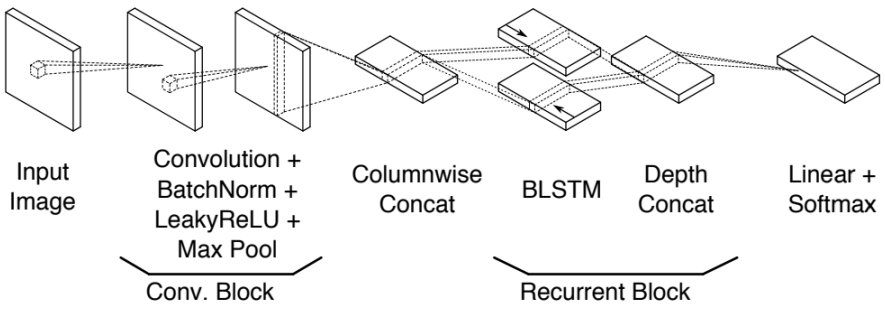


Figure 3.1: Pylaia's architecture using 1D-LSTM as presented in the paper [Puigcerver, 2017].

Puigserver reduced the computational costs drastically without losing any or little accuracy from multidimensional layers. Besides presenting a better architecture, Puigcerver also shows that error rates are significantly reduced when performing certain random distortions in the training images (data augmentation).

The input for the model is a polygon snippet of one handwritten sentence (an image of one line). Transkribus is an easy-to-use interface built on top of the network to create the separate input sentences/lines from the image data that consists of multiple lines. The toolkit of Pylaia has options to train the HTR model, and to decode the text line images so it returns the text. It is also possible to output the character and word segmentation boundaries of the

symbols. Because Transkribus is built on top of Pylaia, this thesis does not further touch or alter any of Pylaia's workings.

One of the reasons Transkribus works surprisingly well on very small datasets is due to its data augmentation. Around 10,000 handwritten words (100 documents of a thousand words) should be enough to train a decent model. In general, Transkribus advises to use an iterative approach to improve one's models (adding more data) till a sufficient performance is reached.

3.3 Danish death certificates

A recent work that is similar to ours, is "Applications of machine learning in tabular document digitisation" [Dahl et al., 2023]. They created their own custom model to retrieve entities from death certificate forms, but also experiment with a model in Transkribus. Note that their tabular data (i.e., forms) are significantly different from our semi-structured certificate texts containing sentences. This also means they use different segmentation and HTR techniques. Nonetheless, because they compare their results to their Transkribus results and traditional crowdsourcing, we see much overlap with our problem domain. So, the remainder of this section gives an elaborate summary of the paper. This enables us to relate our research to it on multiple different components.

In the paper, they are looking into ways to obtain transcriptions and prefer to find a fully automated way, because of the format of their data (tabular) which often has big volumes, e.g., population records. They have available to them a database of 250,000 Danish death certificates, which they use to create training data from, but they also select a subset (23,263 documents) that have crowdsourced transcriptions. These are collected over the years by multiple humans at different locations. This enables them to compare the crowdsourced transcriptions against their model output and Transkribus output.

Methods

They first perform a layout classification to collect data of one type of certificate. Then, table segmentation is applied and an HTR model transcribes the entities. For the layout classification, they train a model using 7,000 documents (+2,184 evaluation) and with that model extract 44,903 (type B) death certificates from the dataset of 250,000 Danish death certificates. Splitting the certificates into different types eases further processing as segmentation can be done per type. In our research, we encountered death certificates with two and three text regions, so this split is also made in our pipeline. Their second step (segmentation) is done via standard computer vision operations. They are not using Transkribus' segmentation steps, because it does not work well on tabular data (yet). Transkribus works better on sentences, which is our data format, so we will still opt for Transkribus' segmentation model. The last step in the paper (HTR) is done with an attention-based neural network, suggested by [Xu et al., 2015]. An advantage of this is that the model only requires rough segmentation and does not rely on text baselines like in Transkribus, which is useful for their tabular data. The segments they obtain, i.a., are table entries with entity 'date' (birth and death). These are all manually reviewed twice. Unreadable dates due to segmentation errors are removed. They obtain 11,320 'date' entries with 1,000 evaluation samples which are used to train their HTR model.

Their HTR model actually performs better than Transkribus. We suspect this is because of the dictionaries they implemented in their neural network. The data they experimented with

were ‘dates’ (only digits). So, their dictionary is small. This reduced the problem space which benefited their attention-based model. Transkribus with its ‘one-size-fits-all’ infrastructure (and without these dictionaries) did not profit from this. So, it can be task specific that Transkribus performed less. It does show that other HTR methods outside Transkribus exist and that there are possibilities other methods can work better for our problem too.

Results

To get a baseline indication of the quality of the crowdsourced dataset, they compare the crowdsourced data against their own created ground truth data. (Their training+evaluation had an overlap of 2,864 dates with the crowdsourced dataset). For ‘date’, they find that 96.3% of these dates are identical. What we learn from this is that crowdsourced datasets contain mistakes / noise. Presumably, their ground truth is not perfect either, although it was reviewed twice. After their model finished training, they find 90.5% dates from the 1,000 samples to be identical. When comparing the model’s predictions on the 46,526 dates that also have available crowdsourced data (after filtering out the samples used during training), the HTR model and the crowdsourced dataset are identical in 83.66% of the cases. This is likely a bit higher due to noise in the crowdsourced dataset, but lower than 90.5% due to possible errors in prior pipeline steps (layout classification and segmentation). The authors argue that a performance of 83.66% might not be acceptable in some cases, but definitely in cases data is large and would otherwise have been infeasible to transcribe in the first place.

Unfortunately, their model in Transkribus only reached a score of 73.9% and took more work. Transkribus failed to segment the tabular structure well, therefore, they decided to transcribe the full certificates and manually extract the dates. They do not provide any details about training in Transkribus, so we suspect they used Transkribus’ default models. Comparing to our own thesis, we agree that these default models are a ‘one-size-fits-all’ (able to handle many forms of historical texts) but are not great when dealing with specific data. However, we believe much performance could have been gained (increasing the 73.9%) when Transkribus was trained on their own data, creating a more specialised pipeline in Transkribus. This might not have been the focus of the authors, though.

Since we want to extract many entities and not only one, and, we handle semi-structured texts and no tabular data, we expect that Transkribus would be easier to use for us than adopting a custom approach similar to this paper. Also, their use of small dictionaries would be unfeasible in our case due to high variability in, for example, names. Comparing the paper to our research problem, we cannot directly conclude Transkribus is impractical.

Automated checks

An interesting point the authors make is the possibility of automatic verification resulting from the relationships between fields. Their entity ‘age’ relates to the entities ‘date of birth’ and ‘date of death’. A similar check can be implemented in our pipeline where problematic records can be flagged for manual review. A check suited for our research is the last name of the father/mother, which should be the same as from the deceased; if only a couple characters differ, there are likely HTR errors. Also, a young age and a profession might warn for a false extraction, just like the name of a partner while the marital status is ‘unmarried’.

Segmentation errors

One important drawback the authors forget to mention are the segmentation errors in the method they use. Though it is not the focus point of their paper to evaluate the segmentation (they simply use an existing method), a fully-automated process should incorporate the segmentation errors. It is fine to remove segments with errors from training and evaluation, as long as it is kept in mind that once using the model on large amounts of data, these data will have segmentation errors and one should think about how to handle these in an automated way. This is something we have to keep in mind for our segmentation step as well.

Conclusions

The methods described in the paper show good HTR results and possibilities for an automated process if data is large. Their custom approach works very well but we have to note that their tabular data is much simpler than ours (they only tested entities with digits, no characters). So, applying the same or a similar approach might not perform similarly on our data. Names have a much higher variety than dates, so we cannot use simple dictionaries as they did for dates.

Their pipeline provides good structure and shows how the problem can be split up in multiple tasks. We see benefit in this as each component can be evaluated separately and altered without other components having to change. This independence enables us to evaluate our Entity Extraction model apart from segmentation and HTR. Each component can also be improved individually or substituted by a different method if performance happens to be insufficient. Improvements in the first steps enhance results downstream too.

Although the paper's results with Transkribus are not great, we believe Transkribus performs well enough to apply in our research, in particular because we are not dealing with tabular data (Transkribus' segmentation works well on localising sentences, which is the format of our death certificates). It also seems like the paper did not train any models in Transkribus, so performance can be gained here. We would argue that if we were to obtain a similar performance as obtained in the paper (around 70%), it is sufficient to continue to the Entity Extraction: this enables us to test the use of regular expressions on our certificate data, without having to put in many hours in the HTR. This is desirable as the regular expressions are something we definitely would like to analyse in this thesis. Another current benefit of using Transkribus is its ease of using: layout segmentation, manual transcription, model training, it can all be done in one place. Also, the historians of the HDSC are familiar with Transkribus. We can even use it for entity labelling and do not need a separate annotation tool.

To conclude, we also see benefit in splitting our methods into separate tasks; we can incorporate automatic entity verification by implementing checks, and, we learnt something about the quality of crowdsourced data.

3.4 Handwritten occupation codes

Another recent work is "Lessons Learned Developing and Using a Machine Learning Model to Automatically Transcribe 2.3 Million Handwritten Occupation Codes" by [Pedersen et al., 2022]. It describes their machine-learning pipeline for transcribing occupation codes from the Norwegian 1950 population census. Similar to Pylaia's architecture, their best network

uses both CNN and LSTM layers. Due to implementing a confidence score, they achieve an accuracy of 97% for the automatically transcribed codes and send 3% to manual verification. They also propose a method to verify the correctness of their results by comparing the occupation code distribution from their model's predictions to the distribution found in the training data. If it does not match, biases towards certain digits could have occurred. In this paper, Transkribus also failed to extract the structure (layout) of the source image. This illustrates another example Transkribus not being suited for tabular segmentation.

Their implementation and decision analysis of the confidence score is interesting as this might help our pipeline too to decide which certificates should be manually reviewed. In this thesis, we will see that the HTR contains biases towards certain words/characters and it has, in particular, difficulty finding the right spelling of names. Certificates the model is not certain about could be filtered, so HTR errors are reduced as much as possible in our Entity Extraction model.

In their evaluation of errors, they spot misclassified digits when 1) the original code has been crossed out and a new one has been entered, 2) the occupation code is difficult to read, or, 3) the transcriber mislabelled the data. This matches with errors we found during this thesis.

3.5 Dutch death certificates (Chat-GPT)

A project at 'Het Utrechts Archief' that is running parallel to our thesis is handling Dutch death certificates from around 1970. These are images from type printed texts, which is an important difference to our handwritten data, as it makes the problem less complex (and, essentially OCR instead of HTR). So, the question how to deal with OCR/HTR errors in the Entity Extraction appears less in their research.

Initially, they started exploring regular expressions to extract the entities, similar to our research, but switched to using Chat-GPT in the process. In a prompt, they ask to return a JSON-object and give along a list of entities. Unfortunately, their results are not yet available. A quick test on our own data gave varying results, e.g., Chat-GPT completely switched up some entities, or it did not return a JSON-object as asked. So, we further keep our focus on building regular expressions ourselves and assess those results in this thesis. Chat-GPT is still being explored by a colleague of ours in the HDSC.

3.6 Other HTR+NER techniques

This section briefly describes two competitions in which teams built their own HTR+NER models. We argue that NER, in general, focuses more on finding broad entities in unstructured texts (e.g., persons, organisations and locations). The terms NER and Entity Extraction can be used interchangeably here, because they also use semi-structured data (similar to ours). In these papers, some opt for a sequential HTR and NER like us, others examine the performance of a joint end-to-end model. All adopt complex neural networks for the NER. In contrary, we would like to investigate whether a simple model using regular expressions suffices.

ICDAR2017 Competition on Information Extraction in Historical Handwritten Records

Firstly, a competition was held with similar data in the "ICDAR2017 Competition on Information Extraction in Historical Handwritten Records" [Fornés et al., 2017]. Multiple teams were challenged to build a system that could extract relevant information/entities from historical handwritten text images. The dataset used for this was 125 pages of the Esposalles database [Romero et al., 2013] containing historical handwritten marriages records from the Archives of the Cathedral of Barcelona. It was written in old Catalan and each record contained not exact but similar information (husbands occupation, place of origin, husbands and wife's former marital status, parents occupation, place of residence, etc) written in some sentences.

One of the teams published a paper [Prasad et al., 2018] in which they experimented with several approaches. In one of their approaches (CITlab-ARGUS-1), they create regular expressions. Unfortunately, there is no detailed information on how they have done this. It is also not one of their best models, but we cannot inspect why. The Esposalles database is not used that often, so we were unable to find more approaches extracting information using regular expressions in the combination with this dataset.

Mainly, teams adopted complex neural networks to solve the entity tagging. The benefit of sequential HTR and NER [Prasad et al., 2018] is the ability to visually see errors occurring after each step. One can analyse the errors, potentially filter those, and have more knowledge of what is used as input to the next components. Executing the tasks jointly has the downside that the NER is affected by mistakes in the HTR transcription. However, former work suggests that it can also lead to similar or better performances, which was the motivation for [Carbonell et al., 2018] to create an end-to-end model.

The PhD-thesis written by [Toledo, 2019] explored different Deep Learning approaches and developed new Information Extraction techniques for loosely structured handwritten documents. Toledo composed the benchmark dataset and the set of metrics which facilitated the ICDAR2017 competition.

ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction

The "ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction" [Huang et al., 2019] two years later also has similar objectives. Its third task was key extraction from scanned receipts. Although these texts were printed instead of handwritten, the rest of the problem follows a similar process (extracting a number of key fields on images which often follow a certain recurrent pattern). The top method uses a lexicon and regular expressions. However, this method only reaches 90% and the runner ups are closely following, so more complex neural networks worked as well in this case. This is evidence that simpler models using a lexicon and regular expressions can perform similarly. However, there is still large space for improvement. It is interesting to see most of the submitted methods use very different ideas and approaches. This problem domain is quite new and presents open research issues; there is no one best model yet and one can expect to see more innovative approaches in upcoming competitions and field of research.

Conclusions

What we can conclude from these related works, is that often it is immediately chosen to use more complex NER models. Although research is done in methods that do the process all

at once, benefits are also seen in cutting down the process into separate problems. Because we will split our research into separate tasks (segmentation, HTR, Entity Extraction), one can assess them separately. We would like to find out whether regular expressions can be a method to extract the entities from the transcriptions, which we believe to be possible for our Transkribus output. Even if its output is not great and contains HTR errors, we can still evaluate the performance of the RegExes. Also, there is more explainability in our model if cut down in parts and it is easier to assess where most improvements can be made. Because the RegExes are a set of rules, it is easier to debug and maintain too. If our HTR method in Transkribus does not give sufficient results, training another HTR model and substituting this into our pipeline can be done without touching the Entity Extraction step.

Chapter 4

Death certificates Curaçao

In this chapter, we analyse and describe variations encountered in the death certificates. We start by describing some interesting aspects about the layout of the certificates, then we will go over characteristics of the written text and show an overview of the entities we wish to extract [RQ-1.1]. Then, we describe how we constructed useful subsets of data [RQ-1.2]. We close this chapter with an explanation how our datasets got transcribed and labelled to use these in consecutive chapters.

4.1 Method

To get a better idea of the data we are dealing with, we conducted a small data exploration in which we just quickly went over many certificates and noted down abnormalities. Not only did we look at our training data (100 certificates), we also manually skimmed through many of the available data to find characteristics of the full data (70.000 certificates). We kept in mind the workings of Transkribus to make assumptions what characteristics of the data could influence results in Transkribus, e.g., damaged certificates. For each characteristic in the next section, we have shown an example certificate in Appendix A.

4.2 Layout characteristics

Most of the certificates are 'normal' certificates which we define as one of the two following types:

- **Clean certificates** (without marginalia[1]). These certificates are well readable, do not have any cut-offs and we expect it to be handled well in Transkribus (no difficult layout).
- **Certificates with marginalia**. About two-third of the certificates contained one or more marginalia in the dedicated space in the certificate. This can be one line, multiple lines, or multiples lines with whitespace in between. In some formats, the certificate number is written in the marginalia; we must extract this. Often, the marginalia contain unimportant information, e.g., which words are approved to be strikethrough; this, we can ignore. In some cases, an entity in the certificate is crossed out, e.g., the month. Then, the correct entity is written behind the crossed-out word, or it is written in the

[1]Marginalia are marks made in the margin of a document. In our case of the death certificates, one whole region on the left side is dedicated for these marks.

marginalia. The latter will be difficult to impossible to extract for a simple RegEx model, since it does not have any contextual knowledge to connect the crossed-out word with the word in the marginalia that is far away from it. We will keep these cases in mind, though we expect this problem to be better suited for a more complex NER model.

Another aspect that occurred in a substantial number of certificates is strikethrough. We must investigate how this is handled by Transkribus and how this influences our Entity Extraction model.

- **Strikethrough in certificates**. In some certificates, e.g., those of stillborns, some words can be crossed out. These are often the printed inline words, so one could write their own sentences in the certificate. It could also be to revise one's name, or a date that is incorrect. Another example is more subtle, e.g., "door ~~de~~ [BLANK] ~~en~~ mij ambtenaar voormeld". In Section 4.5, we describe how we deal with strikethrough text.

Besides the majority of the certificates mentioned above, there are certificates that occur every once in a while with other layout aspects we suspect to be more difficult to handle in Transkribus:

- **Certificates with another certificate at the top/bottom/left-side/right-side**: a simple segmentation model will retrieve all snippets containing text in the image, so, Transkribus will see the text from other certificates. Although there might be workarounds to not extract these texts, it could create noise for our RegEx-extraction method. So, we decide to include these samples in our datasets to see if we can train Transkribus to ignore other parts than the main certificate text in the middle.
- **Certificates with lots of whitespace**: some certificates have lots of whitespace between printed text, e.g., due to small handwriting or short words. We include these in our data too to see if Transkribus is able to draw one long horizontal line, even if there is some whitespace in between.

We also found the following subsets:

- **Birth certificates:** one subset included birth certificates by accident instead of death certificates. We include one in our validation, as the layout is the same, only the printed text is different. So, it is similarly structured as a death certificate but just a different text format. We hope including this in our validation data decreases biases as the model is validated on pieces of printed and handwritten lines that it does not have training data for, so the model is forced to better recognise the actual characters.
- **Certificates with three regions** (marginalia-certificate-marginalia): this format includes all certificates ranging from 1831 to 1869. Because we do not focus on these years, we will only build a segmentation model suited for two-columns. If needed, another model for three-columns can be trained.

The following points are some exceptional cases that occurred very little. For each, we ran some samples through our Transkribus layout model (from Chapter 5) to see its behaviour.

- **Small inserted piece of paper as certificate**: by default, Transkribus will read all lines, so, also the certificate partially visible behind it. We found Transkribus' segmentation indeed creates lines for each line of text, so an extra piece of paper can be read but will have a lot of noise around it due to lines of text on the background.

- **Vertical text in certificate**: Transkribus can read text vertically, however, we noticed this is only possible if all text in the image is vertical, so reading a small snippet vertically fails.
- **Big tear in certificate**: these certificates have some tear in them causing the lines to disconnect. If the two snippets are close enough to each other, Transkribus is still able to draw one line underneath it.
- **Piece is missing**: these certificates also have a tear so some characters or words can be missing. A whole cut-off (e.g., a missing corner piece) is also possible. Transkribus' segmentation will create baselines where it sees text. This does not lead to problems, though the full original text cannot be retrieved.

Besides these variations in certificates, there can be other exceptions, such as fully shredded certificates or an image that is not a certificate. We exclude these and the last list mentioned above further from our research. Note that the lists from this section may not be complete, as not every data sample is looked at individually. This small analysis just served to get a quick idea of our data.

4.3 Text and Entity characteristics

As already mentioned in the introduction (i.e., see Figure 1.2), the death certificates have a mix between handwritten and printed words throughout each certificate. We found a couple of variations in the printed text over the years, e.g., *"compareerde ten burele voor mij ..."* or *"compareerde voor mij ... ten mijnen bureele"* or *"verscheen voor mij ..."*. We make sure these variations are sufficiently present in our datasets. Because the number of text formats is limited, and entities are often properly written at their designated place, this makes our certificates more structured than other types of documents.

Information	**certificate**	**informants**	**death**
	certificate_number	name_informant_(1/2/3)	date_of_death
	certificate_date	age informant_(1/2/3)	time_of_death
	certificate_district	profession_informant_(1/2/3)	place_of_death
Information	**deceased**	**parents, partner**	
	first_names	name_father	
	last_name	profession_father	
	age	deceased_father	
	profession	name_mother	
	date_of_birth	profession_mother	
	place_of_birth	deceased_mother	
	sex	name_partner	
	marital_status	profession_partner	

Table 4.1: Entities in the death certificates of Curaçao.

Table 4.1 shows the entities targeted for extraction from the death certificates. Each certificate starts with information about the certificate itself: the certificate number, and where and when the certificate was drawn up. Then, information about the informant(s) is

given (full name, age, profession). Some formats only have one informant, some two, and some also describe one or two additional witnesses. The information about the deceased can be split up in two: information about their death (date, time, place) and personal information about the deceased (last name, first name(s), sex, age, date and place of birth, their profession and marital status). It is also possible that more information is given in the certificate, such as information about the father or mother (full name, profession, whether they have deceased (Y/N)) or information about their partner (full name, profession).

These entities align with columns found in the Excel sheet of the unknown crowdsourcing. Two differences we made is that we merged 'place_of_birth' and 'country_of_birth' into one single entity, and we chose for the full certificate_date instead of only the year.

4.4 Overview datasets

Since we are dealing with many data, but data labelling is time expensive (transcribing one image takes several minutes), creating data for this research is expensive. Therefore, we carefully select data samples we want to use for training, validation, and testing. The training set has size 100. Before we joined the project, two annotators were already asked to pick 50 training samples, where annotator A focused on having variation in the visual layout of the certificates, and annotator B focused on having variation in the handwriting styles. Both annotators took samples from random years within the range of our interest. There are no duplicates in our training data. In addition, we selected 15 validation samples on top of the training data, to increase the data size a bit more (and not pick 15 from the training data).

The training and validation sets were transcribed in the first two months of this thesis. Then, Sample_known was made in the third month and Sample_regex in the fourth. These names will be used throughout the remainder of the chapters. This list below shows an overview of the Curaçao data used in the remainder of this thesis.

- **Training_set** size=100. These samples are taken from the years ranging 1879-1895, 1905-1909, 1930-1939 and 1945-1949. 50 of them are selected to have much variation in the visual layout, others are selected to have many different handwriting styles. As starting point, an HTR model trained on Suriname data was applied on the certificates (with no great quality). The set is transcribed and labelled as described in Section 4.5. The 100 samples serve as training data for the HTR model in Transkribus. In our training set, there are 24 samples without marginalia and 75 with one of the marginalia types. We also made sure that there are certificates present that have other certificates at the border, lots of whitespace, or some strikethrough, because this is important to train in Transkribus. Our validation set also follows the same pattern.
- **Validation_set** size=15. Taken from the same years as the training data. It is created similarly as the training data. These certificates are used in the validation step of the HTR training. We made sure that the (important) variations found during our layout analysis in Section 4.2 are sufficiently present. So, there are 3 certificates without marginalia and 11 with marginalia in our validation set. In addition, 4 certificates have another certificate at the bottom, 2 above, 2 at the left and 2 at the right. There are 9 certificates without another certificate visible. There are also 3 certificates with lots of whitespace and 7 certificates contain strikethrough. Lastly, we chose 1 certificate to be a birth certificate (which is not in the training data but follows the same visual

characteristics). By doing this, we seek to reduce bias as the HTR model is more forced to actually read the characters instead of learning patterns in the death certificates.

- **Sample_known** size=100. These samples are randomly taken from years 1831-1878, 1896-1904, 1910-1929, 1940-1944 and 1950 (i.e., years not falling in the range of the training data). They are firstly transcribed by the HTR model (to save time), then, corrected by the annotators. This data is expected to have a database entry in the death register database which is transcribed by unknown people. Sample_known is then used to compare the quality against the database, but it also indicates in our RegEx-experiment how well the RegExes will perform on data from different years.
- **Sample_regex** size=100. An additional set with certificates from similar years as the training data. First, transcriptions are generated by our HTR model. Then, in a copy of the data this is corrected by the annotators. Because we did not train the HTR model on this data (yet), this set serves as a good test set to view the HTR quality. In the final part of our research, it also shows how well the RegExes would perform on imperfect data if one would take HTR from the machine output, instead of the human-transcribed text.

4.5 Data annotation

Data is annotated through a multi-step process which consists of four phases: transcribing, checking, labelling, and checking. Half of the data is done by one of the two annotators. Checking is done on the other half of the data to revise each other's work. Transcribing does not happen from scratch. We already corrected the layout before an HTR-model returned the text. The annotators see the model's output where quality was quite bad in the first set (Training_set and Validation_set) and became better in the second (Sample_known) and third dataset (Sample_regex) as the model was already trained on the first set of data.

We asked the annotators about their transcribing process. They indicated that they would first quickly glance over the certificate to spot severe mistakes, then they would go over each word individually. In particular names got extra attention as these have a lot of variety. One can learn the handwriting style and recognise the characters by looking at other more readable words in the same certificate, one can also look if the signatures are of any help to the correct spelling of the names.

We also made the annotators aware of possible biases while transcribing. Because the HTR model pre-filled in words, it is possible that the annotators skip over words too quickly and assume them to be correct. In particular, in the second and third dataset in which the model is already quite good, it is more difficult to spot mistakes. One might also be biased to not adjust a name if one doubts between two options and one is already given by the model's output. These biases might be avoided when manually transcribing from scratch, but it increases transcription time, so we chose not to do this. We hope increased awareness already reduces mistakes.

For the labelling phase, we created an annotation guideline which was revised along the way to remove ambiguities. We instructed the annotators to label entities which are explicitly stated e.g., names and dates, but also to label words one can deduce an entity from, e.g., the sex can be deduced from either the word 'echtgenoot' (husband) or 'echtgenote' (wife). This labelling process is different from the current crowdsourcing method as the annotators label the words in Transkribus. So, labels are linked to the words in the plain text and the baseline

in the image. The entities are not written down in empty fields of a form which is filled in by volunteers in the current crowdsourcing process. This has the benefit that we are able to link every entity to the position in the text and image, its downside is that the entity must be taken exactly from the text; one cannot adjust the words in the entity, e.g., if part of a date or name is crossed out, the entity label is split over a correction in the marginalia and the label in the certificate itself which is difficult to combine into one entity again.

Unfortunately, we found a bug when exporting the labels to Excel: not all entries were present, so data was not always complete. We could not find any correlation why certain entries were missing, this also varied over time when executing the exact same export. We worked around this bug by using the XML of the metadata. An HDSC-team member wrote a function to extract the entity labels from the XML.

Another choice we made in Transkribus is to label signatures with Transkribus' default text tag 'unclear'. This enables the option to remove these sentences from training and analyse the differences in including or excluding the signatures from HTR-training. As signatures are very unique with different styles and often do not form any actual word, they might add noise to the model. In Section 5.2, this question is analysed. Also, another design choice we made was to include strikethrough to our data. Transkribus has a nice interface in which crossed out words are also displayed as such. Transkribus claims that text styles can be learnt quite well by the HTR-model[2]. As some certificates contain strikethrough and the historian are wondering how to deal with these, this strikethrough is also analysed in Section 5.2.

In chapter 6, the quality of the annotations is evaluated in combination with the available database. In chapter 7, we find mislabelled entities and more false negatives as these are correctly matched in our RegEx-patterns.

4.6 Conclusions

When manually skimming through many certificates, we found that the majority are good quality scans, i.e., readable and no unusable characteristics. We suspect this data to be suited for HTR in Transkribus. The death certificates this thesis focuses on has a two-column format; at the left, there is room for marginalia, and the right contains the main certificate text. There can also be other certificates visible at the sides of the scan. Some certificates have a lot of whitespace, this is due to the mixture between printed and handwritten words in sentences. It also causes some certificates to contain strikethrough, e.g., in the case of stillborns, some words are crossed out and other sentences are written down.

When determining our training and evaluation sets, we made sure these characteristics are sufficiently present. We created a training set (size=100) and validation set (size=15) to train our HTR model. The training set is also used to build our RegExes. The additional two datasets (both size=100) serve as evaluation sets. This will help us analyse the existing death register database, and it will tell us something about the performance of the RegExes.

[2]READ-COOP Tag Training. `https://readcoop.eu/glossary/tag-training/`

Chapter 5

Handwritten Text Recognition with Transkribus

This chapter describes the motivation, process [RQ-2.1] and evaluation [RQ-2.2] of the first components of our model from image data to text files. We opt for the best HTR result possible within reasonable time spans. Transkribus is used for detecting the sentence segments (Layout Analysis), and for training the HTR model to obtain the plain certificate texts. After analysing the HTR training results, we acted upon some serious errors and improved these with two additional layout models.

5.1 Method

Before we are able to match regular expressions on the certificates, we need the plain text. This two-step process for retrieving entities is chosen, because previously the HDSC ran a pilot attempting to locate the entity fields visually, which proved to be too sensitive to errors.

Initially, the segmentation and transcribing were not a focus point of our thesis and we would only focus on the second part investigating how to obtain entities from plain text. However, once we joined the HDSC and started our thesis, this text data was not yet available to us, so we proceeded from there and took over the work in Transkribus. Soon we noticed how important these segmentation and HTR steps are.

Although part of our research investigates how to deal with errors from imperfect transcriptions in our entity matching, our goal is also to obtain the best possible results. Errors are accumulated; repairing mistakes in a later part of the process is more difficult perhaps even impossible. So, we want to reduce errors in the Layout Analysis and HTR as much as possible. Though, we do this taking a reasonable time frame in mind, as we would like to proceed and investigate the Entity Extraction through regular expressions in particular.

The next sections first cover our HTR training in Transkribus, then describe the additional layout models trained for segmentation. Figure 5.1 shows the order in which the components should be applied in real setting. Appendix B gives instructions how to apply these in Transkribus. Note that Future work would like to improve this pipeline, e.g., by adding filters to signal Layout Analysis has failed (Section 5.4.3).

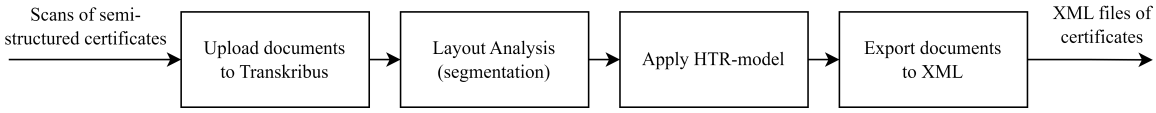


Figure 5.1: Flow of the first components of our HTR+RegEx model.

5.2 Training HTR models

Once the Training and Validation set were transcribed, several HTR models in Transkribus were trained. First, two models with default settings and without any base model were trained to create a simple benchmark. Then, a couple of models were trained using default base models from Transkribus. Unfortunately, these models were unable to learn (CER remained above 90% or 80%). We later find and describe in Future work 9.2 that lowering the batch size does enable the models to learn. Another base model that the National Archives shared with us did give better results, the model with base model 'IJsberg'. We continue using this base model in the remainder of the experiments. Table 5.1 shows a summary of the most important trainings.

ID	Base model	HTR parameters	CER(train)	CER(val)	WER(val)
1.1	no basemodel	default settings	3.7	11.2	33.0
1.2	no basemodel	default settings	3.0	10.8	32.6
2	Transkribus Dutch Handwriting M2	default settings	84.4	91.2	100
3	IJsberg model	default settings	3.0	6.0	20.3
4	IJsberg model	default settings, ignore signatures	2.3	5.0	18.9
5	IJsberg model	default settings, with strikethrough	3.9	7.7	21.7

Table 5.1: Training scores for the most interesting experiments.

We did not make any split between printed and handwritten text. In every experiment, we train one HTR model that learns both the printed and handwritten words. Most OCR and HTR problems only encounter one of the two. We decided to ignore this and set the HTR training to handwritten text. Although our method might improve when dealing with this mixture instead of ignoring it, we decided to not look into this further. We did think of possible solutions if needed. Possible directions we could have taken are 1) making the printed text bold to enforce the difference between printed and handwritten text, or 2) using another textual tag (1&2 are visually different but inherently the same while training), or 3) use two different textual tags for printed and handwritten texts and train two different models where one of the two tags is masked during training. Structural tags do not provide a solution here, since their training is based on locations in images (this is used for P2PaLA-training and not HTR).

Due to all the different certificate formats and our desire to obtain output in the form of sentences that contain the printed and handwritten words together, we do not want to use any of the tags described above. We believe any other split in printed and handwritten to train two separate models is also difficult, because they occur mixed in each sentence in itself. After the HTR training and experiments, we conclude that we see more mistakes in the handwritten words, but this is to be expected. Also, the results are good enough that we decided to not go for any of the above provided possible improvements.

5.2.1 Flaw in HTR-training on strikethrough

Furthermore, Table 5.1 shows the model trained with the strikethrough tag enabled. The lower scores may be a result of the more difficult task, or because of the flaw we found in the Transkribus software: when examining predictions from the model, we find noise in there which has the shape of 'strikethrough:true' but some characters omitted. We contacted the READ-COOP team about this after which they answered 'this is not really a bug, but rather the expected behaviour of the tag training'. We suspect this is due to the way the model is trained indeed. Pylaia is a BLSTM-network that takes as input an image of one line of words, and outputs digital text with tags and properties encoded in this text. In the case of words that should be crossed out, the network needs to output characters 'strikethrough:true'. If part of this property is returned wrong by the model, the interface of Transkribus will see them as normal characters and the strikethrough property is lost. Unfortunately, the READ-COOP let us know they will not make improvements on this in short term. This leaves our question how to deal with strikethrough data an open problem.

5.2.2 Signatures

The last experiment we conducted is training an HTR model without signatures. These signatures are different from normal handwriting, so they might be noising the training. The scores for this model are higher (Table 5.1), however, these are scores over the full text. Ignoring signatures leads to better scores because mistakes in signatures are not taken into account anymore. So, we evaluated the HTR performance on all text except signatures, and found the opposite results. Apparently, training with signatures leads to better understanding of the normal handwritten texts (more data the better), so it is better to keep the signatures while training.

The model has a hard time predicting the signatures (average-CER=45%). However, leaving them out of training worsen performance slightly. This can be due to randomness, or the model might actually benefit from more handwritten data. In any case, leaving them in does not worsen the performance of the entities and other words. They might actually benefit from it, so we advise to leave them in during training, but to keep them in mind when evaluating (one can leave them out of evaluation if they are not important further down the process).

5.2.3 Strikethrough

As an addition test, we evaluated our Sample_regex machine output once we obtained its ground truth (later in this thesis). We did this because it was still unclear to us whether we should use the model trained on strikethrough, or the model that did not take strikethrough into account. We also evaluated the performance without the marginalia to see what the performance is inside the main certificate text, and we give an indication of the performance on names by taking the line of text after the line containing 'is overleden'. This is a quick measure as it also contains other words such as profession if it is on the same line. Table 5.2 shows the scores. The Sample_regex data used for this has perfect layout (manually segmented), but text output from our HTR model.

Model	Sample_regex data	CER	WER
5	Full text	5.01	13.39
3	Full text	3.51	9.43
3	Without marginalia	3.10	8.39
3	Without marginalia and signatures	3.10	8.48
3	Only line with name of deceased	16.06	39.48

Table 5.2: Evaluation on parts of the Sample_regex data with models from Table 5.1.

One can see that scores are again better when strikethrough is not trained on. This might lower when tags are filtered from the certificate text, but for now, we will use the model without strikethrough. Comparing that model, we see better scores in the main certificate text. This means that marginalia are difficult for the HTR model. We suspect this is because it is handwritten and has more variation. Removing signatures does not give an overall better score, which means that many correct words are the signatures itself. This might be due to repeating signatures, e.g., the same official or informant, which are included in the language model.

5.2.4 Names

The last important remark are the lines with the names of the deceased. The HTR model has lots of difficulty recognising the correct name. In Table 5.2, one can see WER=39.48. In Chapter 6, we find reasons to believe some of this might be due to mistakes in transcriptions from our annotators. But mostly, we think this is due to the high variety of names. In Future work (Section 9.2), we mention several techniques that have potential to increase the transcription quality of names.

5.3 Errors in Layout Analysis

After training the HTR models, we looked at the predictions in the validation sets and we noticed many mistakes were not due to the models, but due to the preprocessing steps in Transkribus. As explained in Section 2.3, an HTR model takes a single line as input, so the layout must be segmented first. Transkribus uses its default layout analysis model for this (Transkribus LA). The mistakes boil down to four main problems:

1) **Too little space between sentences from different regions**. Sometimes, a word from the marginalia is just outside the marginalia field, into the certificate field, or very close to the border. If it aligns with a sentence from the certificate, Transkribus will see this as one big line. Transkribus has some parameters to adjust for this, however, trying several options turned out unsuccessful. This is probably also due to how the Transkribus LA model works. It first recognises the baselines, and then creates regions based on these and some parameters. So, when sentences are close to one another, Transkribus fails to see two regions, so all the marginalia text appears in between sentences of the certificate, noising the text for part II of our research.
2) **Too much space between sentences in a region**. In our certificates, there is printed text, then, room for the civil servant to write down information. This can lead to a lot of whitespace in between words if not all space is used up, so, Transkribus creates separate

lines for these parts. This does not have to be problematic, however, Transkribus mixes up the ordering of the sentence, i.e., the right part occurs before the left part. When we export the text for part II of our research, the words are in incorrect order. This makes matching on regular expressions much harder. The reason for the mix-up is due to the pixel height of the top left corners of the line regions. So, a line arrives first if the top left corner is a couple pixels higher than the other line.

3) **Incorrect certificate number baseline**. Many certificates have a wavy symbol underneath the certificate number. Also, the combination of the printed and handwritten characters makes it difficult to determine the position of the baseline. This becomes problematic once the baseline is not correctly positioned underneath 'No.' or the number itself, because the HTR model will not be able to see it.
4) **Text outside the certificate and marginalia**. For some images, there is a part of another certificate at the top, bottom or side. Transkribus creates baselines for every word / line of words it sees, so words from other certificates are added too. This is not necessarily problematic, but it adds a lot of noise to the beginning or end of the text for part II of this research. So, if possible, it is desirable to remove this noise.

For the first and fourth problem, we decide to add structure tags 'certificate' and 'marginalia' to the two text regions in our training data. We make sure the certificate is a rectangle following the printed lines if present in the format. The marginalia can be a rectangular box too, but can also be smaller if the marginalia do not fill up all the space in the marginalia field. If there are two marginalia in the marginalia field, we draw one big region containing them all. If the marginalia is empty, there is only a 'certificate' region. We suspect the model can learn this format, as our data has multiple different layouts, but they follow the same structure (marginalia at the left, certificate in the middle-right of the image). So, lines can be split up if it lays in two regions, solving the first problem. Also, it will not include the words belonging to other certificates anymore, solving the third problem.

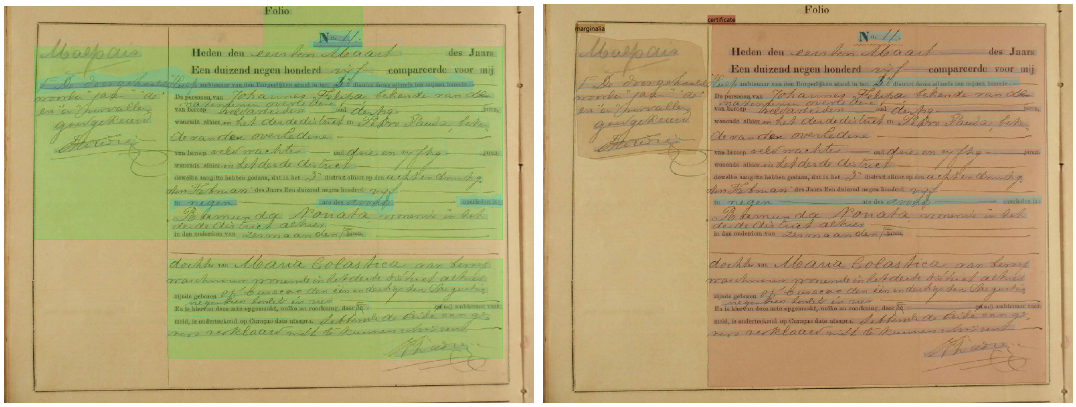
For the second problem, we adjust the training data such that each sentence with lots of whitespace becomes one full line (merge them into one baseline). If we can teach Transkribus to keep them as one line, we eliminate the chances of wrong orderings. This saves us time in the end as we can simply export the certificates as text files. Another solution would be a post-processing step looking at the coordinates of the baselines and writing a function to fix the order. This step can be avoided if our trained model is performing sufficient enough.

Lastly, we hope by giving the LA model enough examples of how the baseline under the certificate number should be positioned that predictions of the model will be better here, solving the third problem.

Figure 5.2a shows an example for problems 1-3 and Figure 5.2b shows what the output looks like when these errors are solved. Section 5.4 will explain how this output is obtained.

5.4 Evaluation LA models

To alleviate the problems described above, we train several models and evaluate which model performs best. We train one Baseline model 'LA_Curacao', and four P2PaLA models. Two of those are trained to only detect regions, the other two are trained to recognise regions and baselines simultaneously. The second difference to the P2PaLA models is that two are trained on 100 training and 15 validation (100/15) as done in HTR training before, and the other two are trained with 100 training and those 100 also used as validation. The reason for this is a bug



(a) Certificate with errors. (b) Same certificate with errors solved.

Figure 5.2: Errors in a certificate where 1) the marginalia is too close to the certificate text, so Transkribus sees them as one line (first dark blue baseline) and creates multiple wrong regions, so marginalia lines are in between certificate lines (green boxes); 2) three pieces of text on the same line are not merged due to the whitespace, therefore, their ordering in the transcribed text can be from right to left if the right part is a couple pixels higher (last three dark blue baselines); and 3) there are two baselines at the certificate number, leading to incorrect HTR output (first two dark blue lines).

in Transkribus that shows a 100/100 split in the overview after training 100/15. Because we were unsure if this was just a visual bug, or something in training also went wrong, we trained the last two models explicitly on the 100/100 split. We expect these models to perform worse in our evaluation, as using data for both training and validation heavily overfits the model. If the models perform similarly to the 100/15 split, this might indicate that the bug also resides in the training.

5.4.1 Baseline model

After training the Baseline model 'LA_Curacao', it became immediately clear (from looking at a couple of certificates) that this model does a much better job at detecting baselines than the default 'Transkribus LA'. Transkribus is a 'one-size-fits-all' architecture, so it works well for many different documents. Nonetheless, using our own data, the model can learn more specifics and patterns in the images, so it is not surprising that training our own Baseline model is beneficial.

Because two P2PaLA models described in Section 5.4.2 are only trained on regions (i.e., they do not return any baselines), we apply this 'LA_Curacao' afterwards to detect the baselines. For the two P2PaLA models that are trained on regions and lines, we also noticed that our Baseline model is better than the P2PaLA itself at detecting baselines. This might be because the task is more focused than training both regions and baselines simultaneously, but can also be due to the fact that P2PaLA has a whole different architecture. For our use-case, the P2PaLA on regions and lines fails to merge a lot of the baselines together that are horizontally aligned (which we explicitly made clear in our training data). So, our Baseline model 'LA_Curacao' is applied after all four P2PaLA models. This removes the baselines from training on regions and lines, but keeps all the regions intact.

Furthermore, we select the option to 'split lines on regions', so lines are cut off at the border of the regions. This avoids that lines from two regions becoming one big baseline. Furthermore, we put 'max-dist for merging' on maximum value 0.99 so there is no width that baselines should not be merged. All other parameters we keep on their default values.

5.4.2 P2PaLA models

For evaluating the four P2PaLA models, we skim through the validation set after one of the models is applied, and give a certain score between 0 and 3 to each certificate to summarise the performance of each model. When regions are misshaped, certain words or characters might fall outside of this region. If it is only (part of) a single or some single characters, we can consider it harmless because tiny mistakes are tolerable due to the padding around baselines. If there are multiple words falling outside the region, it is getting harder for RegExes to find the right matches. So, if there are multiple mistakes, or the layout is completely wrong, we only give it a score of 1 or 0, respectively. Note that this evaluation method is a bit ad hoc, but tries to make the manual evaluation a bit more explicit than just human skimming.

The evaluation on the validation and test set did not bring forward any significant results. That is why we decided to apply the P2PaLA models on the 100 certificates in Sample_regex. There, we clearly saw a difference after examining the first 20 samples. We observe that training on regions and lines performs better than only training on regions. We believe detecting baselines simultaneously can help the training and the determination of the regions. Only training on regions is more prone to miss some lines in its regions. Furthermore, the models with 100/15 split (i.e., an accurate validation set) perform much better on this new data. This provides an example of how important it is to have a separation between training and validating to avoid overfitting. But also, it shows the importance of having a separate test set to evaluate the models, as evaluating on the validation data did not show any performance differences (because all data was already validated on, using it as evaluation step again gave better results than was actually accurate). So, when doing an additional testing/evaluation phase, it is important to use a separate test set again with data that was not in training or validation.

5.4.3 Flaw in P2PaLA predictions

When applying the best model (P2PaLA on regions and lines with the proper validation set) on the remaining 80 samples, we found some samples with a completely wrong, chaotic layout. These 12 samples are similar to the others: 4 have exactly the same format but have a white instead of yellow background, 4 have a new font, but exactly the same words, the rest are samples that are similar to what is seen during training. When applying the same P2PaLA model again over these samples individually instead of over all 100, the result is much better. 10 out of 12 become fully correct. This is an interesting finding as it suggests that the output depends on the input given, where looking at individual cases returns better results than taking all at once into account. So, we suspect that Transkribus feeds all samples together to the Pylaia model. When we apply the model to our 100 samples all at once, the predictions are repeatedly the same, when we apply the model to an individual certificate repeatedly, the result is the same. So, P2PaLA is deterministic, given its input, but not per certificate.

So, to improve performances, it is useful to quickly go over each sample and run the model over individual cases with a chaotic layout, as this occurs in about 12% of the time.

Unfortunately, this becomes infeasible when working with thousands of certificates, but there are some workarounds, e.g., a check after HTR if the returned text is really bad, this might be due to the bad layout, so detecting this one can act on this and run the model again over the certificate, or, finding a way to automatically create HTR jobs for each sample individually instead of one collection.

5.4.4 Revisions on the errors

Based on Section 5.4.1 and 5.4.2, we conclude that we obtain the best layout when applying the P2PaLA model trained on regions and lines, with a proper validation set, and then applying the Baseline model 'LA_Curacao' with the parameter settings as described. Now, from the 100 samples, 91 have a correct layout with all words belonging to the certificate in baselines in the right region (some harmless errors allowed). Two certificates include text from another certificate at the top, two others include some words from the marginalia in the main text. The remaining five certificates have some missing baselines or baselines cut off, so some words or characters are outside the region. These five certificates leading to incorrect baselines are tolerable considering the improvements made for the Section 5.3 problems. Now, we will evaluate these four problems with our best model configuration:

1) **Too little space between sentences from different regions**. First of all, this caused some sentences to be one big baseline. This is fixed by selecting the 'split lines on regions' option after regions are created with our P2PaLA model. From our 100 test samples, there were 4 samples where the default Transkribus LA failed and created one big baseline. With our LA configuration, all 4 samples are now correctly segmented. Secondly, 'Transkribus LA' creates regions after the baselines, so from the 54 certificates that have marginalia, there were 27 that created one region, causing all the marginalia lines to mix up with the certificate lines. Using P2PaLA, our certificates are split in a 'marginalia' and a 'certificate' region, so, from our 27 wrong samples, there are now only two samples left that contain some words in the wrong region.
2) **Too much space between sentences in a region**. Looking at the first 20 samples and applying Transkribus' default LA, there were 15 samples with an incorrect ordering of sentences due to the splitting of baselines. This results in difficult RegEx matching when the ordering is like: "[*dat*] [*overleden is:*] [*ure des avonds*] [*te negen*]" instead of the correct "*dat te negen ure des avonds overleden is:*". The remaining 5 samples were correct, but also their baselines were unmerged. Looking at all 100 samples and our model configuration, most errors disappeared in the important part of the certificate texts. If we ignore the last unimportant sentences (without entities), 84 certificates are correct with merged baselines for the just given example. 11 from 100 still have a wrong ordering that may harm the retrieval of entities, 15 certificates have unmerged baselines but they do not lead to an incorrect ordering. We observe that most mistakes are made in earlier years, as the certificates contain more whitespace in these formats.
3) **Incorrect certificate number baseline**. This is improved by training our own Baseline model. When we look at the first 20 samples, one mistake is due to P2PaLA so discarded, but all other 19 certificate numbers have a correct baseline underneath. For five samples, there was still a second baseline underneath the wavy decoration line, which may lead to some noise. However, all 19 are readable for the HTR model. If we apply the default 'Transkribus LA', we observe in all cases a second baseline (creating noise),

and in only 14 samples the first baseline being correct: four samples missed the actual digits (the entity we want to extract) and one missed the text 'No.' (less harmful, but still a mistake).

4) **Text outside the certificate and marginalia**. This is improved by training the positions of the 'certificate' and 'marginalia' region, and training a Baseline model to not detect baselines at the border of the images. When looking at the default Transkribus LA, it detects baselines of other certificates in 18 samples (simple 'folio' excluded). We wish to ignore the surrounding certificates to reduce noise in the exported text. For our best LA configuration, we observe only two cases where this is unsuccessful (containing baselines of another certificate at the top). So, 16 samples are correctly segmented due to P2PaLA and the Baseline model. We observe that both P2PaLA and the Baseline model contribute to this improvement, as there are a few examples where the P2PaLA includes the other certificate from the top in its region, but the Baseline model excludes these sentences, creating a correct layout.

5.5 Conclusions

In this chapter, we described how plain text is obtained from the certificates. We tried to reduce errors as much as possible to ease the retrieval of entities in chapter 7, balancing quality against keeping reasonable time spans in obtaining this quality. First, the segmentation of regions and baselines is done by Layout Analysis. Then, our best HTR model is applied to obtain the transcription.

For the Layout Analysis, we used two models (P2PaLA and Baseline LA) to get the baselines as good as possible. The regions 'certificate' and 'marginalia' are created using P2PaLA and the best baselines are created with Baseline LA. This combination improved positioning of the baselines after which sentences in the certificate were less mixed (thus less in wrong ordering). An important note here is to keep in mind that about 1 in 10 certificates may return a completely wrong layout when the model is run over all certificates at once. Going over the wrong certificates individually afterwards results in a better version, but that may not be feasible when up-scaling our research to thousands of certificates.

We further find that using the default HTR settings in Transkribus does not work for all base models (especially the ones released by the team of Transkribus). Using the IJsberg model improves our baseline from CER=11 to CER=6. An interesting finding is that signatures might make training slightly better. We suspect the more data, the better the model can be trained. That is why we recommend to keep on enriching the model with more training data in later stages of the project. Whether to use a model that was taught to recognise strikethrough or not depends on the desired outcome. The model can detect some strikethrough, but it also still makes many mistakes, i.e., part of the tag appears as text in the transcription, so the overall HTR score is lower. We do not know whether this impacts the HTR quality of other words. We should also note that it is difficult to draw any hard conclusions, because of randomness in training (i.e., model 1.1 and 1.2 differ while trained with the exact same settings, so each model might have this variance as well).

Based of the revisions in Section 5.4.4, we argue that Layout Analysis is an important step before HTR and should not be forgotten. At first sight, Transkribus looks promising as it can retrieve the sentence segments with a click of a button, however, the 'one-size-fits-all' architecture does not perform sufficient enough on large amounts of data and our set goals;

Transkribus is suitable for helping historians with analyses on small data. Nevertheless, the LA improvements made, the fact that our data is sufficiently homogeneous, and the ease of using Transkribus for non-Data Scientists, make Transkribus a reasonable choice for our HTR.

Chapter 6

Evaluation death register database

This chapter evaluates the quality of the death register database [RQ.3] of which its origin and collection process are unknown. It is suspected that the collection is also made by volunteers in a similar crowdsourcing fashion as in HDSC, however, we do not know whether there was an expert annotator involved. This chapter sheds light on the design choices that were apparently made in the existing database and the HTR quality of this data. The chapter also serves as evaluation of our training data as our annotator judgements can be compared against the database.

6.1 Method

Former research by HDSC-colleagues indicated that certain periods are missing information, i.e., only the name of the deceased is entered in the row of the scan. These periods are 1879-1895, 1905-1909, 1930-1939 and 1945-1949. Therefore, this chapter focuses on the other years in 1831-1950. We selected 100 random samples spread over these years and assume data is available in the database.

These 100 samples were then uploaded to Transkribus. Our models from chapter 5 were used to create the baselines and text regions. This was manually revisioned, because not all certificates fitted the model (e.g., some samples had three text regions and our P2PaLA model is not trained on this). After cleaning the lay-out, we ran our best HTR-model over the certificates. Our two annotators revised this and added the entity labels. One of the annotators also helped us comparing the entity label and the database entry for every entity for all 100 certificates. We did this manually, because we wanted to examine the reason if two did not align.

We have multiple reasons for investigating the available database. Firstly, by comparing our annotations against the database, we get a sense of our annotators' quality, thus, the quality of the training data we are feeding to the HTR model. Secondly, if the database happens to be of good quality, we can think of ways to incorporate these data into our research, e.g., to provide us a lot of free extra training data. In our evaluation, we assume a value to be correct if both entries (of our annotations and the database) are similar. If not, we manually inspect the certificate and try to find which of the two is correct.

In the next sections, we start by giving some general remarks about the database and our annotations (Section 6.2). Then, we go into depth and give a thorough analysis of the two entities certificate_number and last_name (Section 6.3 & 6.4). The comparisons of those and

the other entities are then summarised in Section 6.5.

6.2 General remarks

Matching the 100 scan names with the three sheets in the Excel file, we found 113 matches with a left join on the 100 scans. 13 of those are duplicates (2 certificates have 3 entries, 9 have 2 entries), and 7 scans did not retrieve a database entry. This entry can be non-existing or the scan name could not be found (due to a spelling error). This shows us that the database is not as complete as initially thought for the years outside our training data.

The duplicates are due to 1) both last name of the father and the mother having a separate entry, or 2) a different spelling is used 'Le Clerc' vs 'Clerc, le'. So, this shows already two design choices made by the people who made the database. They preferred multiple rows for one person if their name could be written in different ways.

Now, two general errors we saw when looking at our annotated data: 1) our annotators made more transcription errors than the database annotators. This is firstly visible in certificate number where we notice bias that the annotators are not adjusting the machine output while annotating. These could have been avoided when looking at the certificate more thoroughly. This also holds for names, though, they are harder to transcribe. 2) Our annotators forgot to label or mislabelled. This common in data annotation. We did not put an actual number on it, but saw it every once in a while, at a rate that did not surprise us.

What we would like to point out about the database is its duplicate entries, but also the problem that a majority of database entries does not overlap with certificate text. This is not necessarily a problem as the entity information is correct. However, we cannot use the database entry to find the position in the certificate text and use it for HTR training. An example is the switching of names. Sometimes in certificates, names are written like 'last_name, first_names'. The database entry switched this to the correct order, but makes it difficult for us to place it back in certificate texts.

6.3 Certificate number

When comparing the 93 samples from our set and that of the database, there were 69 entries identical. When examining the remaining 24, we discover the following:

- **5 entities are correct but in different format**. The number is in both sets the same, however, the database set writes '5212' or '5-212' instead of the correct '5/212'.
- **2 entities are misread in our set**. We argue that it is very difficult to distinguish the 1 from 7 and 3 from 5 in handwriting. We know the database is correct (and our annotators are incorrect) because the certificate number is also in the scan name, and the images in the database are in order. Our two annotators did not have this additional information.
- **6 entities where the two annotators are completely wrong**. If we inspect the image, we clearly see the handwriting characters and conclude the database is correct. We suspect that the two annotators were oblivious to these mistakes when correcting the predictions from the HTR model. So, the wrong digits arise from biases in our HTR model and the annotators missed to revision them. Another reason can be that corrections are forgotten to be saved or incorrectly saved in the Transkribus tool. We

tried to reduce synchronising issues by dividing the certificates over the two annotators. It may have happened that two users were on the same page simultaneously resulting in saving old incorrect pages. We checked user activity and found two users rarely active at the same moment. If so, they saved different scans, not close to each other, so they were not working on same pages simultaneously. Therefore, we find the first reason for the mistakes (missed revisions by the annotators) to be more likely.

- **2 entities the two annotators forgot to label**. This is an indication that mistakes can be made when labelling. We suspect forgotten labels in other entities as well. For some entities this is easy to check (some entities should be in every certificate). Other entities from the additional information (about family, sex, place of birth) are more difficult to immediately check.
- **1 entity in both sets incorrect**. One certificate number is 7/321 which is entered as 321 in the database and labelled as 7/324 by the annotators.
- **1 entity in the database is incorrect**. The certificate number 6/496 is entered only as 6. This is in contradiction with the previous bullet point where the database only had 321. There does not seem to be agreement in how to enter the certificate number here. For our annotators, we agreed to enter what is visually seen in the certificate image, so both numbers before and after '/'.
- **7 entities in the database have an unknown addition**. The entities are written as 4/B/3 while the certificate number is only 3. We do not see in the images where the B comes from. This is important to keep in mind when using the database as training data in future research.

So, for both datasets the quality is not perfect. The two biggest mistakes are ignorance by the two annotators and a weird 'B' addition in many entries in the database. The possible bias that the two annotators failed to spot mistakes from the well performing HTR-model calls for one more run over the data. Although our Training_set might not suffer from this bias, it is still worth going over it as we have only about 100 samples. Because this is so little, errors have a larger effect on the performance of the HTR-model.

6.4 Last name

Last names cover a wide range of words and variations, so examining the characters thoroughly is needed. Comparing our set and the database yields immediate similarity of 62 samples (66,67%). The differences are the following:

- **11 entities with different spellings**. The database and our set are very similar but differ in a couple of characters. From our observation, we conclude that the database seems to be more correct. Though, it is hard to compare as some characters are very similar to one another.
- **14 entities in the database with the last name of the mother**. In these certificates, the names written from the deceased person are all classified as first names. The last name is derived from the mother. Our annotators took the last word of the name as last name and the ones in front as first name(s). Which approach is correct is not that straightforward. Back then, name shifting happened a lot when people were only known by their first names and documented without last name.

- **2 entities of stillborns**. Our annotators did not label any last name here, the database used the mother's last name.
- **3 entities with other separation in first and last name**. If the full name consists of one word, e.g., 'Andriesen', it can be ambiguous whether this is the first name or last name. This also holds for long full names, such as 'Esther van Binjamin de Marchena' where the last name might start at 'van' or 'de'.
- **1 entity in the database is incorrect**. This entry uses the last name of the informant instead of the deceased person. We did not find a reason why this would have been done.

In the last couple of years, names can have a different ordering than the literal text from the certificate. So, in the certificate text is written 'Rosalia, Maria' which has database entry 'Maria Rosalia'. Our most important observation is that the database has better transcription quality than our annotators. In the next section, we will analyse this more thoroughly. Lastly, we observed the database making a different split in first name(s) and last name. For some certificates, they assumed the name of the deceased were their first name(s), and the last name should be taken from the family written next. In our Future work (Chapter 9), we will discuss this phenomenon in more detail, as this split is very difficult to make (to machines, but even to humans).

6.5 Analysis summary

For each entity, we briefly note down important findings. Access to our full comparison (each entity, 100 comparisons) can be given upon request in the form of an xlsx file.

- **Certificate number**. We notice bias in the annotators not adjusting the machine output, so having more transcription errors which could have been avoided looking at the certificate more thoroughly. For the database, not all entries align, because of weird additions that are not written in the certificate text.
- **Certificate district**. Not all certificates have a district explicitly written down in the certificate text. The database does have complete information, because the scan name contains the district. We can use the information from the database.
- **Certificate date**. Both our annotators and the database are correct all the time. The database only wrote down the year, though, we instructed our annotators to label the full date. These dates are written out but can easily be parsed to dd/mm/yyyy format.
- **Date of death**. Only some minor errors like explained in Section 6.2. We are perfectly able to parse to dd/mm/yyyy format, however, we cannot parse it back to the fully written dates as multiple formats exist, e.g., 'acht', 'achtste' or 'achtsten'.
- **Date of birth**. What is different is that the database did not write down if the date of birth was unknown if it was specified in the certificate text. There is also again parsing needed from fully written dates to dd/mm/yyyy format. This results in the majority of date of births being the same between the two datasets.
- **Time of death**. Both the annotators and the database were correct all the time. The time is not directly parsable to its original text as the database has entries like '09:00:00' but this can have original text like 'negen ure te morgens' or 'morgens te negen ure'. This only makes parsing in one direction possible (from our certificate label to the database entry).

- **Place of birth**. Often the certificate contains 'geboren te Curaçao'. Our annotators labelled this correctly, though, the database only did this for the second half of the data. All 8 special cases with a place of birth other than Curaçao were correct for both our data and the database.
- **Place of death**. We found this entity to be very uninteresting for this set of data, because it all pointed towards 'Curaçao' or one of its districts. The database is empty for the first half of years. Its second half only consists of 'Curaçao'. So, the database is not useful in future for this particular entity.
- **Age**. This entity is troublesome, because no entry aligns directly with the certificate text (all written digits are converted to actual digits). If we would parse these back, still only half of the entries align, as there is still a lot of variety in noting down. For example, 'naar gissing zestig jaren' can have a database entry looking like '60 jaar naar gissing', or '∼60 jaar', but also many punctuation changed or the word 'en' between number of months and days is replaced by a comma.
- **Age informants**. First half of the years do not have the ages of informants. In the second half, they are all correct and easily parsable, because only the year is given, and no months or days. So, '66' or '66 jaar' can be parsed back to the original certificate text 'zes en zestig jaren'.
- **Sex**. About half of them have a label, these can all be correctly parsed to male or female. A benefit of the database is that there is a value in all entries. We assume they have guessed the sex based of the name of the deceased.
- **Marital status**. The majority is okay, we did find our annotators to be more complete, because there are some entries missing in the database.
- **Deceased father/mother**. These two entities only occur very little. We found our annotators to be more complete, because they found 22 times (out of 200 cases) whether a father or mother had deceased; the database only had 6 of these written down.
- **Professions**. These are generally okay, there are no alignment issues. Sometimes, a trancription error is made by the annotators, e.g., 'Kapper' should have been 'Sjapper'.
- **Names**. Again some minor general errors, but it is mostly the transcription that differs one to a couple characters. Looking at the certificate more closely with both options in mind, we concluded that the database is more often correct. See Table 6.1 for the comparison. For some database entries, there is no full alignment with the certificate text anymore, i.e., 'Aliuve, Teresa' has entry 'Teresa Aliuve'. This is correct, but can cause troubles if we were to insert database entries back in certificate texts. Another major problem was already written in Section 6.4 about the last name of the mother. Lastly, there is the problem of aliases in marginalia or behind names in certificate texts that are in some way written in the database. Often, this is done by adding '(a)' and the alias behind the real name, but this is no consistent format and does not always align with the actual certificate text.

6.6 Conclusions

To conclude, we suggest to do some data cleaning before releasing the database to the public. We need to get a better visual of which scans have complete information, which have partial information, and which still need to be transcribed, so are a good candidate for our HTR+RegEx pipeline.

Name	**Last name**	**First name**	**Informant_1**	**Informant_2**	**Mother**	**Father**	**Partner**
Total	91	89	93	85	49	22	11
Our annotators wrong	5	10	24	31	9	6	5
The database wrong	0	2	10	3	2	2	0

Table 6.1: Transcription quality of our Sample_known compared to the values in the database, after manually examining thoroughly once more by one of the annotators. This table does not show other types of errors or when both sets were wrong.

Also, not all entities can easily be used to create more training data for our project. The quality of the database is very good, but not fitted for our HTR as the database entries do not 100% align with the literal certificate text. Some parsing, e.g., for ages, dates, times, is possible, but only from its written-out form to digital format. That is because there are multiple written-out formats. So, the database can be used as evaluation data, not as training data.

The database has some interpretation from humans that cannot easily be solved on machine level. For example, regular expressions fall short when a sex should be derived from the person's name. The database could serve as knowledge base, though, showing which first name and sex occur often together. Another example are people with aliases. These could be explained in the marginalia, which human transcribers can read and add the alias to the person's name. For a machine, this is more difficult to extract.

Finally, this analysis calls for one more look over our own data, especially for the spelling of names. We saw the database having higher transcription quality, so first of all, Sample_known should get some revisions. We think it useful to look at the names in the other sets (Training_data and Sample_regex) too and try to improve the names.

Chapter 7

Entity Extraction using RegExes

This chapter analyses the use of regular expressions for some interesting entities in our data. As experiment, regular expressions are created based on the training data. We assess the complexity of these expressions against their precision and recall. We find whether the expressions are robust enough to find matches in certificates from both similar (Sample_regex - ground truth) as different years (Sample_known) [RQ-4.1]. We also evaluate the performance on imperfect HTR output (Sample_regex - model output). These data show how our expressions would perform in a final pipeline when no ground truth is availale [RQ-4.2].

7.1 Method

For this experiment, we decided to only look at the training data while creating the regular expressions, to test the variety between these 100 samples and the other 200 samples. The Training_set against the Sample_known in particular shows how much variety there is in other years. We suspect the majority of entities follow a similar structure, so there will also be matches in data we did not look at.

The false positives are analysed to distinguish mistakes that can be solved by regular expressions, e.g., mismatches, and mistakes that would remain undetected in real setting, e.g., HTR errors. We evaluated what performs sufficiently and what needs further improvement. After the experiment, the regular expressions can be improved by adding the new patterns found in the other two datasets (Sample_known and Sample_regex). New false positives will occur once scaling up to the full dataset of certificates; we expect a similar rate of this, because of the representativeness of the 300 certificates (it was randomly selected from nearly all years). We will see that the HTR for some entities significantly needs to be improved. An advantage of this method is that the RegExes can iteratively be improved. One can examine the set of certificates without a match to find missed matches.

The entities we decided to analyse thoroughly capture the variety between entities and how they should be detected. Some follow a certain format, e.g., dates, so they have a limited range of possible words. Others, such as names, have a huge range of possibilities. We also investigate entities, such as sex, that are not explicitly written down, but can be deduced from other words (e.g., 'dochter/zoon van') or from the name itself. The latter is not possible with regular expressions, as knowledge about male and female names is needed (which can be even ambiguous and hard to determine for humans). The same we will acknowledge in the split of first and last names, as 1) more knowledge about Caribbean names from the 19th-20th

century is needed, and 2) regular expressions can only split words on a syntactical level. To summarise, we divide the entities into the pools described below.

- **'simple' entities: certificate number, certificate district.**
  Short entities, very useful for part of the HTR evaluation as ground truth is available for *all* certificates (these are written in the names of the files). The entities have a limited range of possibilities: digits and district names. Interestingly, not as 'simple' to the HTR model as many machine errors are made.
- **'format' entities: certificate date, date of death, date of birth, time of death, place of birth, place of death.**
  Follow a limited pattern of options. With a parsing function, one can transform the written-out digits and day parts to numerical date and time.
- **'highly variable' entities: name, age, profession of all persons (deceased, informants, father, mother, partner).**
  These entities are difficult to extract, not only because some of them have a lot of variety (e.g., names and their variable length), also the surrounding words have a lot of variety (e.g., profession can start right after age, so the split is difficult to make for a RegEx). Because the name should desirably be split into first name(s) and last name for the deceased person, this one is chosen to analyse thoroughly in this experiment. We also view the name of the deceased as the most important entity to extract.
- **'derivative' entities: sex, marital status, are father and mother deceased.**
  These entities have some indicative words throughout the certificate to determine its value. It is important to search for sex and marital status only in the information about the deceased (and for father/mother deceased in the father/mother section). These entities occur in a minority of the certificates. For sex, this means a human is needed to manually evaluate the names and determine their sex. This is where RegExes fall short.

The remainder of this chapter will describe entities: certificate number, date of death, name and sex of the deceased (one from each category). The precision and recall of all other entities are presented in Section 7.8. These entities got less attention, so improvements can definitely be made. We did want to build regular expressions for all entities, to find out if any major problems would occur.

The regular expressions and functions are built in Python. Our scripts are available in our GitHub repository[1]. Figure 7.1 shows an overview of the flow. We start by pre-processing the certificate texts, i.e., removing the newlines so one continuous text is obtained. When building the RegExes, we started by creating a simple RegEx that matches one or multiple certificates from the training data. Then, we inspected the certificates without any matches and iteratively extended our RegEx till we were satisfied with the number of matches. We based this on our time spent compared to the potential increase in matches and decrease in false positives if we were to continue. We also assessed how easy we thought it was to make the improvements. For some entities, we conclude (Section 7.8) RegEx-patterns became too complex and we failed to extract these entities in a time limit of 8 hours.

[1] `https://github.com/LisaHoek/HTR-RegEx`

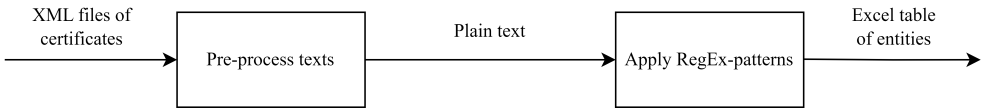


Figure 7.1: Flow of the second part of our HTR+RegEx model.

7.2 Certificate number

This entity is least complex, so easy to start with and serves as quick indication of the performance. It is always at the top of the certificate (in plain text after the marginalia). All 300 samples have a certificate number. Retrieval is done by matching on 'nr' or 'no' and we allow for a dot and space before the digits are matched. RegEx 1 shows the pattern for the training data. Green is used to indicate the surrounding words that serve as boundary between the certificate text and entity that should be retrieved. Orange shows a named group of the entity to be able to put it individually in a database.

For the training data, we find this RegEx pattern already matches all 100 correctly.[2] Table 7.1 shows the number of matches for the other datasets too. We find a new pattern where the certificate number consists of two parts separated by '/'. This can be added in a second version. For Sample_regex, we find all 100 certificate numbers correctly on the ground truth annotated data. However, this greatly reduces once we retrieve the entities from the imperfect HTR data. Then, there are 41 certificate numbers wrongly transcribed by the HTR model. The majority of these are visually close numbers being mixed up, such as '3' and '5'. The model also sometimes misses the number '1'. It is impossible for the regular expressions to recover from these mistakes. That is why improvements in the HTR model are needed to lower these false positives.

```
n[ro]\.?\s?(?P<certificate_number>\d{1,4})
```

RegEx 1: Certificate number

Dataset (#entities)	**Fuzzy (LD)**	**#matches**	**#correct**	**#FP**	**Explanation FPs**
Training_set (100)	0	100	100	0	
Sample_known (100)	0	100	88	12	(12x) non-digit inside certificate number, e.g., 'No. 7/324'
Sample_regex (100) *Ground Truth*	0	100	100	0	
Sample_regex (100) *Model output*	0	96	55	41	(41x) wrong HTR output (other numbers)
	1	100	55	45	(3x) no digits in HTR, (1x) 'no' or 'nr' missing
	3	100	55	45	

Table 7.1: RegEx results for certificate number

[2]We found 3 certificates the annotators forgot to label and added them before showing the results.

7.3 Date of death

The second entity we investigate is the date of death. We chose this entity because all 'normal' certificates should have this information. This entity is similar in format as certificate_date and date_of_birth as the dates are all fully written out. We wrote an additional function that parses the dates once retrieved into d(d)-m(m)-yyyy format which is similar to the existing format in the database and makes comparing easily. We continue improving our regular expression until 96 matches are found in the 100 training data. After a manual check we see that 93 of those are correctly retrieved and parsed. The regular expression (simplified) can now be expressed as RegEx 2. In Appendix C, one can find code for the named groups day, month and year.

op (den)? <day> <month> (des jaars <year> | dezes jaars |) (ten? | des)

RegEx 2: Date of death

The colour blue is used to better highlight a group in the expression with options for the year. If the certificate says 'dezes jaars' or the year is deliberately left out, we retrieve the year from the certificate date. If we now execute this regular expression on our 100 from 'sample_known', we find 87 matches of which 86 are correct. In hindsight, if we allowed ourselves to look further than the training set, we could have improved our expression (+3 matches) by making the month optional too. This can easily be done in a next version of the RegExes. Looking at 'Sample_regex' ground truth data, we find 96 matches of which 93 are correct. When turning to the imperfect HTR data, there are still 74 dates fully correct and only one false positive is obtained. This shows there is potential for automatic extraction of dates of death, and, when no match is found, manual transcription.

Summarising the false positives, we see that 1) some can be avoided when expanding the regular expression (adding '1800' and 'één'), 2) some other dates are matched such as the date_of_birth, for which a solution can be to use the place_of_death as prefix as this pattern does not occur in front of other dates, 3) there are difficult certificates because they contain strikethrough; we do not have the best solution for this yet, as our HTR cannot detect strikethrough that well, and 4) there are mismatches because the HTR model outputs wrong words, which will become less once we have better HTR models trained on more data.

7.4 Fuzzy matching

To investigate whether fuzzy matching can help our Entity Extraction, we have results of the same datasets but allowing for 1 or 3 Levenshtein changes. Table 7.2 shows these results. It shows that fuzzy matching helps obtain more matches in all cases. For the training data, we see this is due to small spelling mistakes the annotators made, e.g., twintigstten. However, the sample_regex with imperfect data is the data that is most representative for data in a real case setting, so this should be inspected more thoroughly:

If we look at the 11 false positives, we see 2 difficult cases that were already incorrect in the ground truth Sample_regex_0, but 9 dates that come from bad HTR output which then matches another day, month or year. If we use fuzzy matching LD=3 instead of LD=0 on our imperfect data, 15 extra are correct. One of those could have been solved by adding 'één' as

Dataset (#entities)	Fuzzy (LD)	#matches	#correct	#FP	Explanation FPs
Training_set (99)	0	96	92+1	4	(1x) date of birth, (3x) strike-through (dezes jaars → previous year)
	1	99	95+1	4	
	3	100	95+0	5	(1x) certificate date when no date of death
Sample_known (100)	0	87	86	1	(1x) date of birth
	1	92	91	1	
	3	100	96	4	(1x) certificate date, (2x) '1800' is no valid option yet
Sample_regex (100) *Ground Truth*	0	96	93	3	(1x) strike-through (month), (2x) 'één' is no valid option yet
	1	99	96	3	
	3	100	97	3	
Sample_regex (100) *Model output*	0	75	74	1	(1x) strike-through (month)
	1	82	79	3	(2x) wrong HTR
	3	100	89	11	(1x) één is no valid option yet, (7x) wrong HTR

Table 7.2: RegEx results for date of death

option to the RegExes, so 14 samples were helped by the fuzzy matching LD=3. So, using fuzzy matching (LD=3) hurts 9 samples, while 14 were helped. We argue this is not enough for our use-case, as too many FPs would go undetected in the final pipeline. The same holds for fuzzy matching (LD=1) in which 3 extra matches are correct, but 2 FPs are obtained. This indicates that fuzzy matching is not a solution to the mistakes in the HTR output.

Another reason we find high precision more important than recall is because not finding a match can be a good indication that the text was difficult for the model to transcribe or that there is something else going on in the certificate (that it deviates from 'normal' certificates, e.g., strike-through). This can serve as a warning or check to put the certificate aside and manually transcribe.

Fuzzy matching might still have a use case for names and professions, when we match them against our available lists of names and professions. This is something we would have to test. But we do not believe so, as this makes matching new names or new variations impossible. This can be a check, if the name is not in the list, let a human transcribe extra. But again, there are already biases that the HTR model might change names to often occurring ones.

The entity 'age' is similar to dates as it contains written-out digits. So, we suspect fuzzy matching performs similar in these cases too, creating false positives because of wrong HTR output. Fuzzy matching might become interesting to look into again once we have better HTR models. This might balance the performance gain into retaining precision and increasing recall.

7.5 Name of deceased

The next entities we create a RegEx pattern for are the last name and first name(s) of the deceased. We decided to do the parsing of first and last name in a separate function and find this split is not as straightforward as initially thought. We encountered in our training set the following splits and made the following assumptions:

ASM-1	Last Name(s) [comma] First Name(s)	e.g., de Lagos, Rafael
ASM-2	First Name(s) Last Name	e.g., Poulina Wilhelmina Wall
ASM-3	First Name(s) [dot] Last Name [dot] [comma]	e.g., Anna. Johanna. Manzana.,

ASM-4 First Name(s) [lowercase affix] Last Name e.g., Louis Napoleon de la Nooy
ASM-5 First Name(s) e.g., Pauwlina

We did not encounter any samples writing only a last name (stillborns follow a different format), however, this does not prove those samples do not exist. What did occur are certificates in which only the first names are written down (ASM-5). We suspect the informants did not know the last name of the deceased in these cases, or, the official meant the last name should be taken from the father or mother.

Once multiple first names are written down without a last name, the parsing becomes ambiguous. Determining whether the last part is the last name or a middle name is difficult and even unclear to the historians in the HDSC. Many last names originate from first names, e.g., 'Anna Martina'. The historians are aware of name changes taking place, in particular in Curaçao. People took first names as last name once the last name was not known anymore. This was not uncommon. In the case of 'Anna Martina', one could conclude 'Martina' is the last name if the father shares the same name (or mother if father is not given). However, this becomes ambiguous once 'Anna Martina' is the daughter of 'Christina Martes' (and no father given).

Future research is needed to lead to better rules how to split, e.g., looking at parents' names or look at lists of possible first names vs last name, maybe there are other patterns hidden in the certificate. For now, we implement ASM-1/5, and apply ASM-2 over ASM-5 if there are multiple names (so, only ASM-5 holds when one name is given).

Unfortunately, when we look beyond the training data, we also encountered an example in which the last name consists of multiple words starting with capitals, e.g., 'Johanna Geertruida Epsteen Boom'. We know the last two words are the last name, as the father is given as 'Lourens Epsteen Boom'. This proves our assumptions to be incomplete. Since this cannot be distinguished on a syntactical level (punctuation, upper/lowercase), we leave it for now.

(overleden is | is overleden) <name>
(in den ouderdom | oud | (zonder | van) beroep | wonende)

RegEx 3: Name of deceased

Dataset (#entities) (L/F)	**Fuzzy (LD)**	**#Matches (L/F)**	**#Correct (L/F)**	**#FPs (L/F)**	**Explanation FPs**
Training_set (82/83)	0	81/82	79/80	2/2	(1x/1x) extra suffix (alias), (1x/0x) other last name in marginalia (0x/1x) wrong parsing due to space in first name, i.e., 'Pauw lina'
Sample_known (94/97)	0	94/97	88/91	6/6	(1x/1x) extra suffix (gouverneur), (2x/2x) extra suffix (alias) (1x/1x) two capitals last name: Sint Jago, (2x/2x) other suffix: ongehuwd/kapper
Sample_regex (93/93) *Ground Truth*	0	93/92	78/79	15/13	(10x/10x) wrong parsing, (2x/2x) alias in marginalia (1x/1x) name in other position, (1x/0x) person 'onbekend' (1x/0x) extra word in match
Sample_regex (93) *Model output*	0	92/90	32/45	60/45	(1x/1x) wrong parsing due to wrong HTR (44x/31x) wrong HTR output

Table 7.3: RegEx results for name of deceased person. It shows the results for 'Last name' on the left and 'First name(s)' on the right.

RegEx 3 shows the pattern with the surrounding words we search for based on our training data. We revisioned one sample in the training data to 'Bernard. Koko.,' which was incorrectly transcribed as 'Bernard, Koko' by the annotators and thus incorrectly parsed under our assumptions. We expect more of these parsing errors to occur when our assumptions do not follow the labelling the two annotators applied. Table 7.3 shows the results.

In Sample_regex, we do observe 10 false positives (wrong parsing) that do not follow our assumptions. It is still an open research question how to parse the name. If we were to ignore those for now and inspect the 10 names, we still observe 8 of them containing spelling mistakes, which would make the #FPs on the ground truth go down to (5/3), which is fairly okay, but the #FPs on the model output to (58/39) which is still quite high. Most false positives are due to wrong HTR output; these names have 1 to a couple characters written differently, e.g., 'Anelia' instead of 'Amelia' or 'Ersten' instead of 'Epsteen'. These (44/31) wrong HTR output is in line with findings in Chapter 5 in which we observed WER = 9.43% on the full text, but once filtered to the line containing the name of the deceased, the errors increased to WER = 39.48%.

Of all those HTR mistakes, it is possible that some machine output is correct, but wrongly annotated by the annotators. One thing we observed is that the annotators were unsystematic in their punctuation, i.e., they often forgot the dot '.' between names. These mistakes are understandable, as they were not made aware that these dots would have big influence on the end results (we also did not know this at the time). What it means is that some certificates following ASM-2 should have been ASM-3 (which is not harmful), but once ASM-1 should have been ASM-3, this changes how the split is made. In the annotated data, we also found the following patterns:

ASM-6?	First names [comma] Last names	e.g., Dionisio, Borja
ASM-7?	Last name First Name	e.g., Domingo Cristoffel

As these names are ambiguous and family members do not give insight into the last name (e.g., 'Domingo Christoffel' is the son of 'Anna Martina'), it remains uncertain how these names should be split. Following our assumptions ASM-1/5, it might be more likely that 1) the names should be switched, so ASM-6 aligns to ASM-1, and 2) ASM-7 could be two first names i.e., ASM-5.

A third false positive are names in which extra information is written down. This can be an extra suffix behind the name (e.g., the title of the gouverneur) or it can be an alias. This alias can be written both behind the name or in the marginalia. In an ideal case, we would like to capture these aliases or additional suffixes. So, a follow up research question could be how to handle the information in the marginalia. This is all written text, so it does not follow a standard pattern. This makes it difficult to handle automatically.

Next, we looked at the certificates without both a first and last name labelled by the annotators. Table 7.4 shows these results. The false positives occur in two cases. 1) When the certificate describes a stillborn baby; the printed surrounding words we search for (i.e., 'overleden', 'oud', etc) still appear in the certificate if the HTR output does not detect strikethrough. It is still an open question how to handle strikethrough in certificates; Transkribus performs poorly to recognise it. Thus, this thesis uses the HTR model that is trained without the tag on strikethrough words, which means our HTR output does not label any strikethrough. This has the downside that the printed strikethrough words still occur throughout the certificate about the stillborn. A solution to reduce these false positives is either to account for this in

Dataset (#entities)	Fuzzy (LD)	#matches	#correct	#FP	Explanation FPs
Training_set (17)	0	5	12	5	(3x) 'overleden' in marginalia, (2x) stillborn
Sample_known (3)	0	0	3	0	
Sample_regex (7) *Ground Truth*	0	3	4	3	(2x) 'overleden' in marginalia, (1x) stillborn
Sample_regex (7) *Model Output*	0	7	0	7	(4x) stillborn

Table 7.4: RegEx results for the last name and first names on the certificates without any name given.

the regular expression, or, to filter the stillborn certificates out sooner. 2) The second false positives are happening when the surrounding words are also written in the marginalia. This is often done to approve the strikethrough. It leads the RegEx to search for a match in the marginalia. These can probably be solved by an improved RegEx pattern.

To conclude, besides improvements on the RegEx pattern, the rules on how to split also need more attention. The above results seem promising, but we are not certain the annotators were correct in their split of first names and last name. Although our current set of assumptions seem to perform mostly in line with our two annotators, we argue another difficulty might arise when used in a real case setting with imperfect HTR. We must evaluate how well the HTR transcribes the punctuation in the names as our parsing depends on it. So, we suggest once a better HTR model is trained, an additional evaluation should look at this punctuation. In our current model, the HTR performs poorly, due to the fact that it was fed incorrect training data. We would like to find out whether feeding more and correct data to a new HTR model is enough to remove wrong parsing due to wrong HTR output. If not, our model will not be suited to automatically detect the split between first name(s) and last name.

7.6 Sex of deceased

Lastly, the sex of the deceased person is an entity that is not explicitly given in the certificates, so interesting to analyse. Annotators from the database guessed the sex based on first name(s) of the person if the sex cannot be deduced from other words in the certificate. We decided to keep the regular expression simple, so we search for words like 'zoon' (son), 'dochter' (daughter) and more.

(overleden | kind) .+? <sex>

RegEx 4: Sex of deceased

RegEx 4 visualises the pattern simplified; the pattern searches for 'overleden' or 'kind' to go to the middle of the certificate (and not match anything in the informants information) and tries to match on the named group <sex> (Appendix C.4). Table 7.5 shows the results. To improve the accuracy of our data, we fixed a total of 8 labels that the two annotators

missed while labelling the 300 samples.[3]. We find 89 certificates in the training data with words that indicate the sex, those are all correctly matched. For 'sample_known', we fail to find one match as this wrote 'manlijk' instead of 'mannelijk'. This can be added to the named group as an option. For 'sample_regex', there are two false positives because of the sentence 'zoon van de overledene' in the informants information. This matches our word 'overleden', so the sex is matched of the second informant and not of the deceased person. This can be improved by adding a word boundary '\b' behind 'overleden'. It is also an indication that we might need to find better cut-offs between information of the informants, deceased and family in a second version of the RegExes.

Once we apply the pattern on the imperfect HTR, there are 8 less matches as our indicative words contain spelling errors, e.g., 'ochter', 'dochte', 'Loon', etc. Although fuzzy matching could help find these matches, we do not apply it here, as we expect more false positives to arise too. A clear example is already the similarity between 'echtgenoot' and 'echtgenoote' which makes fuzzy matching (LD=1) already ambiguous.

Dataset (#entities/#noLabel)	**Fuzzy (LD)**	**#matches**	**#correct**	**#FP**	**Explanation FPs**
Training_set (89/11)	0	89/7	89/4	0/7	(1x) difficult certificate, (6x) sex of parents (stillborns)
Sample_known (62/38)	0	61/0	61/38	0/0	
Sample_regex (94/6)	0	94/2	92/4	2/2	(2x) sex of second informant
Ground Truth					(2x) sex of parents (stillborns)
Sample_regex (94/6)	0	86/2	84/4	2/2	
Model output					

Table 7.5: RegEx results for the sex of the deceased person. It also shows the certificates without a label, so these should return no matches.

Just like the previous entity, we looked at certificates without the label 'sex'. We do not want to find any matches in these sets, as these would be false positives. Table 7.5 also shows these results. We observe FP's when the certificate describes stillborns; when no sex is given, the pattern still matches on the sex of the mother or father. Two possible solutions can be to filter all stillborn certificates earlier in the process and handle them differently (they have lots of strikethrough too), or, improve our RegEx pattern. As already mentioned, we would like to avoid matches of sex in the informants section (before the deceased information), but now also in the family member section (behind the deceased information).

We expect this regular expression can be further improved to reduce false positives, which then becomes very suited on bigger scale. We suggest other methods than regular expressions to retrieve the sex for the remaining certificates that did not find a match, e.g., using databases that link the first names to sex, or, using other linking data such as birth and marriage certificates.

[3]These 8 samples were found after finding correct matches in our samples that were indicated to not have the label 'sex'.

7.7 False Positive Analysis

From our thorough analysis of the entities above, we summarise the false positives as follow:

1) **Stillborns**. These certificates need special attention as these are written inside the printed structure of the certificates, but often with strikethrough to write custom sentences. Since we have seen that these custom sentences follow some structure themselves, it should be well possible to extract all stillborn certificates from the full dataset. We suggest to handle them separately and split the pipeline to reduce errors in the Entity Extraction of 'standard' certificates.
2) **Strikethrough**. Transkribus is unable to sufficiently learn strikethrough. Many of the certificates containing strikethrough are certificates of stillborns. The solution provided for this alleviates the number of strikethrough certificates. Also, many strikethrough relate to marginalia text. Often, the correct substitute can be found here. A strict compromise could be to extract all certificates with marginalia text, and submit those for manual transcription. However, based of our training data, this would be more than half of the certificates. We believe a smarter solution should be possible to handle strikethrough in combination with marginalia text, e.g., detecting which marginalia are not useful and do not need manual transcription. A solution could also be found for strikethrough in itself. Transkribus is continuously improving their models, so this might work better in the future.
3) **Wrong HTR output**. We observed that wrong HTR output leads to a lower recall, because some sentences do not match our pattern with spelling mistakes. Fuzzy matching does not provide a desirable solution here, as it also increases the number of false positives which would go undetected in a production setting. In particular names have a much higher CER and WER. A solution to explore in future work is whether we could add a name list to the language model.
4) **Surrounding words not exclusive**. Text in the marginalia gives opportunity that long false matches are found when a starting word from a RegEx resembles one in the marginalia, e.g., 'overleden'. It is important to write RegExes in such a way that this is avoided. This is well possible, e.g., already by adding the check to only match after 'heden den' the start of the certificate text. So, a solution should be found in the RegEx patterns to make it exclusive enough that it does not find matches in wrong places (e.g., date of birth instead of date of death).
5) **Entity-specific FP's**. These false positives can only be filtered out after manual evaluation. This chapter conducted the first round; FP's can at least be reduced for the 300 data we analysed. We suggest such an evaluation on FP's is useful again once the problems 1-4 are improved, as the majority of the errors lay here.

Problem 4 and 5 are due to insufficient RegEx patterns and can easily be improved by altering the RegExes. Recall can also increase by revisioning the RegExes. We suggest to first solve or improve problems 1-3 before doing another round of evaluation. In addition to these, it is currently an open problem how to automatically split the names.

7.8 Overview results

Table 7.6 shows the precision and recall for the entities analysed in this experiment, but also for the other entities in the certificates. The regular expressions are built in a short amount of

time, to show how easily a majority of matches can be obtained (or to conclude it is unfeasible). Note that the described difficulty is our subjective view on it. This is a first version of the RegEx patterns, so these should definitely be improved at a later stage. Once the main false positives are dealt with, it becomes useful to evaluate the same expressions again to see the improved results.

	Training_set		*Sample_known*		*Sample_regex GT*		*Sample_regex MO*		
Entities	P	R	P	R	P	R	P	R	*Difficulty*
Certificate number	1.00	1.00	0.88	0.88	1.00	1.00	0.57	0.55	Easy
Certificate district	Not executed due to file_name having this information								(Easy)
Date of death	0.96	0.93	0.99	0.86	0.97	0.93	0.99	0.74	Okay
Certificate date	1.00	0.98	1.00	0.93	1.00	0.93	0.99	0.83	Okay
Date of birth	1.00	0.86	1.00	0.23	1.00	0.70	0.97	0.47	Okay
Time of death	1.00	0.94	1.00	0.97	1.00	0.96	0.99	0.78	Okay
Place of birth	0.96	0.95	0.89	0.88	0.84	0.79	0.76	0.70	Easy
Place of death	1.00	0.99	1.00	0.44	1.00	0.98	0.90	0.89	Easy
Last name	0.92	0.96	0.94	0.94	0.81	0.84	0.32	0.34	Hard
First name	0.92	0.96	0.94	0.94	0.83	0.85	0.46	0.48	Hard
Age	0.76	0.76	0.56	0.25	0.65	0.64	0.42	0.41	Hard
Profession	0.96	0.96	0.88	0.75	0.98	0.90	0.81	0.64	Hard
Name informants*	0.94	0.94	0.87	0.40	0.87	0.83	0.27	0.25	Hard
Age informants*	1.00	1.00	1.00	1.00	1.00	0.95	0.92	0.84	Hard
Profession informants*	1.00	1.00	0.98	0.96	1.00	0.95	0.70	0.64	Hard
Name father/mother/partner	Proved to be too difficult to create within 8 hours								Too difficult
Profession father/mother/partner	Proved to be too difficult to create within 8 hours								Too difficult
Sex	0.93	1.00	1.00	0.98	0.96	0.98	0.95	0.89	Easy
Marital status	0.69	0.93	0.96	0.98	0.80	0.88	0.83	0.83	Easy
Deceased father/mother*	0.96	0.95	1.00	0.80	1.00	0.93	1.00	0.75	Okay

Table 7.6: Precision and recall for each entity on the datasets Training_set, Sample_known, and the ground truth and machine output of Sample_regex. Difficulty relates to the time invested in creating the pattern. **Scores of Informant 1 and 2 and the parents are combined.*

For a first version, the recall is already fairly high. The recall does not lower much when comparing Sample_regex (ground truth) to the training data (similar years). Recall is sometimes low when the entity has a new format (e.g., other surrounding words), so it is not matched with the current RegEx. It means that each low recall in Sample_known can be increased by adding new patterns, and we expect these patterns to translate to most of the certificates from the same time range as Sample_known. So, a new set of 100 certificates will likely have a similar recall. For the machine transcribed certificates, recall can be improved by better HTR.

Unfortunately, our experiment did not successfully retrieve information about the father, mother and partner. Unlike the informants, the information about the family is loosely structured. It has many different formats behind the information about the deceased. This makes RegEx matching difficult, because the surrounding words to match on vary a lot. So, it is difficult to find a match, but also to enclose them without any noise. We believe spending more hours on the RegExes can result in a more stable version with some precision and recall similar to other entities. Though, one might not find this worth the effort, so maybe other solutions exist.

The scores in Table 7.6 for the other entities show that we do not expect any (new) big challenges for the entities anymore, as a fairly high score is reached within short amount of time. The false positives mainly consist of the ones described in Section 7.7. These should definitely be improved.

7.9 Conclusions

This chapter categorised the entities into four pools and analysed one from each category thoroughly. For certificate number, we saw many HTR mistakes which we suspect to be biases by the HTR model. Because the entity is also written in the filename of the scan, we can always use this entity as some indicator of the HTR performance, without the availability of ground truth transcription data. When examining the data of death, we found fuzzy matching increasing the number of matches, but also the number of false positives. Because quality is important for the HDSC, we argue fuzzy matching should not be used. When a date of death is missing, it is better to send the certificate for manual verification. Then, we analysed the split between first name(s) and last name and encountered its ambiguity. We did not find a covering answer. Also, there are still many HTR errors in names. About one third of the words are wrong, i.e., some characters off from its ground truth. For the last pool with entities such as sex of the deceased, we found that there are not many false positives, but more false negatives arise when HTR errors in the label occur. It is also not always possible to derive the sex from the certificate. We summarise the false positives into cases of 1) stillborn certificates, 2) entities with strikethrough, 3) wrong HTR output, and 4) insufficient RegExes. The last one can be improved by spending more time on it. This way, precision and recall can be increased for all entities.

Future work will add RegEx-patterns in front of the main RegExes to separate stillborns from normal certificates. There might also be some filter for strikethrough. Furthermore, additional checks might be added to signal for manual transcription. We are satisfied with the first version of the RegExes. We do think this can be a viable method for our model. However, this does need an involved expert who can write RegExes for the certificates, i.e., other certificates such as certificates from Suriname, or birth or marriage certificates need new RegExes. An advantage of RegExes is that they are easily extendable and can be improved over time.

To conclude, this experiment shows that we can use regular expressions on larger scale; we will be able to find a lot of correct matches when RegExes are built on a small sample (size=100), then executed on imperfect HTR data.

Chapter 8

Conclusions

This thesis tried to develop a framework to increase the efficiency of large-scale Entity Extraction in handwritten civil records by investigating the case of semi-structured death certificates from Curaçao 1831-1950. We built an HTR+RegEx model that takes as input scans of the certificates, and returns for each certificate the entities found. We used Transkribus for the HTR component, and created a Python script for the Entity Extraction with RegExes. Based of our research questions, we conclude that:

[RQ-1.1] The 70.000 death certificates of Curaçao mainly consist of good scans. A small minority of the scans have unreadable entities, e.g., a big tear in the certificate. Important characteristics are: 1) Its two-column format, with space for marginalia at the left, and the main certificate text at the right. 2) About two-third of the certificates have marginalia. These can be corrections, approvals of strikethrough, or additional information about a person. 3) There can be other certificates at the sides, and some certificates have lots of whitespace, due to the handwritten words. 4) There is a mix between printed and handwritten text, alternating within sentences. Transkribus is able to recognise both simultaneously, which is important because we needed the full certificate texts with right word ordering to apply the RegExes. 5) Certificates can contain strikethrough, e.g., in the case of stillborns.

[RQ-1.2] When creating the datasets used in this thesis, we kept in mind the characteristics mentioned above. We made sure these characteristics were sufficiently present. Our dataset Sample_known evaluated our annotators' transcription quality, the quality of the death register database, and the RegEx-performance on certificates from different years than the ones from our training set. Sample_regex was created to evaluate RegEx-performance when executed on imperfect HTR data compared to ground truth data.

[RQ-2.1] We trained multiple models in Transkribus. P2PaLA can automatically create two text regions for the marginalia and the main certificate text. LA_Curacao automatically detects the baselines. These models work better than default Transkribus Layout Analysis. When training our HTR model, we conclude that signatures can be treated normally. How to handle strikethrough is still an open question, as Transkribus says they can detect this well, but we observed the opposite.

[RQ-2.2] So, strikethrough does not work well in Transkribus. Also names have a very high error rate (WER=39.48 for baselines with the name of the deceased). We also encountered a bug in P2PaLA: sometimes, it returns a completely wrong, chaotic layout. This can be solved by running P2PaLA again over the certificate individually. We conclude that there is much improvement needed in the HTR-model in Transkribus. In Future work, we discuss several

techniques for this.

[RQ-3] The death register database is incomplete, has duplicates and needs some cleaning before it could be published online. We encountered many interpretation and paraphrasing issues: entities did not necessarily align 100% with the words in the certificate text, or, the last name was taken from the mother. However, this database does have higher quality than our annotators' data. When comparing the two transcriptions along with the certificate image, we concluded that the database was better in transcribing names.

[RQ-4.1] To find how well RegExes can be used as Entity Extraction method, we manually constructed RegEx-patterns based on the training data, and tested its performance on three test sets. Entities from the 'format' and 'derivative' category are rather easy to retrieve as their range of possibilities is limited. Names, age and professions were more difficult to retrieve; we failed to extract the information about the parents with RegExes built in less than 8 hours. All other entities succeeded, but can be improved by spending more time on them. In particular, more possible RegEx-patterns were found in Sample_known, because those certificates consisted of other years. We did not see much performance decrease when testing on ground truth data from Sample_regex. This shows us that RegEx-patterns from Training_set + Sample_known will be well representing the larger set of data. So, these RegExes extend to the majority of the data and are useful for automatic Entity Extraction.

[RQ-4.2] Unfortunately, due to many errors in the HTR-component in Transkribus, much RegEx-performance is lost when executed on imperfect transcription data obtained from the Transkribus output (Sample_regex - machine output). This does not necessarily mean RegExes fail as Entity Extraction method; we should rather focus on improving the HTR-component in Transkribus, so less false negatives or positives arise.

Returning to our main research question 'How can we increase efficiency of large-scale Entity Extraction from handwritten civil records while maintaining quality similar to crowdsourcing in the HDSC?', we demonstrated that we can built a pipeline consisting of Layout Analysis and HTR in Transkribus to obtain the transcriptions of the certificates, and a RegEx-component to extract the entities. Although this pipeline is not fully automatised and does need someone processing the certificates in Transkribus and Python, this model is more time-efficient on large data compared to transcribing each certificate individually. Unfortunately, quality is not great yet, so future work should investigate how this model can be integrated into a crowdsourcing process to better balance quality against efficiency. We do see possibilities how our HTR+RegEx model can be used to speed up the crowdsourcing in the HDSC and describe these in Future work 9.3. Hence, this thesis shows potential using HTR and RegExes in large-scale Entity Extraction of semi-structured handwritten civil records in combination with crowdsourcing.

Chapter 9

Future work

This chapter discusses what we could have done differently or improve upon if more time would have been available. The last sections explain two items more thoroughly. The first advocates for an improvement in the HTR component. Especially names lack performance with many transcription errors. The last section discusses possible ways of integrating this thesis into the current crowdsourcing process. There is no direct answer which method is best, so we give an illustration of some possibilities. This information can be used as starting point for future work in crowdsourcing.

9.1 General improvements

The following list consists of things that could or should be done in future to improve the current version of our HTR+RegEx model:

- **Data exploration**. Once we began our research, some data exploration was already done by colleagues of the HDSC. They already picked 100 training samples with different layouts and handwritings. Looking back, we would have liked to better explore the data to get a good overview of all years. For example, by applying some clustering algorithm on the images, we expect to filter out non-certificates, and find other characteristics. Also, we argue that our method of randomly skimming through certificates was not that systematical.
- **Check for noisy regions**. Due to the bug in layout creation with P2PaLA, described in Section 5.4.3, future work could implement some filter before the HTR model is applied. This is because the HTR model is not free of charge. Applying a filter in front that detects noisy, weird layouts (which lead to incomplete or wrong texts), saves us using credits. Another solution can be to detect noisy layouts by the HTR text it retrieves, however, then the layout still needs manual fixing after which another credit is spent to apply the HTR model again. So, it is more desired to spot invalid regions, and correct those, before detecting the baselines.
- **Improve baselines**. In Section 5.3, we observed the fact that sentences with lots of whitespace get cut in parts, but their ordering is sometimes wrong, because Transkribus displays them from top to bottom. We trained a baseline model to force the creation of one big baseline covering the whitespace, however, this did not work 100% of the time. An additional solution could be to automatically adjust the ordering in the XML

document: if baseline coordinates only differ a couple pixels in height, read them from left to right instead of top to bottom.

- **Stillborns**. In Chapter 7, we observed that certificates of stillborns follow a different text pattern than normal certificates. This created false positives. It eases and improves our RegEx method extracting the entities when stillborns are filtered out first. This can be done with additional RegExes describing patterns that occur in stillborn certificates, such as *'ter wereld gebragt een kind van het manlijk geslacht'*.
- **Strikethrough**. We contacted the helpdesk of READ-COOP about our troubles in training strikethrough text (Section 5.2.1). They responded their priority is renewing the Transkribus web app. Their goal is to implement all functionalities from Transkribus Expert in it, to deprecate the Expert version in future. Once this is completed, they will improve the recognition of text styles, i.e., look at the strikethrough behaviour we identified. If we keep using Transkribus, we are dependent on them and can only wait for improvements. Unfortunately, lots of quality loss in our model is due to strikethrough; if it goes undetected this creates false positives in our data. So, other possibilities to detect strikethrough should definitely be explored.
- **Improve RegExes**. In our experiment, we showed what the performance of the Entity Extraction is when RegExes are built mostly quick and dirty. The positive results show potential that the current version of RegExes can be expanded further to include more cases and have less false positives, but also show how easy (within 8 hours) the majority of certificates is matched. So, this method is scalable to bigger data, but it is always beneficial to improve RegExes more, which can be done by looking at false positives (if ground truth data is available) and non-matches.
- **Use of death register database**. In Chapter 6, we concluded not all entities are useful for our own model due to interpretation differences (some entities are not matching the exact certificate text). However, the data is of good quality. Future work could explore whether parts can be turned into useful data, to either 1) be ready for release, or 2) serve as data in one of our model's components.
- **Ambiquity first and last names**. The last problem that is still open ended is regarding splitting first and last names. In Section 7.5, we saw that there exist a couple ways of writing down the name (e.g., last name in front of first name(s)). If these formats can be distinguished on a syntactical level (i.e., with punctuation), RegExes can make the split. However, we also observed certificates where the last name was not written down and should be derived from the parents. Then, RegExes can not determine whether the last word of the name is indeed the last name, or only the middle name of a person. Sometimes, other information from the certificate (e.g., names of informants and family) can help determine the split. However, this is not always the case. Future research might find correlations in how the split should be made, e.g., are children more likely to be written without last name, is the last name not written down when both parents are mentioned? Does it depend on the official who wrote the certificate? Once a database is established correlations between names and other entities can be analysed.
- **Other methods**. Another thing that could be explored in future is the use of other HTR and Entity Extraction methods. This thesis uses Transkribus, but other HTR tools do exist. We believe Transkribus is not the best in HTR quality, but it is definitely easy to use for historians (and other non-machine learning experts). Also, colleagues from the HDSC and other researchers from this field explored the use of Chat-GPT for Entity Extraction. If quality is good, it might be less time-consuming to ask Chat-GPT

for the entities from the text, then to write new RegExes all the time. This might not require an expert in the field of RegExes.

9.2 HTR improvements

In short, to improve the current HTR component in our pipeline, we suggest to 1) improve the existing training data, 2) expand the training data, and 3) experiment more with base models and parameters.

Improve existing training data

We suggest going over all datasets once more. In Sample_known, we have seen many names being spelled slightly differently, and suspect the database we investigated to be more correct. All those who differ are analysed and can be adjusted if needed. One can also inspect the machine output of Sample_regex and compare this to our own annotations; the names that differ should be manually reviewed again. We also advise to go over the training and validation sets once more. From our database analysis in Chapter 6, we learnt that our annotators' quality is less than that of the crowdsourcing for the database, so we suspect better transcriptions for our own data can be obtained when looking at the names more closely and spending more time per certificate. Also, we observed the possible bias that the two annotators might have failed at correcting mistakes from a good-performing HTR model. This calls for one more run over the data, especially to correct the transcription of names. Because names currently have bad HTR performance, it becomes even more important that the HTR model is fed correct data. Our dataset was small (size=100), errors could have had a larger effect on the performance. So, this future work of correcting the training data will definitely be executed in next months in the HDSC.

Expand training data

Transkribus reports that around 10,000 handwritten words should be enough to train a decent model. In our case, this is similar to 100 documents of a thousand words. However, because our certificates follow a certain pattern, it exists of many the same (printed) words. So, the performance on those words will be great, however, words that occur in the tail of the set of words, e.g., names, are difficult to recognise for the model. This might also be a reason for the bad performance on names we have seen. If so, adding more training data should be able to improve our quality. So, after this thesis, a a second version of the HTR model will be created with improved training data, also expanded with Sample_known and Sample_regex. This triples our training data in size. Although Transkribus works well on small data, we suspect that the repetition of printed words and the small number of names compared to the whole certificate text did not cover the wide variety of names to predict these well. To evaluate the updated model, we will create a new test set and hope to see a performance gain, especially for names.

HTR training

We already explored some new HTR models by tweaking parameters and training with the newly released base model 'the Dutchess'. During these experiments, we discovered that

lowering the batch size is needed to properly fine-tune large base models such as 'the Dutchess' and Transkribus' default base models. Figure 9.1 shows an updated overview with our new models (in grey, our trainings from chapter 5). One can see that better CER and WER scores are obtained than our best model (ID=3) which we used in Chapter 6 and 7. More research should point out whether this is indeed an improvement. For now, we view the scores to be very close to each other. So, we believe this does not significantly change the findings in our research chapters. In Figure 9.2, one can see that names of the deceased person, unfortunately, still have a very high WER (=38.18).

ID	Base model	HTR parameters	CER(train)	CER(val)	WER(val)
1.1	no basemodel	default settings	3.7	11.2	33.0
1.2	no basemodel	default settings	3.0	10.8	32.6
2.1	Transkribus Dutch Handwriting M2	default settings	84.4	91.2	100
3	IJsberg model	default settings	3.0	6.0	20.3
4	IJsberg model	default settings, ignore signatures	2.3	5.0	18.9
5	IJsberg model	default settings, with strikethrough	3.9	7.7	21.7
6.1	Dutchess model	default settings	90.9	94.8	100
2.2	Transkribus Dutch Handwriting M2	default settings, batch size 6	3.6	5.5	19.1
6.2	Dutchess model	default settings, batch size 6	2.9	5.2	18.3

Table 9.1: Training scores for the most interesting experiments. By default, batch size is 32.

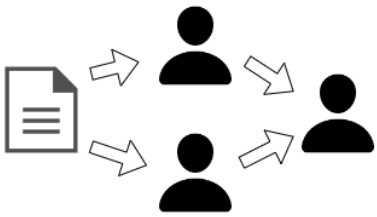
Model	Sample_regex data	CER	WER
5	Full text	5.01	13.39
3	Full text	3.51	9.43
3	Without marginalia	3.10	8.39
3	Without marginalia and signatures	3.10	8.48
3	Only line with name of deceased	16.06	39.48
6.2	Full text	4.30	12.77
6.2	Only line with name of deceased	15.46	38.18

Table 9.2: Evaluation on parts of the Sample_regex data with models from Table 9.1.

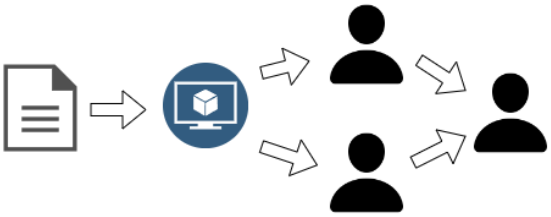
9.3 Model integrations

Bundling our knowledge obtained in our research chapters, here, we describe integration methods of our model into the current crowdsourcing process and its advantages and potential risks. This section serves as overview for the possibilities, but it does not take a final decision on what integration method is best. More research in Citizen Science should be conducted to observe the effects on the volunteers when a computer model is integrated in their process.

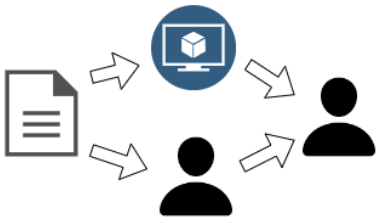
Figure 9.1 shows three possibilities how our model can be integrated. In the current crowdsourcing project (9.1a), each certificate is entered by two randomly selected volunteers. The transcription is then checked by a third in the control project in which both outputs are seen and a final decision is made. If both two outputs are the same, we think it unlikely the third person is making another final answer, but this is not impossible. Each certificate can be changed in the last round. This crowdsourcing process is slow, but the two-eye principle is ensuring quality.



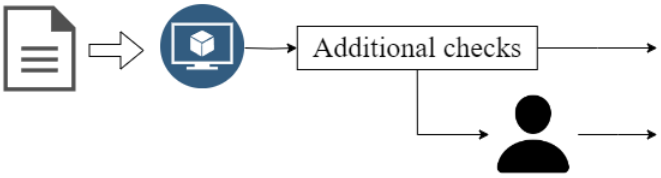
(a) Current crowdsourcing of two transcribers per document with a third in the control project.



(b) Integration of computer model showing pre-filled in entities that human transcribers need to check.



(c) Integration where computer model substitutes one human transcriber.



(d) Independent computer model with human transcription after certain filters.

Figure 9.1: The current crowdsourcing project (a) and three possible model integrations (b/c/d).

Pre-filled in entities

Figure 9.1b is a possibility the historians from the HDSC are considering. It might speed up the process, as volunteers see model output in the entity fields they need to fill in. The entities are already filled in and they only need to check the answer (and adjust if needed). The historians are discussing with current crowdsourcing platform Het Volk if this would be possible in their environment. However, we are a bit hesitant to use this 'pre-filled in fields' method. We have seen in Section 3.1 that it is important to the citizen scientists to feel meaningful which can be done by giving them meaningful tasks. Correcting computer output is substantially different than transcribing a document from scratch. Also, during our research, we observed our two annotators making sloppy mistakes (not changing model output). Some mistakes were obvious errors, like a wrong digit, other mistakes were more reasonable, like a couple characters off in names. Though, if spending a couple more minutes on the transcript, another spelling becomes more prominent (when looking at other characters). It could be that this phenomenon is solely occurring in our set-up, and volunteers in the HDSC-project are more eager to spot mistakes than our two students annotating. So, we suggest more research.

Model as human transcriber

In Figure 9.1c, one annotator is replaced by machine output. This is faster and does not deal with the disadvantages mentioned in the previous paragraph. Compared to the current crowdsourcing project, it is possible to save even more time when the control project does not check entities with two exact same answers (from machine and transcriber). We can assume the answer to be true. Although ground truth is unknown in real appliance of the model, we foresee five possible cases that can occur in this pipeline combined with crowdsourcing:

- **Both person and machine correct**. Desired behaviour.
- **Machine correct, but person incorrect**. This case, the document is sent to an extra human transcriber to make a final decision. It depends on the quality of our full pipeline how well our machine predictions are. It is unlikely that we will reach a state soon where the model is better than a human. Once this state is reached, no crowdsourcing would be needed anymore. For now, this case does not occur often, as the model is not great. This case is similar to two volunteers transcribing (one of them being incorrect) and a third choosing the correct answer.
- **Person correct, but machine incorrect**. Currently, this happens very often. We hope future work can improve the machine model. We do not think this scenario is harming quality that much, because every transcript is still seen by two annotators (maybe one if the first annotator and machine output are the same). This method is still more time efficient than the current crowdsourcing in which every certificate is looked at three times.
- **Both different incorrect answers**. It is perhaps difficult to determine any ground truth here. We believe this case can happen often right now, because are machine model is not that great yet, and one annotator does make some human errors. We hope the extra person in the control project can find a third answer which is the correct one. That person might be biased to choose the (incorrect) answer from the previous annotator, because it is more convincing than the rubbish output from the machine. We suggest the last person making the final decision to be an expert to reduce this problem.
- **Both same, but incorrect answers**. This will become a mistake in the database that goes undetected, if the certificate is not looked at again, or, when the last person missed to revision the answer. We argue that the machine output must have been a feasible answer, because chances are very small the annotator wrote down the exact same incorrect answer from the HTR+RegEx. To elaborate, when the RegEx finds a match at a totally different location, it is unlikely the annotator will have had the same wrong location, or, when the machine found some match in the marginalia, leading to a weird entity, it is unlikely the annotator has this too. Besides complete mismatches in our RegExes, we only found mistakes due to wrong HTR output, which we observed to be only 1 to a couple mistakes per word. We argue that this sounds similar to mistakes a human annotator can make. So, if we can make our HTR error rate similar to a human error rate, we believe this method can have similar quality as the current crowdsourcing (plus an efficiency boost).

Automatic extraction and some manual transcriptions

We did not include a fully independent computer model in the Figure, as we deem this impossible in any near future for it to be realistic. Figure 9.1d shows how the model could

operate on its own, though, if some additional checks are added to filter transcriptions that need some manual evaluation. This method has the highest efficiency, as human labour is limited. However, risks for mistakes are higher and our model can still be improved in multiple components, so this method is ambitious but not yet there. As additional checks, we envision four sorts of filters for entities: 1) entities with very high performance can immediately pass, 2) entities that should be present in each certificate, so filter the certificates without that entity for manual evaluation, 3) entities with high performance, but some filters can be added such as an alarming date (e.g., year outside expectation or invalid date such as 30-2), and 4) entities that were not great (e.g., names) so these should all be inspected at least once manually. So, based on model performances per entity, some mix between 9.1c and 9.1d could be made. We could also keep in mind the already available database. Perhaps this can serve as the first annotator in method 9.1c speeding up the process even more.

Conclusions

We have described possible ways how our model can be integrated into the current crowdsourcing process. This leaves many open research questions what method is best, e.g., do citizen scientists prefer pre-filled in fields? Does this harm final quality, because they are biased by what they see? And, how much time is saved by using pre-filled in fields? Maybe it is better to replace one of the annotators by model output. Furthermore, is it a desirable future in Citizen Science to create one fully functional and independent machine model? Right now, quality is not high enough (for us) to work without citizen scientists. However, integrating a computer model in crowdsourcing is likely affecting the citizen scientists. To keep them motivated and ensure the HDSC's future viability, it is crucial more research is done in the field of integrating AI into Citizen Science.

Bibliography

[Andro and Imad, 2017] Andro, M. and Imad, S. (2017). *Digital Libraries and Crowdsourcing: A Review: Towards Knowledge Ecosystems*, pages 135–161.

[Burnett, 2021] Burnett, J. (2021). Contemplating crowdsourcing in digital cultural heritage. Cultural Heritage Informatics initiative. `https://chi.anthropology.msu.edu/2021/12/contemplating-crowdsourcing-in-digital-cultural-heritage/` Accessed on 2023-06-08.

[Carbonell et al., 2018] Carbonell, M., Villegas, M., Fornés, A., and Lladós, J. (2018). Joint recognition of handwritten text and named entities with a neural end-to-end model. *CoRR*, abs/1803.06252.

[Carletti et al., 2013] Carletti, L., Giannachi, G., Price, D., McAuley, D., and Benford, S. (2013). Digital humanities and crowdsourcing: an exploration. In *MW2013: Museums and the Web*.

[Clotworthy, 2019] Clotworthy, A. (2019). Engaging the human in digital-humanities projects: How participating in crowdsourcing projects impacts quality of life among volunteers at the danish national archives. DHN2019 Conference date: 06-03-2019 through 08-03-2019.

[Dahl et al., 2023] Dahl, C. M., Johansen, T. S. D., Sørensen, E. N., Westermann, C. E., and Wittrock, S. (2023). Applications of machine learning in tabular document digitisation. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, 56(1):34–48.

[Deines et al., 2018] Deines, N., Gill, M., Lincoln, M., and Clifford, M. (2018). Six lessons learned from our first crowdsourcing project in the digital humanities. `https://blogs.getty.edu/iris/six-lessons-learned-from-our-first-crowdsourcing-project-in-the-digital-humanities/` Accessed on 2023-06-08.

[Ehrmann et al., 2023] Ehrmann, M., Hamdi, A., Pontes, E. L., Romanello, M., and Doucet, A. (2023). Named entity recognition and classification in historical documents: A survey. *ACM Comput. Surv.* Just Accepted.

[Fornés et al., 2017] Fornés, A., Romero, V., Baró, A., Toledo, J. I., Sánchez, J. A., Vidal, E., and Lladós, J. (2017). Icdar2017 competition on information extraction in historical handwritten records. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1389–1394.

[Grüning et al., 2019] Grüning, T., Leifert, G., Strauß, T., Michael, J., and Labahn, R. (2019). A two-stage method for text line detection in historical documents. volume 22, pages 285–302. Springer Verlag.

[Hamdi et al., 2021] Hamdi, A., Linhares Pontes, E., Boros, E., Nguyen, T. T. H., Hackl, G., Moreno, J. G., and Doucet, A. (2021). A multilingual dataset for named entity recognition, entity linking and stance detection in historical newspapers. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2328–2334, New York, NY, USA. Association for Computing Machinery.

[Huang et al., 2019] Huang, Z., Chen, K., He, J., Bai, X., Karatzas, D., Lu, S., and Jawahar, C. V. (2019). Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520.

[Kang et al., 2022] Kang, L., Riba, P., Rusiñol, M., Fornés, A., and Villegas, M. (2022). Pay attention to what you read: Non-recurrent handwritten text-line recognition. *Pattern Recognition*, 129:108766.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Pedersen et al., 2022] Pedersen, B.-R., Holsbø, E., Andersen, T., Shvetsov, N., Ravn, J., Sommerseth, H. L., and Bongo, L. A. (2022). Lessons learned developing and using a machine learning model to automatically transcribe 2.3 million handwritten occupation codes. *Historical Life Course Studies*, 12:1–17.

[Ponti et al., 2021] Ponti, M., Kloetzer, L., Miller, G., Ostermann, F. O., and Schade, S. (2021). Can't we all just get along? citizen scientists interacting with algorithms. *Human Computation*, 8(2):5–14.

[Ponti and Seredko, 2022] Ponti, M. and Seredko, A. (2022). Human-machine-learning integration and task allocation in citizen science. *Humanities and Social Sciences Communications*, 9:48.

[Prasad et al., 2018] Prasad, A., Déjean, H., Meunier, J., Weidemann, M., Michael, J., and Leifert, G. (2018). Bench-marking information extraction in semi-structured historical handwritten records. *CoRR*, abs/1807.06270.

[Puigcerver, 2017] Puigcerver, J. (2017). Are multidimensional recurrent layers really necessary for handwritten text recognition? In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 67–72.

[Puigcerver and Mocholí, 2018] Puigcerver, J. and Mocholí, C. (2018). Pylaia. `https://github.com/jpuigcerver/PyLaia` Accessed on 2023-06-13.

[Romero et al., 2013] Romero, V., Fornés, A., Serrano, N., Sánchez, J.-A., Toselli, A., Frinken, V., Vidal, E., and Lladós, J. (2013). The esposalles database: An ancient marriage license corpus for off-line handwriting recognition. *Pattern Recognition*, 46:1658–1669.

[Terras, 2016] Terras, M. (2016). Crowdsourcing in the digital humanities. In Schreibman, S., Siemens, R., and Unsworth, J., editors, *A New Companion to Digital Humanities*, pages 420–439, United Kingdom. Wiley-Blackwell.

[Terras, 2022] Terras, M. (2022). Inviting ai into the archives: The reception of handwritten recognition technology into historical manuscript transcription. In Jaillant, L., editor, *Archives, Access and AI*, Digital Humanities Research, pages 179–204. Transcript Verlag.

[Thompson et al., 2023] Thompson, K., Rosenbaum-Feldbrügge, M., and Quanjer, B. (2023). Maternal death and child mortality among enslaved populations in the dutch colonies suriname and curacao, 1839-1863. 5th Conference Of The European Society of Historical Demography, Nijmegen, the Netherlands.

[Toledo, 2019] Toledo, J. I. (2019). *Information Extraction from Heterogeneous Handwritten Documents*. PhD thesis, Universitat Autònoma de Barcelona.

[van Hyning, 2019] van Hyning, V. (2019). Curating crowds: A review of crowdsourcing our cultural heritage. In *DHQ: Digital Humanities Quarterly*, volume 13.

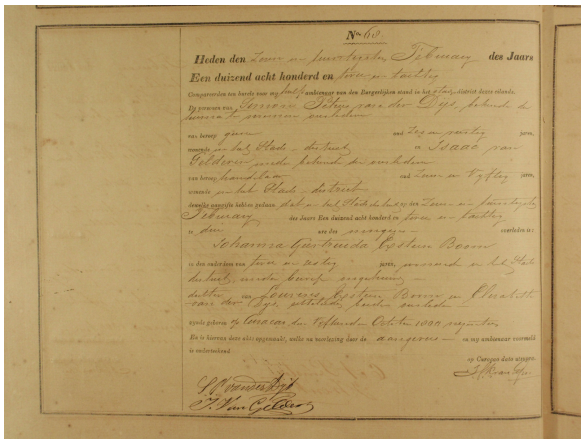
[van Oort et al., 2022] van Oort, T., Quanjer, B., Mourits, R., van Galen, C., and Kok, J. (2022). Hybrid indexation: Combining the strengths of htr and crowdsourcing. the case of the historical database suriname and curacao. Madrid, 4th Conference European Society of Historical Demography.

[Xu et al., 2015] Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2048–2057. JMLR.org.

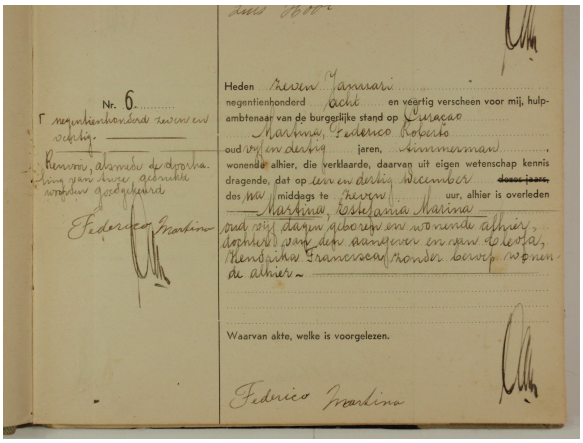
Appendix A

Examples of certificates

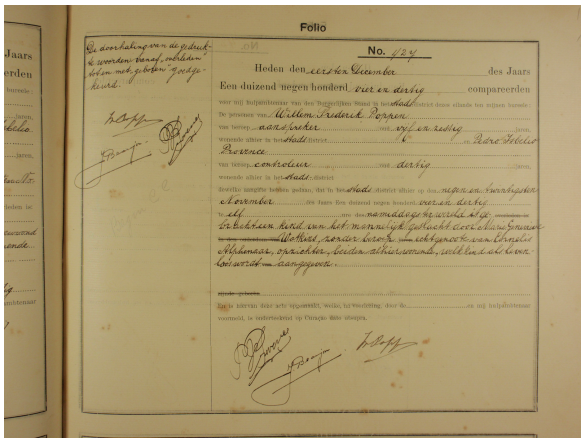
This Appendix shows examples of certificates discussed in Section 4.2.



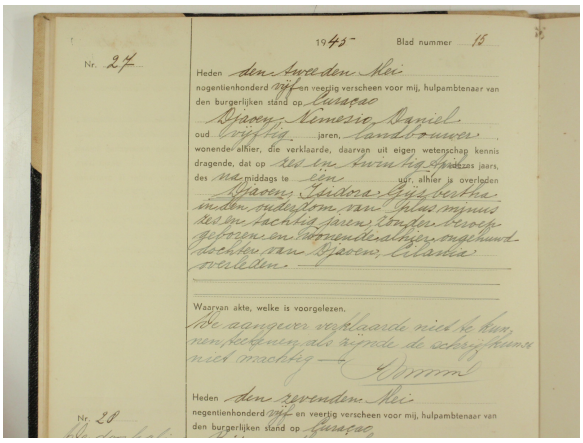
(a) A 'clean' certificate with readable text and no marginalia.



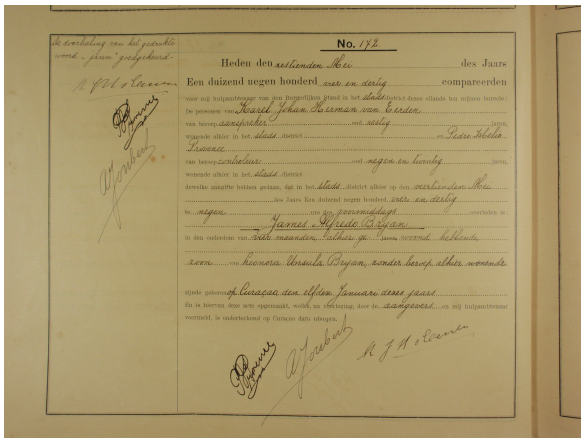
(b) A certificate with marginalia at the side. This case, the certificate year is adjusted to the year before. Most often, marginalia contain unimportant information.



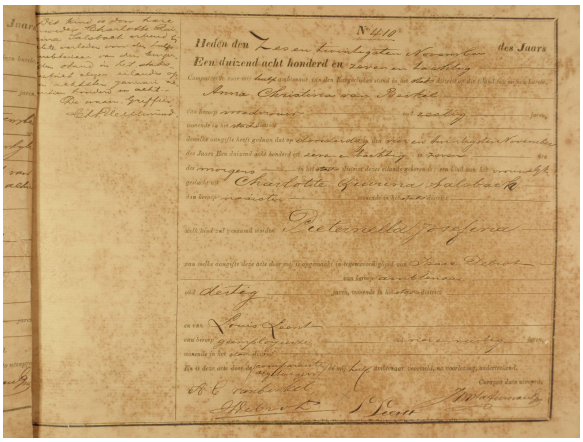
(c) A stillborn certificate with strikethrough.



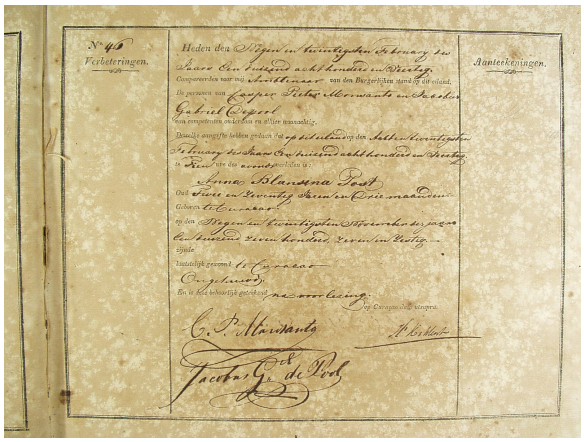
(d) A certificate with another certificate at the bottom that should be ignored.



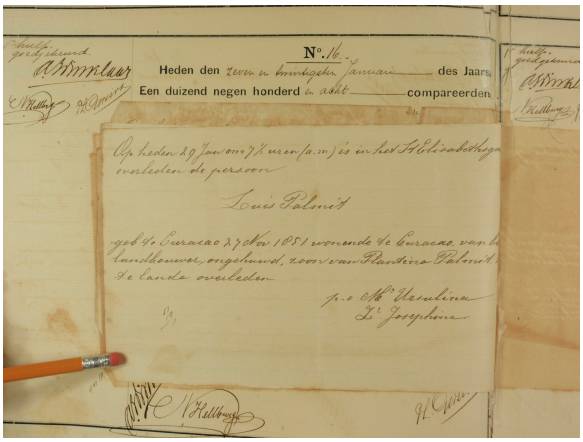
(e) A certificate with lots of whitespace, so the model does not detect one single baseline per line.



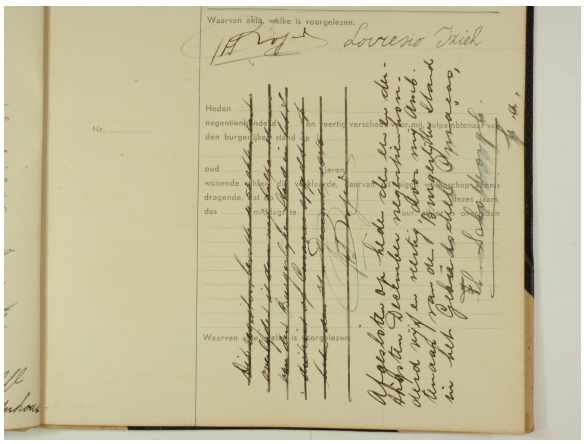
(f) A birth certificate with similar layout but different text and entities.



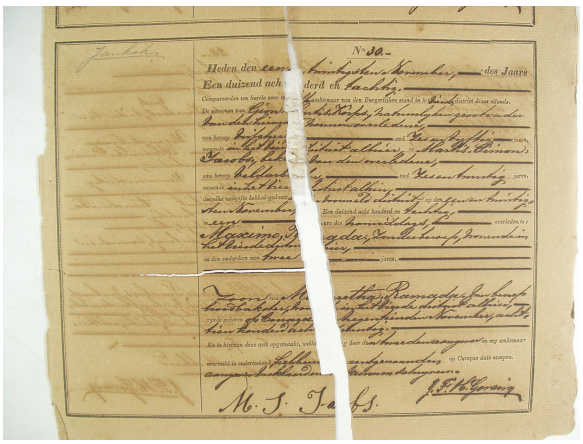
(g) A certificate with three text-regions. Certificates from 1831-1869 consisted of this format, but is not the focus of this thesis.



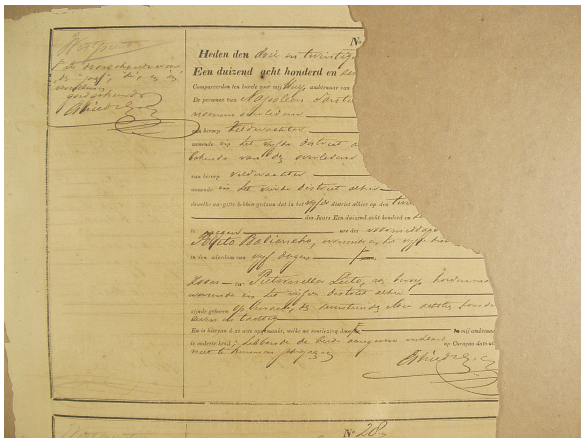
(h) A certificate on an inserted piece of paper; this does not follow our two-region format.



(i) This document contains vertical text. Our Transkribus models are not able to read this properly.



(j) A big tear in the certificate makes detecting baselines more difficult and erased some words from the certificate.



(k) A piece that is missing does not influence our models, however, some information can be lost.

Appendix B

Instructions Transkribus

These instructions can be followed to use our models in Transkribus. Note that these instructions are for the Transkribus Expert version, because we did not work in the Lite version ourselves. Transkribus is continuously improving and updating its system, so these instructions might be outdated soon.

We advise to read the preliminaries on Transkribus (Section 2.3) first, so one has a good understanding of the possibilities of Transkribus. For access to our Transkribus models, please contact the HDSC via `slavenregisters@let.ru.nl`

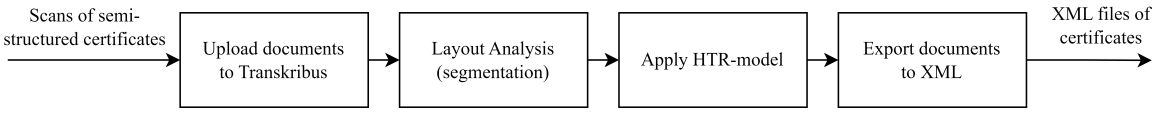


Figure B.1: Flow of Transkribus. (Same as Figure 5.1).

Figure B.1 shows the workflow for obtaining transcriptions from image data. Below, we give a detailed explanation for each of the components. This workflow can also be followed with default Transkribus models or one's own models, it is not restricted to the models developed in this thesis. We assume the user has created a Transkribus account and is logged in to the Expert version.

1. **Upload documents**. To keep data organised, one can create collections and documents within collections. These documents exist of pages, i.e., your images. To create a collection, click the bar next to 'Collections:' and click 'create'.
   Then, go to the collection and click 'Import document(s)' in the top bar of Transkribus. We advise to 'Extract and upload images from PDF', because there is a chance images will be upside down when uploaded as JPG.
   Once the document is uploaded, view 'Jobs' to see its status. Re-click your collection to reload the documents. Click on your document to see the images inside. For each image, the transcription box is empty, and regions and baselines do not exist yet.
2. **Layout Analysis**. To create layout (regions and baselines) automatically, go to 'Tools'. One can also execute HTR immediately which uses Transkribus default models to create the layout. In our thesis, we first create regions with P2PaLA, then, create baselines with a LA baseline model.

i. **P2PaLA**. Click 'P2PaLA...'. Select which pages to run the model on. In 'Select a model for recognition', the model 'P2PaLA_Curacao_bestModel' should be visible after we shared it with you. Leave other options at its default value. Then, click 'Run' and the P2PaLA model is started. This creates regions and baselines, but we can improve on the baselines by running LA_Curacao.

ii. **LA_Curacao**. Under 'Layout Analysis', keep 'Method: Transkribus LA' but click 'Configure...' to select our 'LA_Curacao' model. Keep it at its default values, except 'Max-dist for merging baselines' because we would like to merge as much baselines together. Put this value at 0.99. Before applying the model, select on which pages to run it and disable 'Find Text-Regions' (we only want to adjust the baselines), but do enable 'Split lines on regions' (because we would like to separate marginalia lines from the main certificate). Then, click 'Run' in the Layout Analysis section.

3. **HTR**. If satisfied with the regions and baselines (these can be manually adjusted), the HTR component obtains the transcription of each baseline. Under 'Text Recognition', click 'Run...' and choose your pages. For 'Select HTR model...', choose our HTR model. *It is important to enable the Language Model in the side bar* to benefit from the words the model learnt during training. Then, click 'Ok' twice. This action will cost you credits. After your job is finished, transcriptions are visible and linked to your image.
4. **Export documents**. To export, click 'Export document' in the top bar of Transkribus. This step is self-explanatory. 'Export page' creates an XML-document for each image. If desired, 'export Image' (JPG) or 'export text files' (TXT) can be useful too. We advise against exporting tags to Excel via 'Tag Export (Excel)', because this can randomly result in incomplete data. Tags can be extracted from the XML with a function provided in our GitHub repository[1].

[1] `https://github.com/LisaHoek/HTR-RegEx`

Appendix C

Named Groups for the RegExes

This appendix consists of dictionaries and functions that establish named groups for certain entities, i.e., dates, time, sex, etc. A full overview can be found in our GitHub repository[1].

C.1 Combine dictionary into RegEx-pattern

```
# From list to (e1|e2|e3) format
join_regex = lambda x: '(' + '|'.join(x) + ')'

# Create regex group with tag
def make_regex(named_group, group):
    return "(?P<"+named_group+">" + join_regex(group) + ")"
```

C.2 Numbers

```
# Specifies the digits below 10 to create numbers like "negentien honderd vier"
first10 = {
    "een": 1,
    "twee": 2,
    "drie": 3,
    "vier": 4,
    "vijf": 5,
    "zes": 6,
    "zeven": 7,
    "acht": 8,
    "negen": 9,
}

# Specifies the digits 11-19 to create years like "negentien honderd achttien"
second10 = {
    "elf": 11,
    "twaalf": 12,
    "dertien": 13,
    "veertien": 14,
    "vijftien": 15,
    "zestien": 16,
```

[1] https://github.com/LisaHoek/HTR-RegEx

```
    "zeventien": 17,
    "achttien": 18,
    "negentien": 19,
}

# Specifies the digits 10,20,30... to create years like "negentien honderd drie en twintig" or "negentien honderd dertig"
tens = {
    "tien": 10,
    "twintig": 20,
    "dertig": 30,
    "veertig": 40,
    "vijftig": 50,
    "zestig": 60,
    "zeventig": 70,
    "tachtig": 80,
    "negentig": 90,
}

# Specifies the century to create years like "achttien honderd" and "achttienhonderd negen en tachtig"
hundreds = {
    "achttien\W*honderd": 1800,
    "negentien\W*honderd": 1900,
    "een\W*duizend\W*acht\W*honderd": 1800,
    "een\W*duizend\W*negen\W*honderd": 1900,
}

# Specifies options of full out written years
year = make_regex("hundreds", hundreds)+"\W{0,3}(en)?\W{0,3}("+make_regex("first10", first10)+"|"+make_regex("first10", first10)+"\W{0,3}(en)?\W{0,3}"+make_regex("tens", tens)+"|"+make_regex("second10", second10)+"|"+make_regex("tens", tens)+")"
```

C.3 Dates

```
# Specifies the days of a month
days = {
    "een|eersten?": 1,
    "twee(den?)?": 2,
    "drie|derden?": 3,
    "vier(den?)?": 4,
    "vijf(den?)?": 5,
    "zes(den?)?": 6,
    "zeven(den?)?": 7,
    "acht(sten?)?": 8,
    "negen(den?)?": 9,
    "tien(den?)?": 10,
    "elf(den?)?": 11,
    "twaalf(den?)?": 12,
    "dertien(den?)?": 13,
    "veertien(den?)?": 14,
    "vijftien(den?)?": 15,
    "zestien(den?)?": 16,
    "zeventien(den?)?": 17,
```

```
    "achttien(den?)?": 18,
    "negentien(den?)?": 19,
    "twintig(sten?)?": 20,
    "een ?en ?twintig(sten?)?": 21,
    "twee ?en ?twintig(sten?)?": 22,
    "drie ?en ?twintig(sten?)?": 23,
    "vier ?en ?twintig(sten?)?": 24,
    "vijf ?en ?twintig(sten?)?": 25,
    "zes ?en ?twintig(sten?)?": 26,
    "zeven ?en ?twintig(sten?)?": 27,
    "acht ?en ?twintig(sten?)?": 28,
    "negen ?en ?twintig(sten?)?": 29,
    "dertig(sten?)?": 30,
    "een ?en ?dertig(sten?)?": 31,
}

# Specifies the months in a year
months = {
    "januar[iy]": 1,
    "februar[iy]": 2,
    "maart": 3,
    "april": 4,
    "mei": 5,
    "jun[iy]": 6,
    "jul[iy]": 7,
    "augustus": 8,
    "september": 9,
    "o[kc]tober": 10,
    "november": 11,
    "december": 12,
}
```

C.4 Sex

```
# Determines the sex of the deceased person
sex = {
    "weduwnaar": "m",
    "weduwe": "v",
    "echtgenoot\\b": "m",
    "echtgenoote\\b": "v",
    "mannelijk": "m",
    "vrouwelijk": "v",
    "zoon": "m",
    "dochter": "v",
}
```