



**Universität  
Zürich** <sup>UZH</sup>

Masterarbeit  
zur Erlangung des akademischen Grades  
**Master of Arts**  
der Philosophischen Fakultät der Universität Zürich

# Named Entity Recognition in Digitized Historical Texts

**Verfasserin: Yvonne Gwerder**

Matrikel-Nr: 10-737-831

Referent: Prof. Dr. Martin Volk

Betreuerin: M.Sc. Natalia Korchagina

Institut für Computerlinguistik

Abgabedatum: 29.06.2017

## **Abstract**

The thesis at hand presents an approach at automatically recognizing Named Entities in legal documents written in late medieval and early modern variants of German and French. It describes the transformation of the digitized texts into a structured XML format, while exemplifying how resources for tokenization and OCR-processing can be adapted and applied to this end. Named Entities are extracted by exploiting indices of place and person names, and subsequently detected in the texts via approximate string matching techniques. The resulting pre-annotated texts are then tagged with a ready-made Named Entity Recognition tool intended for the modern language. Ultimately, by training and testing own Machine Learning models, the thesis aims at illustrating the main possibilities and limitations characteristic of historical data.

## **Zusammenfassung**

Die vorliegende Masterarbeit befasst sich mit dem automatischen Erkennen von Eigennamen in deutschen und französischen Rechtsquellentexten, welche im Spätmittelalter und in der frühen Neuzeit entstanden sind. Die Arbeit beschreibt die Umwandlung der digitalisierten Texte in ein strukturiertes XML-Format, wobei die Verwendung sowie Anpassung von Tokenisierungs- und OCR-Software erläutert wird. Die Eigennamen werden zunächst aus den entsprechenden Orts- und Personenregistern extrahiert und danach mit Hilfe von unscharfen Suchmethoden im Text lokalisiert. Die daraus resultierenden vor-annotierten Texte werden zusätzlich mit einem Standard-Tool für Eigennamenerkennung getaggt. Abschliessend sollen durch das Trainieren eigener statistischer Modelle die spezifischen Besonderheiten und Probleme von historischen Texten veranschaulicht werden.

# Acknowledgement

I would like to express my gratitude to all people who assisted me with my thesis.

I am very grateful for having had **Natalia Korchagina** as my supervisor, and I would like to thank her for being a very competent and patient mentor. Starting from the point when this thesis was merely an idea for a seminar project up to completion, her advice and expertise has always been of much help.

Many thanks go to **Martin Volk**, not only for assisting me in organizational matters, but also for providing me with precious ideas as well as motivation. In addition, he enabled and supported the collaboration with the *Swiss Law Sources Foundation*. In this regard, I would also like to thank and credit **Pascale Sutter** who offered me the unique opportunity to work with data from the *Collection of Swiss Law Sources*.

Special thanks are dedicated to my fellow student **Susie Xi Rao**, as she inspired me to continue our group project, and ceaselessly helped me throughout the whole semester with all kinds of matters.

Finally, I want to express my gratitude to my supportive friends and family members whose encouragement has been very much appreciated.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgement</b>	<b>ii</b>
<b>Contents</b>	<b>iii</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Acronyms</b>	<b>viii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Research Questions . . . . .	3
1.3 Thesis Structure . . . . .	4
<b>2 Theoretical Background</b>	<b>5</b>
2.1 NLP for historical texts . . . . .	5
2.1.1 General challenges . . . . .	5
2.1.2 Dealing with spelling variation . . . . .	7
2.1.2.1 Normalization . . . . .	7
2.1.2.2 String similarity measures . . . . .	8
2.2 Named Entity Recognition . . . . .	11
2.3 NER in historical texts . . . . .	14
2.3.1 Rule-based approaches . . . . .	14
2.3.2 Data-driven approaches . . . . .	16
<b>3 Text Pre-processing and Corpus Creation</b>	<b>19</b>
3.1 Data . . . . .	19
3.1.1 The <i>Collection of Swiss Law Sources</i> . . . . .	19
3.1.2 The SSRQ digitization project . . . . .	21
3.1.3 Selection of volumes . . . . .	23
3.2 Pre-processing . . . . .	25

3.2.1	XHTML parsing . . . . .	25
3.2.2	Restoring pagination . . . . .	26
3.2.2.1	Problem . . . . .	26
3.2.2.2	OCR approach . . . . .	27
3.2.2.3	Evaluation . . . . .	28
3.2.3	Cleanup . . . . .	30
3.2.4	Sentence boundary detection . . . . .	32
3.2.5	Tokenization . . . . .	33
3.2.6	Remaining issues . . . . .	35
<b>4</b>	<b>Name Extraction and Recognition</b>	<b>36</b>
4.1	Indices of place and person names . . . . .	36
4.1.1	Pre-processing . . . . .	38
4.2	NE database compilation . . . . .	39
4.2.1	Extracting and generating name variants . . . . .	39
4.2.2	Extraction of other information . . . . .	43
4.2.3	Post-processing and output . . . . .	44
4.2.4	Remaining issues . . . . .	45
4.3	Recognition and annotation of entities . . . . .	46
4.3.1	Procedure . . . . .	47
4.3.2	Direct and approximate search . . . . .	48
4.3.3	Stand-off annotation . . . . .	50
4.3.4	Evaluation . . . . .	51
4.3.4.1	Precision . . . . .	51
4.3.4.2	Recall . . . . .	53
<b>5</b>	<b>Machine Learning Approach</b>	<b>57</b>
5.1	Applying modern language NER systems . . . . .	57
5.1.1	<i>Stanford NER</i> . . . . .	58
5.1.1.1	Classifiers . . . . .	58
5.1.1.2	Building an IO-tagged corpus . . . . .	60
5.1.1.3	Comparison and evaluation . . . . .	61
5.2	CRF classifier training with <i>Wapiti</i> . . . . .	65
5.2.1	Data and features . . . . .	65
5.2.2	Evaluation . . . . .	67
5.2.2.1	Silver-standard comparison . . . . .	67
5.2.2.2	Gold-standard comparison . . . . .	68
<b>6</b>	<b>Conclusion</b>	<b>71</b>
6.1	Summary . . . . .	71

6.2 Outlook . . . . .	73
<b>References</b>	<b>74</b>
<b>Curriculum Vitae</b>	<b>81</b>
<b>A Python Scripts</b>	<b>82</b>
<b>B <i>Wapiti</i> Feature Templates</b>	<b>84</b>
B.1 Template 1 . . . . .	84
B.2 Template 2 . . . . .	84
B.3 Template 3 . . . . .	86
<b>C Modules and Tools</b>	<b>87</b>

# List of Figures

1	Spelling variation in historical documents . . . . .	6
2	Representation of nested NEs . . . . .	11
3	OCR errors in Parliamentary records . . . . .	14
4	SSRQ manuscript and transcription . . . . .	20
5	SSRQ print and XHTML . . . . .	22
6	OCR comparison table . . . . .	28
7	XHTML line and footnote markers . . . . .	31
8	Usage of square brackets . . . . .	33
9	XML corpus excerpt . . . . .	35
10	Index of place and person names . . . . .	37
11	Problematic index entry . . . . .	45
12	Stand-off annotation file . . . . .	50
13	Error sources for False Negatives . . . . .	55
14	Project pipeline . . . . .	73

# List of Tables

1	Levenshtein distance . . . . .	9
2	NE types . . . . .	12
3	Selected SSRQ volumes . . . . .	24
3	Error rates in the OCR-processed pages . . . . .	29
4	Tokens and types of the selected volumes . . . . .	34
5	Spelling variants with parentheses . . . . .	40
6	Abbreviations and written out names . . . . .	42
7	NE and class distribution . . . . .	42
8	Precision of approximate search . . . . .	51
9	Examples of correct and wrong matches . . . . .	52
10	Query and entry recall . . . . .	54
11	Number of <i>Stanford NER</i> and rule-based tags . . . . .	62
12	Distribution of class tags . . . . .	62
13	Precision of <i>Stanford NER</i> . . . . .	63
14	Tab-separated corpus format . . . . .	66
15	Classifier evaluation against silver-standard . . . . .	68
16	Classifier evaluation against gold-standard . . . . .	69

# List of Acronyms

BIO	Beginning Inside Outside
CER	Character Error Rate
CRF	Conditional Random Fields
CSS	Cascading Style Sheet
CSV	Comma Separated Values
HTML	Hypertext Markup Language
IE	Information Extraction
LCS	Longest Common Substring
ML	Machine Learning
NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language Tool Kit
OCR	Optical Character Recognition
OLR	Optical Layout Recognition
PDF	Portable Document Format
PER	Position-independent Word Error Rate
PNG	Portable Network Graphics
POS	Part-Of-Speech
QA	Question Answering
SSRQ	Sammlung Schweizerischer Rechtsquellen
TEI	Text Encoding Initiative
UTF-8	Unicode Transformation Format (8-bit)
UZH	University of Zurich
WER	Word Error Rate
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language

# 1 Introduction

## 1.1 Motivation

Named Entity Recognition (also referred to as NER) is a subtask of Natural Language Processing (NLP) and plays a fundamental role in the field of Information Extraction (IE). With the rise of big data and social media, as well as the resulting need for efficient extraction mechanisms for proper names, NER has certainly gained interest during the past decade. A Named Entity (NE) is usually defined as a linguistic expression referring to an individual object in the real world, such as a person, organization, product, or location (Carstensen et al., 2010, 596). Depending on the application of the NER system, other NE categories may come into play, including expressions of time (e.g. specific dates) or measurement (e.g. weight, currency). Most of the current NER systems are based on data-driven methodologies, using supervised or semi-supervised Machine Learning (ML) techniques on large amounts of human annotated data (cf. Benikova et al. (2014), Nothman et al. (2013)). Thus, state-of-the-art NER systems are expected to achieve best results when applied on well-resourced languages and genres (such as English newspaper texts). However, numerous historical languages (but also regional language varieties) are less-resourced, so that only limited annotated material for them is available. Manually annotating these texts may be challenging if not impossible, since the variety at hand could be mostly unknown or outdated. In addition, historical texts usually do not have a normalized orthography. This fact poses special challenges for conventional NLP, as many of its aspects "critically rely on the assumption that the text to be processed is written in a single language with a standardized orthography" (Piotrowski, 2012, 3). In summary, NLP presently lacks "a conceptual model of spelling variation, genre differences, and language change", while these aspects are characteristic of historical documents (Piotrowski, 2012, 9). As a result, most of the currently prevalent NLP applications – including Part-of-Speech (POS) tagging, syntactical parsing and morphological analysis – do not perform well on this type of text. The number of available tools designed for specific historical data is sparse, and only few of them focus on NER.

Usually, Named Entities in historical corpora are of high interest for cultural, social and political research within a specific historical period, which makes NER a valuable process in the Digital Humanities. Once certain NEs in a text have been identified, it is possible to investigate different aspects of them, such as an organization's or individual's area of influence, their co-occurrence and interaction with other individuals, or which currencies they used within a given period of time, etc. It is also possible to research entities from a linguistic point of view and establish the different ways of spelling the same proper name. From a diachronic perspective, it could be of interest to explore how and where, for instance, place names have changed (or even disappeared). Sometimes the aim is to create a gazetteer listing all names of persons and places occurring within a document, so that a reader can easily look them up. Here, NER can be applied to automate the process and avoid – or at least reduce – time-consuming manual work. These are only a few examples of how research in the humanities may profit from NER, and a lot of possibilities are still to be discovered.

The thesis at hand has been motivated by the challenges for current NLP in relation to historical texts on the one hand, and by the importance of NER for the Digital Humanities on the other. Collaborating with the *Swiss Law Sources Foundation*<sup>1</sup>, the UZH Institute of Computational Linguistics presently conducts investigations addressing the extraction of temporal entities from historical legal texts<sup>2</sup>. I have been given the opportunity to write my thesis in cooperation with the *Swiss Law Sources Foundation* which provided me with digitized texts from their database, the *Collection of Swiss Law Sources*<sup>3</sup>. This database encompasses over 100 volumes written in historical variations of German, French, Italian, as well as Latin, corresponding to more than 70'000 pages of source material. All texts originate from regions nowadays belonging to Switzerland and include a time span from the Middle Ages up to the Early Modern era (1798). Since the source material has been digitized and made available in different file formats, it is now possible to process the data within an experimental NLP framework. The overarching goal of my research was to model a NER pipeline for the historical variety of two languages, German and French, which includes pre-processing of the data, extracting NEs via rule-based and statistical algorithms, as well as delivering corpus data and NE-annotated files which may ultimately provide a basis for future projects.

---

<sup>1</sup>*Schweizerische Rechtsquellenstiftung*; <https://www.ssrq-sds-fds.ch/>

<sup>2</sup>This project, launched in 2014, is lead by Prof. Dr. Martin Volk and Dr. Pascale Sutter. Research is being conducted by Natalia Korchagina. Cf. <http://www.cl.uzh.ch/en/research/digital-humanities/hist-temporal-entities.html>.

<sup>3</sup>*Sammlung Schweizerischer Rechtsquellen* (SSRQ); <https://www.ssrq-sds-fds.ch/online/>.

## 1.2 Research Questions

Apart from creating a NER pipeline as mentioned above, another objective in my thesis was to experiment with several different applications from various sections of automatic text processing, and discover how they can be modified and implemented to work with historical data. Moreover, I wanted to illustrate what kinds of problems arise when applying off-the-shelf tools which are meant for processing modern texts. In a more general sense, the thesis also aimed at demonstrating the challenges, problems and intricacies of NLP with historical data. The following research questions shall be addressed:

- How can string comparison methods be efficiently employed for corpus pre-processing and recognizing NEs? Which techniques work best for the given data?
- How well does an off-the-shelf NE tagger perform on the historical texts?
- Is it possible to train an own Machine Learning system based on the output of different NER methods? If so, what kind of results can be expected?
- How can the pre-processed data be arranged in a clearly structured and reusable format?

In order to answer these questions, the project described in this thesis has been divided into a sequence of processing steps. The most important ones are as follows:

1. Pre-processing of the source texts and creating machine-readable corpus files for each volume. This involves:
  - a) Restoring the original layout with the help of Optical Character Recognition (OCR) and string comparison methods
  - b) Sentence boundary recognition and tokenization
2. Extraction of NEs using regular expressions and rule-based filtering methods
3. Finding and annotating the extracted NEs in the corpus by using approximate string matching techniques
4. Applying a ready-made NE tagger and combining the output in order to train an own statistical model for NER
5. Based on the extracted NEs, automatically creating annotation files and dictionary-like databases.

## 1.3 Thesis Structure

In the following chapter 2, I outline the main concepts and principles of NER while addressing the issue of NER in the cultural heritage domain. Chapter 3 introduces the data used for this thesis, and summarizes the corpus compilation and cleanup process. In chapter 4 I first describe secondary sources – the indices of place and person names – that were used for storing names, spelling variants, and other relevant information into a NE database. Next, I delineate the hand-crafted rules and algorithms for locating the NEs in the text. Chapter 5 explores the possibilities and limitations of Machine Learning, and surveys the usage of ready-made systems. Since chapters 3 to 5 deal with different components of the pipeline, the evaluations are given within the corresponding sections. I therefore abstained from adding a separate chapter dedicated to an overall evaluation. The last chapter 6 summarizes the thesis project, and mentions possible future work.

## 2 Theoretical Background

This chapter aims at giving a general introduction to the challenges of working with historical texts. It also summarizes the task of Named Entity Recognition, and defines the most important concepts related to the task. The last part of the chapter presents a selection of previous studies dealing with NER in historical texts.

### 2.1 NLP for historical texts

#### 2.1.1 General challenges

Historical documents have not been used for NER (or any NLP application in general) for a relatively long time. This is due to two main reasons: firstly, a large number of documents have not been available in any digital form, and secondly, the specific linguistic properties and orthographic variation in historical texts pose many challenges to NLP. However, there is now an increasing interest in making documents accessible not only for a public readership, but more specifically for historians, linguists, and paleographers. With the rise of the Digital Humanities, more and more historical documents (such as old books and letters) are being digitized. Since many of the documents are threatened with physical ruin, this is often the only way to preserve their cultural heritage.

Thus, digitization is the first important step in making historical data accessible. (Hauser et al., 2007, 147) state that this step "in symbolic form opens the door for using modern techniques of information access such as Information Retrieval, text mining, hyperlinking, flexible rendering and presentation of documents". Since merely scanning or photographing documents grants only very limited access, digitization usually involves manual transcription or OCR. Both of these processes pose challenges when it comes to historical texts. Depending on the origin of the source material, the text may be written in a specific historical typeface (e.g. Gothic type) or it may possess other paleographic features (e.g. majuscules). Manual transcription, in that case, requires expert knowledge, and is certainly very time-consuming.

It seems that applying OCR on historical documents would be the easiest and fastest way of transforming the input into machine-readable text formats; yet this assumption is far from true. The vast majority of OCR systems are trained to process present-day fonts such as sans-serif and are therefore error-prone with historical texts (which may also have poor material quality, inducing more errors into OCR). In addition, Optical Layout Recognition (OLR) software often struggles with reconstructing the original text flow in "complex magazine-like layouts (with images, text boxes, non-rectangular text blocks, etc.), initials, marginals, captions, or footnotes" (Piotrowski, 2012, 31).

The biggest challenge to NLP of historical texts, however, is the absence of an orthographic standard connected with the presence of a big variety in different spellings. The orthography in German documents, for example, has not been formally regulated until 1901. Figure 1 illustrates the frequency of variant spellings found in German documents over a time span of 450 years, based on a text normalization project by Ernst-Gerlach and Pilz (2006). The high percentages be-

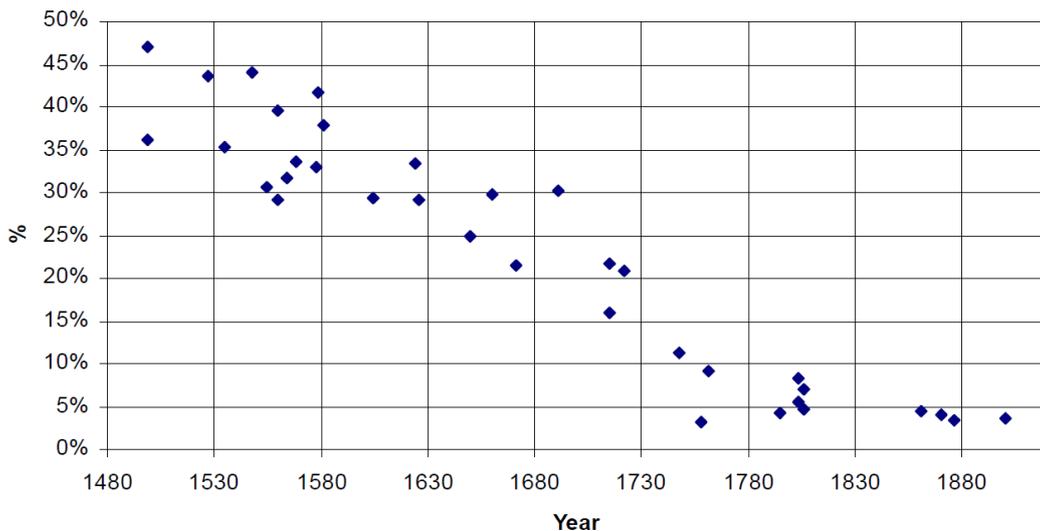


Figure 1: Frequency of variant spellings in German historical text documents from 1480 to 1885, adapted from (Kempken et al., 2006, 2)

fore orthographic standardization, and especially before the 1700s, indicate a great number of different spelling variants for the same words. As (Piotrowski, 2012, 11) states, this extent of variation is a problem for numerous applications in language technology:

NLP methods generally assume that the text to process adheres to a consistent orthography. This is because, at some point, most NLP techniques need to consult lexical resources to obtain information (e.g.,

part of speech, case, or number) about the word forms encountered in the text, and the word form serves as key for accessing this information.

Due to the issues mentioned above, studies focusing on IR (and more specifically on NER) in historical texts are still relatively marginal.

## 2.1.2 Dealing with spelling variation

### 2.1.2.1 Normalization

Spelling variation complicates the development of both ML and rule-based NER systems. ML needs meaningful features which are often based on character sequences of tokens; the more overall variation in the tokens, the harder it will be to make correct predictions. On top of that, the absence of available linguistic tools for historical texts such as POS-taggers and morphological analyzers makes it impossible for a ML system to learn from anything other than the tokens in the text. Rule-based systems rest upon external resources, but in the context of historical NER, these are often insufficient or – at worst – non-existent. Ultimately, general techniques such as frequency profiling, collocation extraction or concordancing will yield poor results. For this reason, string similarity measures are an essential component when dealing with historical language varieties. They are commonly used in text processing and NLP applications<sup>1</sup> to calculate the similarity or distance (therefore also called distance metrics) between strings. For historical texts, they are most useful in the process of normalization (Piotrowski, 2012, 69ff.).

The goal of normalization is ”to bring the historical language closer to the modern language in order to enable the use of the methods, tools, and resources available for the modern language. Thus, given a historical word form, we are interested in a corresponding modern word form” (Piotrowski, 2012, 69). This can be achieved by finding the modern word form (in a given reference file or dictionary) which is closest to the historical variant, i.e. which has the highest similarity score. It is also possible to perform normalization automatically or semi-automatically by creating sets of replacement rules or by making a ML system learn the rules by itself.

An exemplary study which used both approaches was outlined by Pilz et al. (2008). Their goal was to create a system which detects and normalizes both German and English historical spelling variants to their modern equivalent in running text (originating in a period between the 13th and the 16th century). On the basis of manually

---

<sup>1</sup>E.g. proper name matching, OCR correction, dialectometry, data linkage and deduplication, or spelling error correction.

collected one-to-one word mappings between non-standard and modern spellings<sup>2</sup>, letter replacement heuristics were derived both manually and automatically. The authors reported good intermediate results; for instance, the three most frequently used replacement rules for German achieved precision scores of over 94 % (Ernst-Gerlach and Pilz, 2006).

(Bollmann, 2012, 2), on the other hand, combined and compared different normalization methods that can be flexibly trained to handle spelling varieties. He stressed that character rewrite rules (as examined in the work by Pilz et al. (2008)) should avoid the generation of nonsensical words, which can be achieved by considering the frequency of the rule during training, and by matching normalization candidates against a reference lexicon (Bollmann, 2012, 3). The study also summarized the *FlexMetric* and *MultiWLD* measures for normalization that are easily adaptable to different varieties. In general, these measures are modifications of the *Levenshtein distance* (cf. section 2.1.2.2 below) with trainable weights for string operations. The advantage of both rests upon the flexible parametrization by manual inspection of only a small portion of the input data. When combining *FlexMetric* and *MultiWLD* with rule-based methods, (Bollmann, 2012, 9) achieved a normalization accuracy of around 93 % using the 1545 version of the Luther bible.

### 2.1.2.2 String similarity measures

There is a great number of different measures and algorithms used for approximate string matching. One matching technique is called **phonetic encoding** which converts a string into an alphanumeric code based on its pronunciation. That means that strings with similar codes are supposed to be phonetically similar. Most of the phonetic encoding techniques, including *Metaphone* as well as *Soundex* and its variations (i.e. *Phonex*, *Phonix*, *NYSIIS*) have been developed for the English language (Christen, 2006, 292). Some of them attempt to better account for non-English input, e.g. the *Meyer-Wilde* algorithm which was adapted to German (Meyer and Wilde, 1988).

Other measures are based on string similarity or edit distance. A very straightforward and common distance metric is the ***Levenshtein distance*** developed in the 60s. It is defined as "the smallest number of edit operations (insertions, deletions and substitutions) required to change one string into another" (Christen, 2006, 292). By default, the cost for each edit operation is set to 1, but can be changed according to the desired strictness (which results in a weighted form of the algorithm, often

---

<sup>2</sup>12'621 for German and 45'805 for English (Pilz et al., 2008, 66ff.).

referred to as Weighted *Levenshtein* Distance). For instance, by setting the cost for substitution to 2, the edit distance between the strings 'Himmel' and 'Hymel' is 3. Table 1 demonstrates how the distance is calculated; note that the result is displayed in the bottom right corner of the table.

		H	y	m	e	l
	0	1	2	3	4	5
H	1	0	1	2	3	4
i	2	1	2	3	4	5
m	3	2	3	2	3	4
m	4	3	4	3	4	5
e	5	4	5	4	3	4
l	6	5	6	5	4	3

Table 1: *Levenshtein* distance between the strings 'Himmel' and 'Hymel', based on (Ernst-Gerlach, 2013, 27)

A variation of the *Levenshtein* distance is the *Damerau-Levenshtein* distance in which transposition is considered to be a single edit operation with cost 1, instead of two different operations (Kempken, 2005, 15).

The ***Hamming distance*** is another edit distance metric similar to *Levenshtein* which counts the minimum number of errors (i.e. substitutions) required to change one string into the other. In contrast to *Levenshtein*, it is only defined for strings of equal length, and is much faster for long sequences (Forouzan, 2007, 274ff.).

Another established algorithm called ***Jaro distance*** is frequently used in data linkage tasks. It calculates the edit distance using the number of transpositions as well as the number of common characters in relation to the string length. Common characters are all identical characters that occur within a given distance (by default, this distance is defined as half the length of the longer string) (Kempken, 2005, 19). Given the two strings  $s_1$  and  $s_2$ , the number of common characters  $c$ , and the number of transpositions  $t$ , the *Jaro* similarity can be calculated as follows:

$$sim_{jaro}(s_1, s_2) = \frac{1}{3} \left( \frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-t}{c} \right) \quad (2.1)$$

Equation 2.1 returns a normalized score such that 0 is equal to no similarity and 1 is an exact match. The *Winkler* algorithm improves upon the *Jaro* metric by increasing the similarity measure for agreeing initial characters (up to four) (Christen, 2006, 292). This is based on the assumption that differences near the start of the string are more significant (ibid.).

Thus, the **Jaro-Winkler similarity** is often used for person name matching, and is calculated as

$$sim_{wink}(s_1, s_2) = sim_{jaro}(s_1, s_2) + \frac{s}{10} (1 - sim_{jaro}(s_1, s_2)) \quad (2.2)$$

with  $s$  being the number of agreeing characters at the beginning of two strings.

Whereas the aforementioned algorithms basically rely on the editing operations of two strings, the **n-gram** approach operates by dissecting them into sub-strings of length  $n$ . Frequently used n-grams are unigrams ( $n=1$ ), bigrams ( $n=2$ ) and trigrams ( $n=3$ ). The string 'Himmel', for example, contains the bigrams 'Hi', 'im', 'mm', 'me', 'el' (Ernst-Gerlach, 2013, 27). The idea behind this measure is that the more n-grams are shared by two strings, the more similar they are. The n-gram similarity measure is calculated "by counting the number of n-grams in common and divide by either the number of n-grams in the shorter string (called overlap coefficient), the number in the longer string (called Jaccard similarity) or the average number of n-grams in both strings (called the Dice coefficient)" (Christen, 2006, 292). There are also extensions and modifications to the n-gram approach such as the comprehensive n-gram similarity, the positional n-gram similarity, or skip-grams (cf. Kondrak (2005) for more detailed descriptions).

The **Longest Common Sub-string (LCS)** technique tries to find the longest character sequence that is a sub-sequence of both strings. (Christen, 2006, 292) claims that this algorithm is "suitable for compound names that have words (like given- and surname) swapped". Similarity is calculated by dividing the total length of the common sub-strings by the minimum, maximum or average lengths of the two original strings.

Of course there are numerous other string similarity measures (and variations) such as *Editex*, *Covington*, *Jaccard*, *Monge-Elkan*, *Smith-Waterman*, and so on. A practical compendium of many of them is given by Christen (2006), Ernst-Gerlach (2013) and Kempken (2005). Naturally, each algorithm has its advantages and disadvantages when it comes to the comparison of strings in historical texts. Thus, the selection of the measures depends upon the task at hand. A combination of different methods is likely to be beneficial for normalization and NER in historical texts.

## 2.2 Named Entity Recognition

NER is an Information Extraction task and a "core component in many Natural Language Processing applications, including question answering (QA), summarization, and machine translation" (Nothman et al., 2013, 151). Its goal, as stated before, is to correctly recognize mentions of Named Entities within text. According to Benikova et al. (2014), the process of NER can be divided into two sub-tasks: first, relevant tokens belonging to a NE have to be found within the running text, which is referred to as Named Entity Detection. Second, the detected entities have to be classified according to a pre-defined set of semantic categories. This step is called Named Entity Classification. Technically, the sub-task of NE detection is a segmentation problem: since names are defined to be sequences of tokens, they may also be nested (e.g. *Zurich airport* is a NE containing the NE *Zurich*). In that case, the detection of nested NEs becomes formally similar to chunking. There are different ways of representing nested entities, including hierarchical visualization or bracket notation. A very widespread scheme is the BIO-notation<sup>3</sup>: tokens marking the beginning of a NE are tagged as B ('beginning'), while subsequent tokens of the NE are tagged I ('inside'). Tokens not belonging to a NE are tagged O ('outside') (Indurkha and Damerau, 2010, 523). Figure 2 shows the Latin nested NE *eclesiam Sancti Martini Rigniacencis* in a tree-like visualization created with *brat*<sup>4</sup> (a), in bracket notation (b) as well as in BIO-notation (c). Other ways of representing interleaved NEs, even

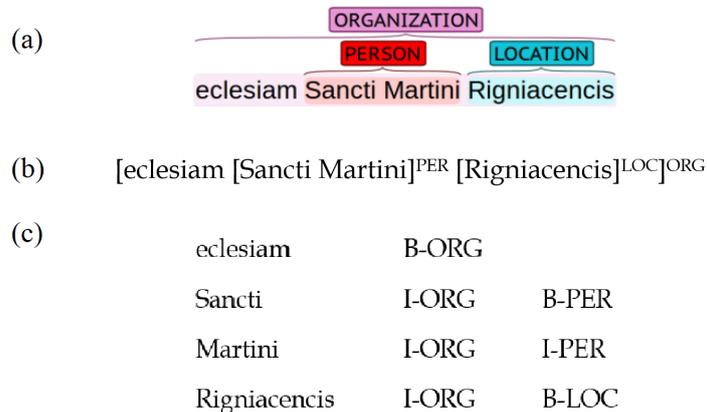


Figure 2: Different forms of representing the same nested NE based on an example given by (Aguilar et al., 2016, 69)

though less prevalent, have been suggested. For instance, Byrne (2007) developed her own multi-word token method for nested NER in historical archive texts.

<sup>3</sup>The BIO tagging scheme, also referred to as IOB, is frequently used in NLP chunking tasks.

<sup>4</sup>*brat* is an online environment for collaborative text annotation, cf. <https://brat.nlpplab.org/index.html>

NEs are usually defined as linguistic expressions, i.e. single tokens or phrases, referring to names of individual entities in the real world. These entities may be persons, locations or organizations, but also specific dates or measuring units. Most of the current NER systems have focused on three to four classes of NE. For instance, both the monolingual NoSta-D system (Benikova et al., 2014) and the multilingual *Stanford NER* tagger (Manning et al. (2014), cf. section 5.1.1) differentiate between person, location, organization and a fourth class 'miscellaneous' reserved for other detected entities<sup>5</sup>. Nothman et al. (2013), on the other hand, have further divided each of these four classes into sub-classes. Table 2 illustrates their classification with an example NE for each type.

Coarse-grained NE type	Fine-grained NE type	Example
Location	Town/city	<i>Bangkok</i>
	Facility	<i>Beijing National Stadium</i>
	Other	<i>Great Wall of China</i>
Organization	Band	<i>Blink-182</i>
	Corporation	<i>Logitech</i>
	Other	<i>Manchester A's</i>
Person	Person	<i>John F. Kennedy</i>
	Other	<i>Yoda</i>
Miscellaneous	Event	<i>South Ossetia War</i>
	WorkOfArt	<i>Entourage (TV series)</i>
	Product	<i>AK-47</i>
	Other	<i>Capoeira</i>

Table 2: Coarse-grained and fine-grained NE types with examples, adapted from (Nothman et al., 2013, 159)

Depending on the framework of the application, the NE labeling may be even more fine-grained. For instance, Ratinov and Roth (2009) have presented a hierarchy containing 18 classes of NE, while the set by Sekine et al. (2002) even uses more than 150 labels. The definition of the classes and corresponding NEs is not always trivial. For instance, while the noun *Italy* belongs to the NE category 'location', adjectives such as *Italian* may or may not be included. Likewise, the temporal expression *Christmas 2017* is considered a better example of a NE rather than e.g. *July*, as the latter is the month of an undefined year.

<sup>5</sup>The *Stanford NER* models may also be modified to differentiate between three classes (person, location, organization) or seven classes (person, location, organization, money, percent, date, time).

Furthermore, NEs tend to be ambiguous. (Carstensen et al., 2010, 598) mention two cases of ambiguity, namely polysemy and synonymy. Polysemy means that the same name refers to different entities, e.g. the term *Winchester* may refer both to a place in England or in the United States, but it can also be the name of a product, a person, and so on. Synonymy, on the other hand, occurs when the same entity has different names. This also includes cases where an entity may be formulated in different ways, e.g. *Barack Obama*, *Obama*, *B.O.*, *Mr. President*, and so on. The disambiguation of NEs, i.e. "associating each classified text span with a referent in the world or some model thereof" (Leidner et al., 2003, 31) is also referred to as grounding. Resolving NEs in a text usually involves mapping them to a unique identifier in a database.

Given the high frequency and complexity of NEs, the current state-of-the-art NER system is based on data-driven methods rather than on handcrafted rules. (Nothman et al., 2013, 151) explain:

With a substantial amount of annotated data and a strong evaluation methodology in place, the focus of research in this area has almost entirely been on developing language-independent systems that learn statistical models for NER. The competing systems extract terms and patterns indicative of particular NE types, making use of many types of contextual, orthographic, linguistic and external evidence.

The learning approaches and features used for recognizing NEs are manifold. However, the benefits of combining different methods have already become apparent in early experiments, such as the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003). Participants of this task made use of lexical and character-based features, POS-tags, chunking, as well as orthographic information. Moreover, the incorporation of external resources such as gazetteers, affix lists, and additional NER systems improved performance. The best results have been obtained by combining Maximum Entropy Models, transformation-based learning, and Hidden Markov Models (Tjong Kim Sang and De Meulder, 2003, 146).

Of course, acquiring large amounts of human-annotated data can be highly cost- and time-consuming. In order to minimize the annotation effort, semisupervised approaches have been suggested (Lin and Wu (2009), Nothman et al. (2013)). The latest NER systems have focused on applying Neural Networks (Santos and Guimarães (2015), Lample et al. (2016)). For English, these produce near-human performance with F-scores of around 93 %.

## 2.3 NER in historical texts

There has been a number of previous studies related to NER in historical texts. According to the methods and tools used in these studies, they can be roughly divided into rule-based and data-driven.

### 2.3.1 Rule-based approaches

Grover et al. (2008) described an approach at recognizing person and place names in digitized records of British parliamentary proceedings<sup>6</sup>. With the ultimate goal of implementing an online NE search engine, they developed a pipeline which includes OCR, annotation, as well as visualization of the data. Generally, this pipeline built on rule-based techniques accessing external resources, whereas ML techniques were deliberately avoided. This decision was made due to the frequent capitalization of common nouns in the documents, and the absence of a POS-tagger suitable for historical English texts. The data used in this study included two sets from the Journals of the House of Lords, one originating from an earlier period (1685 - 1691) than the other (1814 - 1817).

Two different systems were applied for OCR (one for each of the two data sets). Despite good quality of the scanned images, the OCR output was reported to be erratic. Text properties which caused problems for the OCR process include 'f'-like realization of the letter 's', quotation marks at the beginning of lines, marginal notes, special characters, and Latin text segments (Grover et al., 2008, 1343). Figure 3 displays a number of tokens in which the OCR software failed to recognize the correct characters. Many of the errors could be attributed to the presence of ligatures (i.e. the binding together of two or more letters).

<i>βands</i> (stands)	<i>feve-rälly</i> (seve-rally)
<i>Erβine</i> (Erskine)	<i>haYe</i> (have)
<i>Ap↯ peals</i> (Appeals)	<i>refpecT:</i> (respect)
<i>Lord↯ fhips</i> (Lordships)	<i>0 ' Done 1</i> (O'Donel)

Figure 3: OCR errors in Parliamentary records data, taken from Grover et al. (2008)

<sup>6</sup>Their study was conducted within the framework of the 18th century parliamentary papers digitization project (BOPCRIS). BOPCRIS delivered over 1 million pages from all surviving 18th century Parliamentary documents, cf. <https://www.bopcris.ac.uk>.

In order to create a gold-standard, 137 OCRed files corresponding to 188'547 tokens were manually annotated with the three entity types person, location and 'interrupt'. The third type was used to mark material inside a NE but which was not part of it (originating from OCR errors). Ultimately, 5'950 person names and 2'185 location names were annotated with an inter-annotator agreement of 91.46 % (Grover et al., 2008, 1344).

The main component of the pipeline was built to tag person and place names by accessing rules defined in grammar files. Amongst others, there were rules for monarchs, lords, earldoms, counties, etc., which access a variety of lexicons and name lists. General purpose name grammars operated to find high confidence names such as *Mr. Stratford* or *Town of London*. Moreover, a grammar to identify and eliminate noise (e.g. marginal notes or non-alphabetic symbols) was applied. The pipeline generated a file with stand-off annotation<sup>7</sup> file in XML format with additional markup for the entities as well as for OCR coordinate information. Since the resulting NE-tagger should be able to display visual output to the user, this standoff output format "was designed to enable highlighting of entities in the page image through a process of mapping from the regions in the OCR XML to the regions in the image" (Grover et al., 2008, 1345).

In the evaluation step, the authors reported average F-scores of around 71 % for both the 17th and 19th century data sets, but stated that the scores for person names were much higher than those for locations (66.53 % vs. 75.60 %). This reflects the fact that person names were more resistant to OCR errors and less dependent on the gazetteer resources<sup>8</sup>. Furthermore, precision was generally higher than recall (e.g. 74.34 % vs. 69.44 % in the 19th century data set). Error analysis demonstrated that bad paragraph breaks and other errors by the OCR process were responsible for this outcome (Grover et al., 2008, 1346).

In spite of various problems (mainly resulting from OCR errors), the prototype system by Grover et al. (2008) nicely exemplified technical challenges and possibilities (regarding annotation, the use of external linguistic knowledge, and visualization techniques) when working with historical data. In future work, the pipeline could be improved by correcting the OCR output, further developing the rules and lexicons, and by incorporating ML techniques into the process.

---

<sup>7</sup>Stand-off (or remote, offline) annotation means that the markup resides in a different location than the data. As opposed to inline annotation, the markup can be accessed separately. Cf. Burghardt and Wolff (2009).

<sup>8</sup>For example, the erroneous string 'Earl of Shagefiury' (*Earl of Shaftesbury*) was still correctly identified as a person name by the 'Earl of XY' pattern matching rule, whereas the place name lookup failed.

Other rule-based approaches relying on handcrafted rules and external resources were proposed, for instance, by Volk et al. (2009). Their method made use of a so-called "learn-apply-forget"-algorithm<sup>9</sup> for NER in the alpine heritage corpus *Text+Berg*<sup>10</sup>. This algorithm accesses large lists of person names and titles in order to save (or "learn") surnames in the text. The saved surname will be recognized in a specified number of subsequent sentences, up until the point where it will be "forgotten" and has to be reintroduced. In addition to that, gazetteer lists for country and city names were used in combination with linguistic processing (lemmatization, decomposing) to recognize location entities.

There have also been more recent approaches which did not directly deal with NER, but nevertheless provided promising solutions regarding IE for historical data. For example, Scrivner and Kübler (2015) proposed a way of exploiting word alignments in an Old Occitan-English parallel corpus for QA. The corpus contains Medieval provençal literature from the 13th century together with modern English translations, while the English side is enriched with POS-tags as well as morpho-syntactic and semantic information. By applying cross-language transfer rules from English to Old Occitan, the historical part of the corpus was automatically annotated and pre-processed for sentiment analysis. Given a parallel corpus with historical/modern word alignments, the methodology described by Scrivner and Kübler (2015) may as well be considered for the development of NER systems.

### 2.3.2 Data-driven approaches

A quite recent data-driven study by Neudecker (2016) looked into the subject of NER in digitized historical newspapers as part of the *Europeana Newspaper* project<sup>11</sup>. In contrast to the rule-based methodologies outlined above, the framework described in this approach was predicated on training *Stanford* CRF models using manually annotated data. *Europeana Newspapers'* entire collection comprised documents from four centuries (1618 - 1990) in more than 40 different languages. Neudecker (2016) focused on a subset with news articles in German, Dutch and French. In a first step, 100 pages containing 486'218 tokens were selected according to a number of specifications (e.g. at least 80 % OCR word accuracy per text). As a next step, all pages were manually annotated with the NE entities person, location and organization,

---

<sup>9</sup>Cf. Volk and Clematide (2001).

<sup>10</sup><https://www.textberg.ch>

<sup>11</sup>*Europeana Newspapers* is a collaborative research and digitization project co-funded by the European Commission "to aggregate full text and scanned images of approximately 12 million pages of historic European newspapers" (Neudecker, 2016, 4348). Cf. <http://www.europeana-newspapers.eu/>.

resulting in 40'801 tagged entities. This annotated material was then used to train a Stanford NER classifier for each language which was subsequently evaluated in a 4-fold cross-validation<sup>12</sup>.

While the authors did not report results for the German data, precision scores for Dutch and French were claimed to be in the 80-90 % range and recall at around 60-70 % (Neudecker, 2016, 4351). This slightly "conservative" bias of the *Stanford* system was considered to be beneficial for the further processing of the data, where the recognized NEs are supposed to be disambiguated and linked to unique identifiers (i.e. grounding). In other words, the mapping of entities to resource descriptions benefits from precision more than from recall. It is expected that recall improves with the inclusion of distributional similarity or semantic generalization in the training, as well as by enlarging the size of the training data (Neudecker, 2016, 4351).

Another methodologically similar study from the same year was outlined by Aguilar et al. (2016). Their work focused on NER in historical sources from the 10th to the 13th century written in Medio-Latin language, using an annotated database from the CBMA<sup>13</sup> group. This database includes cartularies, i.e. medieval manuscript volumes with transcriptions of legal documents, and charters produced in abbeys of Burgundy. Annotations of NEs were made in 5'300 documents of the corpus using the NE types person and location. In order to find the most significant features for training a NER classifier, the authors closely analyzed the morphology of medieval Latin. They noted, for instance, that personal names usually have the endings '-o', '-us' and '-um'. Moreover, there are indicator words such as *villa*, *castrum* and *mansum* characteristic of place names, titles such as *sanctus*, *rex* or *dominus* which introduce person names, and finally some other "trigger" words (e.g. specific preposition and affixes). (Aguilar et al., 2016, 69) also isolated linguistic information from the annotated database, including POS-tags, lemma, suffix and capitalization.

The training data contained more than 1 million words and was transformed into a 7-column format for training with an out-of-the-box CRF toolkit. Moreover, BIO-tags were used in order to deal with nested entities (cf. Figure 2 above). By using 26 different uni-, bi- and trigram patterns, (Aguilar et al., 2016, 70) obtained F-scores of 96 % for person names in beginning of entities (B-PER), and 88 % inside entities (I-PER). Similarly, recognizing the beginning of place names seemed to be easier

---

<sup>12</sup>Unfortunately, it is not declared in the paper which features were used for training the model. I therefore assume that at least some standard CRF features such as character and token n-grams were included.

<sup>13</sup>CBMA stands for *Chartae Burgundiae Medii Aevi*, an open access database of diplomatic sources produced in the Middle Ages in Burgundy. Cf. <http://www.cbma-project.eu/>.

than recognizing the rest of them (92 % F-score for B-LOC vs. 80 % for I-LOC). The high quality was attributed to the formulaic nature of the documents which was relatively easy to represent with patterns, while most of the errors were related to the presence of long, nested entities (which were also inconsistently annotated in the training data). (Aguilar et al., 2016, 71) claimed that future work would involve verifying the robustness of the model by testing on different types of Latin documents from different time periods. In addition, the long-term goal was to be able to recognize more than two NE types.

In summary, studies dealing with NER in historical data have shown that systems based on handcrafted rules or external resources (such as lists or gazetteers) tend to be the most straightforward to implement; they also yield good precision and recall. On the other hand, these systems are usually very project-specific and not generic enough to be reused without further modifications. Rule-based systems require professional knowledge of the historical language variety as well as a certain degree of NLP expertise. ML systems do not build on as much linguistic proficiency, but for satisfying results, big amounts of (manually) annotated training material are necessary.

# 3 Text Pre-processing and Corpus Creation

This chapter introduces the data from the SSRQ database upon which my project was based. It also describes the pre-processing of the selected texts and the subsequent transformation into structured corpus files with XML markup.

## 3.1 Data

### 3.1.1 The *Collection of Swiss Law Sources*

The *Collection of Swiss Law Sources* (SSRQ) was established in 1898 by the *Swiss Law Sources Foundation*, a research institution belonging to the *Swiss Lawyers Society*<sup>1</sup>. The source material in the collection was created in Switzerland (or on territory now belonging to Switzerland), and includes documents from all its linguistic regions. Thus, the texts are written in regional historical forms of German, French, Italian, Rhaeto-Romanic, as well as Latin. The documents edited by the foundation date back to the early Middle Ages and cover a time-span up to modern times (to 1789, the year which marked the downfall of Switzerland's *ancien régime* (Höfler and Piotrowski, 2011, 78).). More than 100 volumes corresponding to over 70'000 pages of source material and comments have been published so far. Amongst other types of documents, the collection contains decrees, acts, ordinances, indentures, administrative documents, and court transcripts (Piotrowski and Senn, 2012, 25). The sources are organized by cantons and further subdivided into areas of jurisdiction. Most volumes are assigned to different sections and series which refer to particular topics; e.g. the volumes of the canton of Fribourg are divided into the two sections *Stadtrechte* (municipal laws) and *Rechte der Landschaft* (land laws), while the first section is further split into two series (rural towns, town of Fribourg). Each volume contains an index of places and persons as well as a glossary.

---

<sup>1</sup>Schweizerischer Juristenverein, <https://www.juristenverein.ch/>.

The primary sources are manuscripts which have been transcribed, annotated, and commented by different editors with expert knowledge. Manual transcription and editing has been performed following pre-defined guidelines<sup>2</sup> and using specific publishing systems. Each volume contains a number of title pages, a lead text, the edited main part (i.e. source texts in chronological order), indices and an appendix. The critical apparatuses are written in modern German, French, or Italian, depending on the predominant historical language of the volume. Figure 4 shows a primary source, i.e. a scanned manuscript excerpt of a volume together with the corresponding transcription. After transcription and editing, the volumes have been printed in the form of books. This project is still on-going, and further volumes are currently being transcribed.

Erstlich hatt wolfgang lippuner landtweibel zu Sarganß  
 vor einem er samen gericht zu Sarganß herren Jacoben, caplonen zu Sarganß, anclagt,  
 das er ime am palms abenndt sin meitli altem bruch nach geschickt zu  
 bychten. Inn derselbigen bycht habe er, herr Jacob, zum meitli gredt, es  
 steli ime das höw, die eyer und das kruth unnd derglychen ungschickte  
 frag stuck mit ime, meitli, triben. Weliches aber sich, ob gottwil, nit  
 erfinden werde unnd vermeine derhalben er, herr Jacob, sölle söliches  
 darbringen oder in des meitlis fußstapffen gestellt werden. Darnebendt  
 er, landtweibel, ouch ettliche wort fallen lassen vor offnem gericht, nam-  
 lich: «Das ouch gotz hie und dört als feldziechen, fasels oder pfaffen  
 schend, wottennd ir mir mine kind zu dieben machen.» etc.

(a) Primary source: A 16<sup>th</sup> century record of a defamation case (Early New High German; StALU, A1 F1, Sch. 395, Mappe Pfarrei Sargans).

Erstlich hatt Wolfgang Lippuner, landtweibel zu Sarganß, vor einem er samen gericht zu Sarganß herren Jacoben, caplonen zu Sarganß, anclagt, das er ime am palms abenndt sin meitli altem bruch nach geschickt zu bychten. Inn derselbigen bycht habe er, herr Jacob, zum meitli gredt, es steli ime das höw, die eyer und das kruth unnd derglychen ungschickte frag stuck mit ime, meitli, triben. Weliches aber sich, ob gottwil, nit erfinden werde unnd vermeine derhalben er, herr Jacob, sölle söliches darbringen oder in des meitlis fußstapffen gestellt werden. Darnebendt er, landtweibel, ouch ettliche wort fallen lassen vor offnem gericht, namlich: «Das ouch gotz hie und dört als feldziechen, fasels oder pfaffen schend, wottennd ir mir mine kind zu dieben machen.» etc.

(b) Transcription

Figure 4: Primary source (a) from the SSRQ SG III/2, No. 113 with manual transcription (b) according to guidelines. Taken from (Piotrowski, 2010, 223)

<sup>2</sup>They are available on the SSRQ Wiki, cf. [https://www.ssrq-sds-fds.ch/typo3/fileadmin/user\\_upload/Editionsrichtlinien/SSRQEditionsrichtlinien.pdf](https://www.ssrq-sds-fds.ch/typo3/fileadmin/user_upload/Editionsrichtlinien/SSRQEditionsrichtlinien.pdf) and <https://www.ssrq-sds-fds.ch/wiki/Transkriptionsrichtlinien>.

### 3.1.2 The SSRQ digitization project

Within the framework of a retro-digitization project from 2009 to 2011, all edited volumes were scanned, OCR-processed and post-edited so that the facsimiles can be accessed on the web via an online viewer. This enables users to search, browse and navigate according to cantons, dates or keywords. Fortunately, the digital typesetting data for the latest 22 volumes were restored in the form of *Adobe FrameMaker*<sup>3</sup> 3.0 and 6.0 files (Höfler and Piotrowski, 2011, 78). The advantage of these files is that the texts are identical to the contents of the printed books, and thus free from OCR- or rekeying errors. However, since the *FrameMaker* files were intended for printing, they merely contain visually oriented markup. In order to make them more usable for research purposes, Piotrowski (2010) automatically converted the files into valid XHTML format with cascading stylesheets (CSS) closely mirroring the layout of the *FrameMaker* documents. In this process, the bodies of the volumes were transformed into TEI<sup>4</sup>, whereas the indices were converted into "an XML format that makes the[ir] logical structure [...] explicit, i.e., headwords, glosses, spelling variants, page and line references, etc." (Piotrowski and Senn, 2012, 26). The converted files of the 22 volumes were then added to the text collection referred to as DS21 (Höfler and Piotrowski, 2011, 79).

According to Piotrowski (2010), the conversion from *FrameMaker* files to XHTML was not a trivial task. First of all, the files were written in different versions of the software with very limited backward compatibility. Second, the paragraph and character formats did not always follow a logical pattern and were sometimes locally overridden to produce a specific visual appearance. Third, a direct conversion was not feasible but instead required interposing a transitional format. For these reasons, a custom-built conversion pipeline operated in multiple stages with the goal to maintain the original formatting by producing equivalent CSS and XHTML markup. In addition, certain tags were replaced or added in order to make the markup regularized and semantically meaningful (Piotrowski, 2010, 226). Figure 5 shows an example text taken from an edited volume: (a) is a scan from the book, i.e. as it appears in the printed version, and (b) is the corresponding XHTML format resulting from the conversion of the *FrameMaker* file.

---

<sup>3</sup>*Adobe FrameMaker* is a text processing software for creating, editing and publishing of print-oriented documents. Its advantage lies within the extensive functionality even when working with large, complex and structured technical texts. Moreover, it allows authors to enrich documents with (XML-)markup ((Kornelsen et al., 2004, 37), (Piotrowski, 2010, 224).

<sup>4</sup>TEI stands for *Text Encoding Initiative* which has developed guidelines on how to use XML markup to structure documents. The format has become a standard in many fields of the humanities. Cf. <https://www.tei-c.org>.

## 2. Ergänzende Satzungen zur Handfeste A

1341 Dezember 5 – 1381 Juni 18

### a. Freiheits- und Körperverletzung bei Burgern, verübt durch Nichtburger außerhalb des Grabens

20

1341 Dezember 5 (an sant Niclaus abent)

Sch, RuB von Burgdorf urkunden, dz wir willeklich, einhelleklich durch unser stat nutz, ere und notdurft mit [...] graf Eberhartz von Kyburg, lantgraven ze Bùrgenden, gunst, heissen und willen uff gesetzet haben und setzen mit dirre geschrift also mit namen:

25

### (a) Excerpt from a scanned volume

```

<p class="Ti_2" id="tf4p291">2. Ergänzende Satzungen zur Handfeste A</p>
<p class="Ti_4_2" id="tf4p292">1341 Dezember 5 – 1381 Juni 18</p>
▼<p class="Ti_4_3" id="tf4p293">
  a. Freiheits- und Körperverletzung bei Burgern, verübt durch Nichtburger auß
  ▼<span id="tf4p293f341">
    ▼<span class="aframe_Left" id="aframe_174">
      <span id="af174t11f1">20</span>
    </span>
  </span>
  <span id="tf4p293f342">erhalb des Grabens</span>
</p>
▼<p class="Ti_4_1" id="tf4p294">
  1341 Dezember 5
  <span id="tf4p294f343">(an sant Niclaus abent)</span>
</p>
▼<p class="norm_2" id="tf4p295">
  <span id="tf4p295f344">Sch, RuB von Burgdorf urkunden,</span>
  ▼<span id="tf4p295f345">
    dz wir willeklich, einhelleklich durch unser stat nutz, ere und notdurft mit [...]
    geschrift also mit nam
    ▼<span class="aframe_Left" id="aframe_175">
      <span id="af175t11f1">25</span>
    </span>
  </span>
  en:

```

### (b) Corresponding XHTML excerpt

Figure 5: Excerpt from the edited volume BE II/9 of the canton of Berne (reproduced by (Rechtsquellenstiftung des Schweizerischen Juristenverbandes, 1995, 24)) in printed (a) and XHTML (b) format. The excerpt describes a charter regarding the personal injury to citizens of Burgdorf.

### 3.1.3 Selection of volumes

As (Höfler and Piotrowski, 2011, 77) claim, legal texts are a fruitful object of study for both historians and linguists. The texts do not only "represent a crucial source of information on the distribution of power in societies past and present"; they also "constitute a special case of highly conventionalized language", giving insights which are relevant for diachronic linguistic studies. Scholars in the humanities greatly benefit from structured corpora which provide them with additional means to study the domain. Regarding historical legal texts from Swiss sources, the DS21 collection by Höfler and Piotrowski (2011) clearly constitutes an important step towards this aim. However, as their work is on-going, there is still a lack of linguistic annotations or general semantic markup, and a redundancy of other markup in the XHTML files. Following the definition by (Auer and Fairman, 2013, 79), the DS21 collection is – strictly speaking – not a corpus (yet), but a well-designed text archive:

While the creation of many corpora is indeed based on a detailed plan that tries to consider the above-mentioned criteria [balance, functionality and representativeness], the starting point [...] is an existing collection of texts representing a particular language/variety/genre.

I have decided to work with texts of the DS21 in order to bring its contents a little step further to being a corpus that can be used for linguistic and historical research purposes. More precisely, the goal was to create an alternative representation of a small subset of the collection, while keeping the main task of NER in mind. Another reason for choosing the DS21 texts was that I could build upon previous work, and benefit from the advantages of the already converted XHTML files.

The decision to place my approach into a multilingual context was due the fact that the volumes are written in four different languages, including Latin. However, the majority of the documents are in German (all volumes except for three contain German material). Volumes with Latin only do not exist, and Italian can be found in merely one volume (which is also the smallest). I therefore opted for working with two languages, German and French, as three relatively large French volumes exist in the DS21 collection. Moreover, I preferred having an equal number of volumes in both languages. Thus, the basis for the work described in this thesis were six volumes of the collection. Preliminary word counts indicated a total number of at least one million tokens, which should also meet the requirements for building a ML system<sup>5</sup>. Table 3 lists the volumes I selected for the thesis. The languages given in the table refer to their historical variants from the time periods indicated.

---

<sup>5</sup>In an earlier seminar project, we successfully trained a CRF model based on a single volume.

Volume ID	Volume No.	Canton	Primary language(s)	Period covered	Total pages	Section /series ID
AG II/10	10	Aargau	German	1303–1797	735	2
BE II/9	9	Berne	German	1267–1797	735	2
LU I/2	2	Lucerne	German	1426–1463	481	1
FR I/2.6	6	Fribourg	French, German	1296–1795	582	1 / 2
VD C/1	1	Vaud	French, German	1530–1797	971	C
VD C/2	2	Vaud	French	1539–1770	925	C

Table 3: Selected German and French volumes of the DS21 collection

The selection comprises all available volumes with French content, except for one (VD B/2, 2nd volume from section B of the canton of Vaud) in which the majority of texts are written in Latin. I chose the three German volumes semi-randomly based on the degree of complexity or "corruption" of the XHTML files. In other words, if manual inspection of these files showed that highly extensive pre-processing and reordering would be necessary, I did not include the volume. I discharged, for example, the XHTML files of SG I/2\_3 (3rd volume from section 1 of the canton of St. Gall) due to technical issues with pagination<sup>6</sup>.

Each volume comes with a separate index of place and person names in XHTML format, as well as a PDF file representing the appearance of the printed book. The converted files of the volume only contain the edited main body, i.e. excluding title pages, preface and appendix. The content of the main body is split into several XHTML files, one of them usually encompassing 30 to 60 pages. The goal was to process and combine these files in such a way that the output per volume is a single, well-formed XML file. In the ideal case, the markup should follow a logical structure where every page, paragraph, sentence and token is enumerated, enclosed in separate tags. Exemplary models of such structured XML files are the corpora developed for the *Text+Berg digital* project (Bubenhofner et al. (2015), Volk et al. (2010)). The markup follows a hierarchical layering of elements (from book, article, sentence, to word level). XML attributes contain information regarding the elements, e.g. language and number of sentences, and lemma and POS-tag of tokens.

<sup>6</sup>The first XHTML file started with page 26 instead of page 1, followed by pages 29-31, then 27 and started again at page 1. Finally at the end of the file, after page 54, there were contents of page 25. A likely explanation for this misordering could be the presence of tabular elements in the *FrameMaker* data, appearing as long juxtaposed lists or tables in the printed version.

## 3.2 Pre-processing

### 3.2.1 XHTML parsing

The desired output format of each processed volume was an XML file similar to the ones built in the *Text+Berg* project. The first step was thus the syntactical analysis of the XHTML files, so that their elements could be accessed and restructured. For this end, I implemented a pre-processing script in **Python**, and used the *ElementTree/lxml* module<sup>7</sup> for parsing. As can be seen in figure 5, the principal component in the XHTML file is the paragraph element `<p>`. There, it is not only used for "typical" paragraphs (i.e. bigger bodies of running text), but for all units appearing as text blocks in the printed version, including titles, subtitles, footnotes, comments, and so on. The textual function of each text block is indicated by the `class` attribute. For instance, the attribute `class='fuss'` means that the current text block (i.e. paragraph) is a footnote. I decided to retain the class attributes so that the paragraph function would remain apparent in the final XML file.

Unfortunately, the naming of the classes is neither very consistent nor transparent, and is different for every volume. While class names such as `haupttext` are relatively evident, others do not give insight into the text function, e.g. `bu_ti_1`, `be`, `standard`, `sta_text`, `li`, or `norm`. Only by manual inspection, i.e. by carefully comparing the XHTML with the scanned version, I was able to identify the function denoted by each class<sup>8</sup>. Ultimately, I found more than 60 different names for class attributes in all volumes. Around half of them referred to titles, and at least 12 were rather cryptic, consisting of digits or underscores only. I therefore decided to discard the naming as it was in the XHTML, and create an own set of class names instead. I merged all attributes and mapped them accordingly to the seven classes `title`, `date`, `text`, `comment`, `footnote`, `source` and `table`. I believe this to be a reasonable generalization as it distinguishes between source text and editor's notes; at the same time, it maintains the differentiation between the most important kinds of annotations (commentaries vs. description of the source vs. footnotes).

I additionally extracted and retained the page numbers that were given as `pagemarker` class attributes. However, I decided not to store paragraph IDs (given in the `id` attribute), since they did not seem to be semantically meaningful.

---

<sup>7</sup>A **Python** package for reading, generating, and modifying XML files, with support for *XPath*. Cf. <https://lxml.de/index.html>.

<sup>8</sup>For instance, a volume chapter always starts with a main title, followed by a date and optional headings before the source text begins. These titles usually have a class name starting with `ti` in the XHTML.

## 3.2.2 Restoring pagination

### 3.2.2.1 Problem

When trying to parse the files and extract all page numbers, I realized that the pagination in the XHTML was not always continuous. This was the case for the volumes LU I/2 and VD C/2, and was caused by the presence of tables spanning more than one page. For example, at the end of page 117 in the printed version of LU I/2, the text is indented and divided into a table-like arrangement with two columns. This arrangement continues until the beginning of page 125, and is represented by `<table>` elements in the XHTML format. However, `<table>` elements do not contain any `pagemaker` attributes, which makes it impossible to recognize the page breaks within the table. In the case of LU II/2, this means that the contents between page 117 and 125 were lumped together to a single block, as illustrated by the following simplified snippet:

```
<a class="pagemaker" id="page_117">117</a>...  
<table id="tbl_330">  
<tr><td><div>&#x2205;</div><div>Anno domini...</div></tr>  
</table>  
<a class="pagemaker" id="page_125">125</a>...
```

One option to restore the original pagination would be to modify the XHTML by manually inserting the missing `pagemaker` attributes. This would require comparing the pages in the printed format to the XHTML, and searching the whole `<table>` block for the positions of the line breaks. However, I opted for a less cumbersome and more automated possibility, which is based on OCR-processing the PDF files<sup>9</sup>. Obtaining a plain text version of the relevant pages makes them directly comparable to the text in the `<table>` block.

By having a separate OCRed file for each page, the original pagination remains apparent. The general idea is that, naturally, the very last text segment of an OCR output file marks the end of the page. If this segment can be located within the XHTML block, the page break must be inserted after it. The missing `pagemaker` tag can then automatically be added at the correct position. Generally, such a text segment may be a single token or of a sequence of tokens. However, a single token is not distinctive enough, as it may occur at numerous positions in the text. I therefore defined a segment to be a sequence of exactly 10 tokens.

---

<sup>9</sup>I also preferred an automated option because of its repeatability (in case other volumes need to be processed in the future).

### 3.2.2.2 OCR approach

The OCR tool of my choice for this purpose was *PyOCR*<sup>10</sup>, a **Python** wrapper for Google’s *Tesseract*<sup>11</sup> engine. After some experiments with other tools such as *Abbyy FineReader*<sup>12</sup> and *Ocular*<sup>13</sup>, I chose *PyOCR* as it is easy to use, open-source, and implemented in **Python**. Moreover, the underlying *Tesseract* engine is language-independent; it operates with segmentation- and character-based algorithms rather than language models (Smith et al., 2009). It also has UTF-8 support and is able to recognize more than 100 languages ”out-of-the-box” (Smith, 2013). I therefore considered *PyOCR* to be suited for processing historical and multilingual texts. Since the *PyOCR* module only accepts input in image format, I first converted the relevant pages of the PDF into PNG files<sup>14</sup>.

I added a **pagemarker** counter to the XHTML parser in order to keep track of the page numbers. As soon as a `<table>` element can be located that interrupts continuous pagination, an OCR comparison function is called. This function first extracts the last 10 tokens per OCRred page. For each page, the function then progressively splits sequences of 10 tokens from the `<table>` text, and compares them to the OCR sequence. As soon as the OCR sequence can be located in the `<table>` text, the page break is inserted, and the function continues splitting up the remaining text. This is repeated until all page breaks are inserted.

However, a one-to-one comparison of the text sequences does not make much sense. It is important to note that OCR-processing does not always yield flawless output; despite the high image quality, *PyOCR* did not correctly recognize all characters (cf. section 3.2.2.3 below). The text of the OCRred files will hardly ever be identical to the one in the XHTML. Therefore, a certain degree of derivation should be accepted. I decided to use **Python**’s *SequenceMatcher*<sup>15</sup> module to find the best matching text segment. This module measures the similarity ratio of the sequences and returns a value between 0 and 1. After a phase of testing, I found that a similarity ratio over 0.6 is enough to consider two sequences to be matches. In this manner, I was able to reconstruct all missing page breaks, and restore the original pagination.

---

<sup>10</sup><https://github.com/jflesch/pyocr>

<sup>11</sup><https://github.com/tesseract-ocr/tesseract>

<sup>12</sup><https://www.abbyy.com/en-eu/finereader/>

<sup>13</sup><https://github.com/tberg12/ocular>

<sup>14</sup>I used the conversion software *PDF Shaper* (<https://www.pdfshaper.com/>, generating one image per page with a resolution of 300 dpi (dots per inch; as suggested by (Holley, 2009, 3)).

<sup>15</sup>*SequenceMatcher* is a class belonging to the **Python** *diffib* library, used to compare pairs of sequences. The similarity ratio is calculated by doubling the number of matching characters, and then dividing by the total number of characters in both sequences. Cf. <https://docs.python.org/2/library/difflib.html>.

### 3.2.2.3 Evaluation

I have applied the OCR processing approach only for a number of pages in two volumes so far. However, I wanted to make sure that it can be reproduced for other volumes, in case their XHTML files are missing page breaks. On that account, I decided to conduct a small-scale evaluation of selected OCR-processed files in order to estimate the general performance. I picked 14 files for evaluation, i.e. the first nine pages of LU I/2, and the first five of VD C/2<sup>16</sup>. In order to obtain the corresponding reference (ground-truth) files for comparison, the XHTML parser was used to strip all contents from the markup, so that only the plain text remained.

Standard metrics for evaluating OCR accuracy are the word error rate (WER) and character error rate (CER) (Farooq and Al-Onaizan, 2005, 105). They are defined as the *Levenshtein* distance (on word- and character level), divided by the number of characters and words respectively in the reference text. The position-independent word error rate (PER) additionally takes the word order into account (Ueffing and Ney, 2004, 73). An open-source tool particularly suited for calculating OCR error rates is *ocrevalUAtion*<sup>17</sup>. I chose to work with this tool mostly due to its visualizing capability, which is especially useful for difference spotting. Apart from a statistical report, *ocrevalUAtion* generates a table with the parallel input texts in which the differences are highlighted. Figure 6 shows the beginning of the table generated for the first page of LU I/2.

LU_I_2_orig.txt	LU_I_2_ocr.txt
<p>1. Stadtwechsel 1426 Januar 4, Juni 28 a)  Besetzung des Stadtwechsels 1426 Januar 4 Wir  haben den wechssel besetzt mit Peter Goltsmit  und Hei<del>r</del>. Hasfurter zen Barfü<del>s</del>sen. Dam<del>a</del>ch uff  hu<del>t</del>t haben wir jnen zii geben Ant<del>o</del>nien Russen,  Wem<del>h</del>er Keller und Germann. — Dis fü<del>n</del>ff  so<del>l</del>llent vollen gewalt han umb wechsslen, gelt  ufzenemen und all ander sachenn. StALU RP 4  fol. 88r (alt 59r): Eintrag des Heinrich Golz zu  feria sexta ante Epiphanyam = [1426] Januar 4.  Zitiert: von Segesser, Rechtsgeschichte 2,281 f.  Anm. 2. BEMERKUNGEN 1. Die Existenz des  Stadtwechsels ist bereits durch RQ LU 1/1 Nr. 43</p>	<p>1]. Stadtwechsel 1426 Januar 4, Juni 28 a)  Besetzung des Stadtwechsels 1426 Januar 4 Wir  haben den wechssel besetzt mit Peter Goltsmit und  Heim. Hasfurter zen Barfü<del>s</del>sen. Dam<del>a</del>ch uff hu<del>t</del>t  haben wir jnen zii geben Ant<del>o</del>nien Russen, Wem<del>h</del>er  Keller und Germann. — Dis fü<del>n</del>ff so<del>l</del>llent vollen  gewalt han umb wechsslen, gelt ufzenemen und all  ander sachenn. StALU RP 4 fo]. 88r (alt 59r):  Eintrag des Heinrich Golz zu feria sexta ame  Epiphanyam = [1426] Januar 4. Zitiert: von  Segesser, Rechtsgeschichte 2,281 f. Anm. 2.  BEMERKUNGEN [. Die Existenz des  Stadtwechsels ist bereits durch RQ LU 1/1 Nr 43</p>

Figure 6: Example of an OCR evaluation/comparison table with highlighted differences, generated via *ocrevalUAtion*

<sup>16</sup>The uneven distribution is because the pages in LU I/2 contain a lot more titles and footnotes, while the ones in VD C/2 contain more running text. However, I wanted to have about the same numbers of tokens for each volume, so they become more comparable.

<sup>17</sup><https://github.com/impactcentre/ocrevalUAtion>

	LU I/2	VD C/2
<b>Pages</b>	1-9	1-5
<b>Tokens</b>	3'167	2'857
<b>Characters</b>	18'054	17'763
<b>CER</b>	4.81	3.69
<b>WER</b>	18.22	9.16
<b>PER</b>	13.96	4.70

Table 3: Error rates in the OCR-processed pages

All 14 OCR-processed pages were compared to their corresponding plain text files by using *ocrevalUAtion*. The results are given in table 3. While the error rate on character level was similar in both volumes (4.81 vs. 3.69), it was considerably higher on word level in volume LU I/2 (18.22 vs. 9.16 WER, and 13.96 vs. 4.7 PER). The *ocrevalUAtion* statistics for LU I/2 revealed that a significant portion of falsely recognized characters were diacritics such as â, ô, ñ, w̃ or ũ. They are used by the editors for the literal transcription of Early New High German<sup>18</sup>, the variety in which the German volumes are written. Thus, the French volumes use diacritics to a much smaller extent. This made the OCR in French less error-prone, resulting in a smaller number of falsely recognized words.

Other sources of errors were certain letters or letter combinations such as 'nr' or 'ün', which were recognized as 'm' or 'im' respectively. As can be seen in figure 6, '1' and 'l' were sometimes recognized as square brackets. Even more problematic were special symbols: *ocrevalUAtion* frequently transformed them into different, but visually similar symbols (e.g. different forms of quotation marks and hyphens). While these kinds of errors were characteristic of OCR, others were rather document-specific. For instance, the text extracted from the XHTML files did not contain the page headline appearing in the PDFs<sup>19</sup>.

As mentioned before, *Tesseract* offers out-of-the-box packages for more than 100 languages and language varieties<sup>20</sup>. The results given in table 3 are based on OCR processing with the standard (i.e. modern) packages for German and French. For each of the two languages, there is one additional language data set available, namely *German Fraktur* and *French - Middle (ca. 1400-1600)*. For experimental reasons, I

<sup>18</sup>By definition, Early New High German belongs to the period 1350 to 1650 (Hartweg and Wegera, 2005, 23).

<sup>19</sup>The headlines indicate page number and number of the source text. While the page number is retained in the XHTML in the form of markup, the source number appears only in the printed version. This difference slightly increased error rates.

<sup>20</sup>Cf. <https://github.com/tesseract-ocr/tessdata>.

OCR-processed the texts again using these data sets, but this did not lead to any significant changes in the error rates. Further investigations will be necessary to analyze the features of the language packages, and to see how they could be used to improve OCR accuracy of the SSRQ texts.

Probably, the OCR accuracy could be further increased by different methods, e.g. by post-processing, rekeying or enhancing image quality. Training the OCR engine so that it is optimized for processing the SSRQ data may be another useful option. However, according to (Holley, 2009, 2), this is usually an "incredibly time-consuming" task. A better solution would be to work on a smaller scale, and improve the recognition only for the characters that are likely to cause errors. In fact, *Tesseract* can be configured by adding words and character patterns to the recognition dictionary. On that note, I would like to argue that for the purpose of page restoration – as outlined in this section – there is no need for maximized accuracy. Even with a relatively high WER, as it is the case for LU I/2, text segments can be successfully matched by applying an approximate string matching technique.

### 3.2.3 Cleanup

After the XHTML structure had been completely parsed and restored, I decided to perform a number of cleanup steps. As can be seen in figure 5, a very frequent element contained in the XHTML markup is the `<span>` element. Its original purpose was to indicate text formatting (e.g. specific font style and size), but in most cases, this information is lost. For instance, both the text segments appearing in bold and italic in the printed version are enclosed in `<span>`-tags, without any distinguishing attributes. In the case of certain numbers and letters, however, the `id` attribute indicates their function. I found that numbers in `<span>`-tags that have an attribute beginning with 'af' are line markers<sup>21</sup> (e.g. `<span id="af78t11f2">20</span>`). Footnote markers, i.e. reference numbers and letters, have an attribute beginning with 'tf'. In the printed version, they appear as superscript characters after a word, and refer to footnotes at the bottom of the page. Figure 7 shows a segment from the scan containing line and footnote markers (a), as well as the corresponding XHTML markup (b).

The resulting XML corpus, however, was not meant to include any information about formatting, but was intended to represent the tokenized text. While removing the XHTML markup resulted in the desired plain text, the line and footnote markers

---

<sup>21</sup>They are used for line enumeration in the printed version, and appear on the margins of a page in front of every fifth line.

- (a) Item so sol des Pfiffers jungfrow<sup>3</sup> xiiij guld. angolt minus iiiij pl. – Gewert vj guld. golt<sup>a</sup>. 20
- (b) `Jtem so sol des Pfiffers jun<span class="aframe_Far" id="aframe_10">`  
 `angolt minus iiiij pl. &#x2013; Gewert vj guld. golta</span>.`

Figure 7: Line 20 of LU I/2, page 1, in printed (a) and XHTML (b) format. Cf. (Rechtsquellenstiftung des Schweizerischen Juristenverbandes, 2004)

remained at their positions between tokens. Thus, I decided to remove all line and footnote markers for the following reasons: first, the distinction between markers and "normal" text will be lost in the XML corpus. The concerned numbers and letters will appear as any other token, which impairs reading fluency, and may lead to confusion. Second, the positioning of the line markers in the XHTML is only approximate, as illustrated in figure 7. Generally, the numbers may occur anywhere in the line, even between tokens. Third, I did not consider the markers to be part of the main content, as they merely serve an explanatory function. The footnote contents themselves will be retained.

Unfortunately, the `id` attribute beginning with 'tf' is not only used for footnote markers, but also for formatted letters which are part of the running text. Removing all elements with a 'tf'-attribute would result in numerous fragmented tokens and incomplete text segments. I therefore defined a footnote marker to be a number consisting of one or two digits. Consequently, I did not remove footnote markers consisting of letters only. Since the markup does not allow a straightforward way to distinguish page markers from other segments, I believe this solution to be a reasonable compromise. Furthermore, the line and footnote markers within `<table>`-elements needed special handling, as tables do not contain any `<span>`-elements. All markers, both numbers and letters, are directly appended to the preceding token. Only by writing rules with specific regular expressions I was able to strip the digits from the tokens. Again, the footnote markers in form of single letters could not be removed, since there is no direct way to recognize them.

The final cleanup step was the conversion of special symbols encoded in UTF-8. For instance, the symbol ł (latin barred l) has the hexadecimal code `&#322;` in XHTML, which is represented as `u"\u0142"` in **Python**. To make sure that all symbols will be correctly rendered, I used an external source<sup>22</sup> to build a conversion table which automatically converts XHTML codes into their **Python** representation.

<sup>22</sup>I downloaded and extracted the entities from the reference website *HTML Arrows*, cf. <https://www.toptal.com/designers/htmlarrows/letters/>.

### 3.2.4 Sentence boundary detection

Following the example of the *Text+Berg* XML, the resulting corpus is supposed to have enumeration on token and on sentence level. This requires sentence splitting and tokenization. A versatile platform which is able to do both is the *Natural Language Tool Kit* (NLTK)<sup>23</sup> implemented in **Python** (Bird et al., 2010). The NLTK sentence splitter has already been trained and tested on at least 10 European languages. For newspaper corpora, the splitter reached an F-score of around 98 % for German and 86 % for French (Kiss and Strunk, 2006, 504). Apparently, it is expected to perform worse for historical texts. A crucial fact is that the SSRQ data is multilingual: the volumes contain text in historical variants of German and French as well as Latin, while the comments and annotations by the editors are written in modern language. In addition, there is a large quantity of historical abbreviations, many of them being specific to a volume. Working with the standard abbreviation list of the NLTK sentence splitter (which contains 70 German and 60 French abbreviations) resulted in many unnecessary sentence splits. On account of this, I decided to extend the standard list by adding abbreviations that occur in the volumes. I made use of the list of abbreviations which is appended at the end of each scanned volume<sup>24</sup>. I collected a total of 180 additional abbreviations from both German and French volumes, and, considering the multilingual factor, decided to use the same list for processing both languages.

Extending the abbreviation list significantly improved sentence boundary recognition. In order to have a vague idea of performance, I manually evaluated the first ten pages of every volume, corresponding to around 500 sentences per language. I discovered only five cases of unnecessary sentence splits (false positives), and three cases of missing sentence splits (false negatives) in German. In the French volumes, there were seven false positives and one false negative. Error analysis showed that the list of abbreviations is almost non-exhaustive, and that the list in the volume appendices only contains the most frequent ones. Also, abbreviations are used extremely variably. For instance, all the following abbreviations stand for *unsere gnädigen Herren* ('our gracious lords'): *uggh.*, *ugghh.*, *ugghhr*, *ugh*, *u. g. hr.*, *ughr.* Moreover, names of persons tend to be abbreviated, such as *Heinr.* (instead of *Heinrich*) or *Russ d. Ä.* (*Russ der Ältere*). Ultimately, the NLTK splitter tended to insert a sentence boundary after digits that are followed by a period, which is problematic for dates (e.g. *5. Januar*) and enumerations.

---

<sup>23</sup>NLTK is a freely available compilation of libraries, tools and corpora for NLP tasks.

Cf. <https://www.nltk.org/>.

<sup>24</sup>For some unknown reason, the volume LU I/2 does not have a list of abbreviations. I therefore copied the abbreviations from another volume of the canton of Lucerne, LU II/3.

### 3.2.5 Tokenization

I used the NLTK for word tokenization, which, according to a rough manual investigation, seemed to perform well. However, the tokenizer did not produce the desired output in the case of square brackets. Square brackets frequently occur inside words, as depicted in figure 8:

Wir Albr[recht] und Leupolt etc. bechennen und t<sup>o</sup>n kunt offenbar mit disem  
5 brief, daz wir unsern getrewn Ulr[ichem] dem Gezzler und Heinr[ichen], sinem

Figure 8: Usage of square brackets in an excerpt from AG II/10. Cf. (Rechtsquellenstiftung des Schweizerischen Juristenverbandes, 2009, 4)

The brackets always contain texts by the editors, who may use them to indicate omissions, additions, unreadable segments, and mistakes in the source text<sup>25</sup>. However, square brackets are treated as single tokens by the NLTK tokenizer, and therefore split from the word they belong to. For instance, the string 'Albr[echt]' in figure 8 above is split into the tokens 'Albr', '[', 'echt', and ']'. I considered three possible solutions to resolve this issue. One option would be the direct modification or "tweaking" of the NLTK tokenizer so that it excludes the brackets as a delimiter. I discarded this option as there seemed to be no straightforward way to achieve this. Another solution is to write rules that join the brackets and words together after tokenization. However, I opted for the third solution, which is to simply remove all square brackets from the text. This decision was predicated on the following reasons: first of all, there were numerous words – often proper names – which contained more than one pair of brackets. Since the subsequent NER task is based on approximate string matching, my claim is that the presence of brackets inside tokens will lower string similarity. This, in turn, may lead to the rejection of strings which are actually matches. Second, while the contents of the brackets are essential for the understanding of the text, the brackets themselves are not. Lastly, removing them is the fastest way to resolve the tokenization problem.

Other symbols leading to incorrect tokenization were low and high double quotation marks, which remained stuck on a word instead of being tokenized. When I replaced them all with the basic quotation mark ", they were correctly tokenized. Ultimately, I replaced a single occurrence of the ampersand & with the word 'und'<sup>26</sup>.

---

<sup>25</sup>As defined by the SSRQ transcription guidelines. Note that in the French volumes, angle brackets are sometimes used instead of square brackets.

<sup>26</sup>The ampersand occurred in the volume VD C/1 inside a German book title. However, it is a symbol with a special function (it declares the beginning of an entity reference) in XML, and therefore needs to be either escaped or replaced.

Once the text had been tokenized, it was possible to determine the number of tokens and types in each volume. The removal of square brackets – as described before – slightly modified the text and lead to a change in tokenization. I therefore calculated the tokens and types for two versions, one in which the brackets are deleted, and one in which they are retained. Table 4 shows the numbers for all volumes.

	<b>Without brackets</b>		<b>With brackets</b>	
	Tokens	Types	Tokens	Types
<b>AG II/10</b>	293'221	35'871	297'322	36'035
<b>BE II/9</b>	411'371	45'915	418'204	45'980
<b>LU I/2</b>	163'450	15'308	167'052	15'331
<b>German total</b>	868'042	97'094	882'578	97'346
<b>FR I/2.6</b>	236'308	21'782	239'767	21'929
<b>VD C/1</b>	407'849	42'405	410'031	42'476
<b>VD C/2</b>	389'736	20'694	390'205	20'703
<b>French total</b>	1'033'893	84'881	1'040'003	85'108
<b>Both total</b>	1'901'935	181'975	1'922'581	182'454

Table 4: Numbers of tokens and types in the selected volumes

Naturally, retaining the square brackets leads to an increase in token and type size. The NLTK tokenizer splits the brackets from the words, creating numerous fragmentary, "nonsensical" tokens. In general, the total number is slightly bigger for French than for German. The whole collection that was ultimately used for corpus creation encompasses 1'901'935 tokens and 181'975 types. The relatively high type-token-ratio (about 10 % per volume)<sup>27</sup> suggests wide lexical diversity, which is characteristic of historical texts with extensive spelling variation.

After tokenization, I assigned an ID for each token and sentence. The IDs always trace back to the current page number, e.g. `<W id='p126-s24-t21'>` refers to page 126, sentence 24, token 21. I also enumerated the pages and paragraphs, and assigned a class attribute to the paragraphs according to the set of classes I defined in section 3.2.1. The ultimate result is a valid XML corpus for each volume, following the example of the *Text+Berg* XML. The beginning of the corpus created for VD C/1 is depicted in figure 9.

<sup>27</sup>For comparison, a modern newspaper corpus described by Perkuhn et al. (2012) consisting of 374'495 tokens has a type-token-ratio of 3.2 %. In contrast to that, VD C/1 (with around 408'000 tokens) has a ratio of 10.4 %, and AG II/10 (293'221 tokens) has a ratio of 12.2 %.

```

<?xml version="1.0" encoding="utf-8"?>
<VOLUME id="VD_C_1">
  <PAGE no="1">
    <P class="title" no="1">
      <S id="p1-s1">
        <W id="p1-s1-t1">1</W>
        <W id="p1-s1-t2">.</W>
        <W id="p1-s1-t3">Réorganisation</W>
        <W id="p1-s1-t4">politique</W>
        <W id="p1-s1-t5">et</W>
        <W id="p1-s1-t6">judiciaire</W>
      </S>
    </P>
    <P class="date" no="2">
      <S id="p1-s2">
        <W id="p1-s2-t1">1536</W>
        <W id="p1-s2-t2">mai</W>
        <W id="p1-s2-t3">13</W>
        <W id="p1-s2-t4">.</W>
      </S>
    </P>
  </PAGE>
</VOLUME>

```

Figure 9: Excerpt from the final corpus in XML format of VD C/1

### 3.2.6 Remaining issues

In section 3.1.2 above, I explained why the conversion from *FrameMaker* files into XHTML was a rather complex task. I also mentioned that for this reason, some converted files displayed a degree of data corruption. For example, in some files, the markup was not valid (e.g. due to missing angle brackets), which had to be corrected manually. The most problematic aspect, however, is that the paragraph distribution in the XHTML files does not always correspond to what can be seen in the printed pages. More precisely, the last paragraph of a printed page was frequently moved to the next page in the XHTML. This happened because the XHTML markup does not allow a page break inside a paragraph; the break is always placed before or after it. So far, I have not found an easily manageable way to resolve this problem. An OCR approach similar to what I used for layout restoration could be an option; however, this requires conversion, OCR-processing and comparison of all volumes, which would be very time-consuming. I therefore decided to leave the paragraph distribution unchanged. The XML corpus arranges paragraphs and their contents in the way they are represented in the XHTML files.

## 4 Name Extraction and Recognition

In this chapter, I explain the methods and techniques applied for extracting as well as generating NEs from the volume indices. I also describe and evaluate the algorithms used for the subsequent detection of NEs in the text.

### 4.1 Indices of place and person names

In chapter 2 I explained that most NER systems make use of external resources such as dictionaries, gazetteers, or other databases containing proper names. This may be especially fruitful for rule-based approaches, in the absence of large manually annotated corpora. This is also the case for the SSRQ volumes which do not have any annotations. However, finding appropriate external dictionaries is not trivial. The names occurring in the SSRQ texts refer to historical persons and places, many of which are very specific to Swiss regions. Names of small valleys, townships, communes or districts of medieval Switzerland are unlikely to appear in a dictionary, even if it contained Early New High German names. I also do not expect that Swiss historical surnames such as *Stannpffer*, *Cauerschwin* and *Geissistein*, or nicknames such as *Clewi* and *Büsingery* can be found in external name lists.

Fortunately, there is another resource that can be employed for the SSRQ texts: as mentioned in section 3.1, every volume contains an index of place and person names (German *Orts- und Personenverzeichnis*, French *Index des noms de personnes et de lieux*). This is highly beneficial to the NER task, as NEs can be extracted and then located according to the page information given in the indices. Furthermore, the indices are specific to the volumes, and therefore only contain relevant names.

The idea of utilizing secondary sources for historical NLP was already advocated in similar projects. (Linde and Mitmann, 2013, 235), who used authoritative glossaries to pre-annotate Old German reference corpora, claim:

Creating a linguistically annotated corpus of texts in a historical language is a task that seems to imply a huge amount of manual annotation

work. [...] Thankfully, modern technical facilities allow us to digitize the secondary sources needed, and to automate both the gathering of information and its assembly into a reasonably searchable data structure.

In the case of the SSRQ volumes, the indices are already available in digitized form. They were converted into XHTML files in the same manner as the the volume bodies, as described by Piotrowski (2010). Based upon the visual markup of the *FrameMaker* files, the XHTML tags made the logical structure of the entries explicit. Figure 10 depicts six entries in the index belonging to volume LU I/2, together with the corresponding XHTML code.

Andres Hensli 145<sup>34</sup>, *als Schweinemetzger*  
245<sup>9</sup>  
Ansorg *s.: Onsorg*  
Antonius, Anthonius, confessor (*der Eremit*)  
– Anthonien hus (*Haus der Antoniter = des Antoniusordens*) 284<sup>17f.</sup>  
– Antoniuskapelle, *-bruderschaft in der Franziskanerkirche* 284<sup>20ff.</sup>

(a) Excerpt from a scanned index

```
<p xml:id="LU_I_2_IDX0016" class="person">
<dfn class="hist">Andres</dfn> Hensli {145:34}, <i>als
Schweinemetzger</i> {245:9}</p>
<p xml:id="LU_I_2_IDX0017" class="person">
<dfn class="hist">Ansorg</dfn>
<i>s.: Onsorg</i></p>
<p xml:id="LU_I_2_IDX0018" class="saint">
<dfn class="hist">Antonius</dfn>, Anthonius, confessor <i>
(der Eremit)</i></p>
<p xml:id="LU_I_2_IDX0019" class="place" title="Antonius">
– Anthonien hus <i>(Haus der Antoniter = des
Antoniusordens)</i> 284<sup>17f.</sup></p>
<p xml:id="LU_I_2_IDX0020" class="place" title="Antonius">
– <i>Antoniuskapelle, -bruderschaft in der
Franziskanerkirche</i> 284<sup>20ff.</sup></p>
```

(b) Corresponding XHTML excerpt

Figure 10: Example entries in the index of place and person names of LU I/2, represented in printed (a) and XHTML (b) format.

The entries in an index appear in alphabetical order and consist of main and subordinate entries. For instance, *Antonius* in figure 10 contains three subordinate entries, indicated by a dash. In the XHTML file, subordinate entries denote the corresponding main entry via the `title` attribute. The first name in a main entry, enclosed in `<dfn>`-tags in the XHTML, is usually a modernized spelling variant. The names following it are the historical variants occurring in the text (e.g. *Anthonius*,

*confessor*), separated by commas. Surnames are always listed before first names. Moreover, there may be an editorial description or comment which is written in italics in the printed version, and embedded in `<i>`-tags. Superscript numbers are used for the relevant lines in the text, e.g. '145<sup>34</sup>' means that the preceding names can be found on page 145, line 34. The XHTML files have different ways of representing page and line numbers: they may be given with or without curly brackets, and frequently appear together with the tag `<sup>`. Lastly, an entry may contain a cross-reference to another entry, as can be seen in the case of *Ansorg*. Each entry in the XHTML has a unique identifier (indicated by the `id` attribute) and is assigned to one of the following four class attributes: `person`, `place`, `saint`, and `term`<sup>1</sup>.

### 4.1.1 Pre-processing

In order to locate and annotate the NEs in the texts, they first need to be extracted from the indices and stored in an expedient format. Thus, my goal was to create an easily accessible database containing the entries, together with all the corresponding information given in the indices. I first used the *ElementTree/lxml* module (cf. section 3.2.1) to parse the index files. As was the case with the volume bodies, a number of structural problems and inconsistencies in the XHTML needed to be resolved as well. For instance, there were some cases of broken markup (missing brackets, redundant tags, etc.), fragmented strings (e.g. names and page numbers disrupted by whitespaces), and literal errors in the classes (e.g. 'pplace' instead of `place`). Also, there were omitted characters (commas, dashes) which are visible in the printed version, but dropped away during the conversion from *FrameMaker*. Since the number of structural errors was small, I decided to correct them manually.

Another problem was that in the French index files, the XHTML identifiers of some entries occurred multiple times: whenever an entry contained a cross-reference, it adopted the ID of the referenced entry. However, I preferred to have a unique identifier for each entry. This was not only expected by the *ElementTree/lxml* parser, but was needed for the later distinction of entries in the database. What is more, the duplication also lead to gaps in the numerical naming of IDs, which made a redistribution of the IDs by hand too elaborate. I therefore implemented a small enumeration algorithm in **Python** that automatically and sequentially assigned unique IDs to every entry.

---

<sup>1</sup>The class `term` is used for entities such as documents, treaties, organizations, conferences, measuring units, campaigns or battles.

Lastly, I decided to remove all Roman numerals from the entries, since they are not relevant for the NER task<sup>2</sup>. Also, as they only consist of letters, they may be mistaken for names in the NE extraction process. I achieved the removal by using a function specifically designed for recognizing Roman numerals.

## 4.2 NE database compilation

Basically, there are two requirements for the name extraction task: exactness and completeness. On the one hand, the goal is to have an assembly which ideally contains all names and their spelling variants listed in the indices. On the other hand, the extraction should be as precise as possible, excluding any strings or elements that are not names. Given the clearly arranged structure of the XHTML files, this seems to be a quite straightforward task. However, this is far from true. The ways in which the spelling variants may appear in an index entry are extremely manifold and often inconsistent. On top of that, except for the modernized variant enclosed in the <dfn>-tags, the spelling variants are not indicated by specific markup. Once other elements such as editorial descriptions and page information are extracted, there is still a lot of "noise" left that should not be treated as a NE. A set of specific rules, algorithms and regular expressions were necessary in order to extract (and reconstruct) the names, as described in the following sections.

### 4.2.1 Extracting and generating name variants

In the indices, the editors made extensive use of **parentheses** to indicate spelling variation. Parentheses do not only contain whole words as alternatives (e.g. 'Johann (Hans)'; 'uf (uff) dem Berg'), but also alternative sub-strings (e.g. 'Uberschlachten (-slachtten)'). They may also contain parts of a name that are not essential (e.g. 'Barill(i)et'; 'Längenberg(wald)'; '(nostre) ville de Berne'). In some cases, parentheses are used before or after a name for reasons of clarity or distinction (e.g. '(Varniman) Heini'. I manually investigated the indices in the volumes, categorized all strings containing parentheses, and, based upon these strings, tried to craft a number of rules that are able to produce the respective spelling variants. I further conducted a small evaluation regarding the usefulness of the production rules. Due

---

<sup>2</sup>Roman numerals in the index refer to pages appearing before the source text (i.e. in the foreword and lead texts) at the beginning of a volume. However, the XHTML files of the main body do not contain these initial pages.

to the fact that parentheses occur much more frequently in the German indices<sup>3</sup>, this evaluation was only based on the results for German. Table 5 exemplifies four usage categories of parentheses with input strings, as well as the resulting output strings. The two rows below each category describe the method that I used in order to produce the given output. The numbers in the third column indicate the percentage of correctly produced variant lists per category. I developed the rules based on observations and generalized assumptions about the usage of the parentheses. For instance, if they appeared at the string beginning, I assumed that they contain surnames (which may or may not occur with the first name). Furthermore, the *Jaro-Winkler* metric (explained in section 2.1.2.2) was applied in order to determine whether the parentheses contain a spelling variation or an optional element. The LCS technique (starting from the back of the string) fit the particular purpose of creating variants with alternative endings.

Given input strings	Generated output strings	Correct (%)
<i>(Felwer) Hans</i> <i>(Stukki) Berchtold</i>	<i>Hans, Felwer Hans</i> <i>Berchtold, Stukki Berchtold</i>	89.3
<b>Assumption:</b> String in parentheses is a surname, followed by first name. <b>Solution:</b> Create two variants: 1) first name only, 2) surname and first name.		
<i>Philippus (Filippus) et Jacobus</i> <i>im (jmm) Hasli</i>	<i>Philippus et Jacobus, Filippus et Jacobus</i> <i>im Hasli, jmm Hasli</i>	79.4
<b>Assumption:</b> If token in parentheses is similar to preceding token, it is an alternative spelling. <b>Solution:</b> Check if Jaro-Winkler similarity of the two tokens is >0.65. Create two variants.		
<i>den Wegus (us) gan</i>	<i>den Wegus gan, den Wegus us gan</i>	100
<b>Assumption:</b> If token in parentheses is not similar to the preceding token, it is optional. <b>Solution:</b> Create variants: 1) including optional element, 2) without optional element.		
<i>Purificatio (-cio)</i>	<i>Purificatio, Purificacio</i>	44.4
<b>Assumption:</b> String in parentheses is an alternative suffix or syllable. <b>Solution:</b> Use LCS to find the point for replacing and inserting suffix. Create two variants.		

Table 5: Examples of strings with parentheses and generated variants

The percentages may be slightly misleading, as they are dependent on the number of cases in which the rules actually came into play. For example, the third rule (which treated the parentheses' contents as optional) performed best, but it generated variants only five times. In contrast to that, the second rule generated output in 34 cases. As mentioned before, the production rules are generalizing, and therefore do not always lead to the desired results. While the assumption concerning

<sup>3</sup>I counted 90 different strings with parentheses in the German indices, and 15 different strings in the French ones.

surnames and first names was mostly true, the rule did not take into account that parentheses may contain several tokens, as in the string '(Christen, Kristen) Jenny'. Furthermore, the *Jaro-Winkler* measure failed to capture dissimilar strings that are actually meant to be spelling variants. For instance, the string 'hofgericht (unseres hofs gericht) am St.' should produce the variants 'hofgericht am St.' and 'unseres hofs gericht am St.'. Also, in the ideal case, the rule should recognize that 'Hensli' is an alternative spelling of 'Johannes'. On the other hand, the rule for parentheses containing a dash was too specific. It did not perform well as it neglected the correct treatment of strings such as 'Ärnizhalde (Erniz-)' or 'Hitzkirch(-Richensee)'. My conclusion is that further specifications of the rules (including the improvement of the similarity and LCS-approach) would lead to improved results.

Another step in the NE extraction process was the reconstruction of **abbreviated or shortened strings**. Given the space-saving aspect, the use of abbreviations instead of the full form (e.g. 'Lu.' for *Lucerne*) is very frequent in the indices. The same applies for dashes which are used as place holders, as in 'Winterhalden, -halten' or 'Biem-, Wiembach'. This is problematic, because the written out form is needed to correctly locate the NE in the corpus. Hence, I decided to produce an additional, written out variant of every string containing an abbreviation or shortened form. I achieved this by making use of the entry name, i.e. the (modernized) name given in the <dfn>-tags. For reasons of simplicity, I defined abbreviations as strings consisting of up to three letters, followed by a period. Whenever these letters corresponded to the beginning of the entry name, I extended the abbreviation accordingly<sup>4</sup>. In order to extend the strings with dashes, I extracted the letter occurring just before or after the dash. If the letter could be found in the entry name, I made a split at this position, and inserted the substring there<sup>5</sup>. Of course, this assumption is relatively naïve, and may lead to wrong reconstructions. I therefore included a number of additional rules so that special cases such as 'Mauritius, -cyus' or 'Bartlo-, Partholomeus' were extended in the right way. This crude way of splitting and inserting worked surprisingly well: out of 40 shortened strings with a dash, 32 (i.e. 80 %) were correctly reconstructed. In five cases, there was no correct split possible by only checking a single letter (e.g. 'Schiessi-, Schiessenbrunnen'). Four splits would have worked if phonetic similarity had been considered (as in 'Lutiß-, Lutishofen', 'Kalch-, Kalkbrunnen', or 'Ow-, Augraben'). Table 6 shows six examples of strings containing abbreviations (column 2), together with their entry names (column 1) that were used to produce the written out strings (column 3).

---

<sup>4</sup>In case the entry name consisted of more than two tokens, I compared the letters to any of its tokens, until I found a match.

<sup>5</sup>The search for the letter proceeded in left-to-right fashion. I made sure that the entry name did not get split at its first or last letter.

<b>Entry name (<code>&lt;dfn&gt;</code>)</b>	<b>Abbreviated variant (Input)</b>	<b>Written out variant (Output)</b>
Egloff	E. von Kriens	Egloff von Kriens
Chessel am Graben	metral de Ch. vierteil amm G.	metral de Chessel vierteil amm Graben
Kleindietwil	Clein-	Cleindietwil
Fischmarkt	-mergkt	Fischmergkt
Laurentius	-cyus	Laurencyus

Table 6: Examples of strings in their abbreviated and extended form

As a last step, I defined short rules for the treatment of names with more than one token. This was, for instance, the case with names containing prepositions and articles. In the index entries, these are mostly listed in a dictionary-like fashion, such as 'Hertenstein von', 'Gilgen Zur', 'Alpes, les', or 'Affry, d". Based on a handcrafted list of keywords, I reversed the orders of the tokens. I then made use of the list as well as the entry name in the `<dfn>`-tag, so as to extend incomplete name variants (e.g. 'Pays de' was extended to 'Pays de Vaud').

Once the name extraction and production process had been completed, I stored all spelling variants with the entry name, and assigned them to their unique entry ID given in the index. As mentioned before, each entry has a class attribute, which I used to classify the NEs. Table 7 shows the total number of entries and name variants as well as the NE class distribution for each language. In total, I extracted 9'658 name variants from almost 5'000 index entries. Person names were the most frequent with 2'873 entries, whereas only 73 entries contained saints' names.

	<b>Index entries</b>		<b>Classes</b>			
	<b>IDs</b>	<b>Variants</b>	<b>Person</b>	<b>Place</b>	<b>Saint</b>	<b>Term</b>
<b>German</b>	3'687	6'329	2'271	1'236	70	110
<b>French</b>	1'421	3'329	606	773	3	39
<b>Total</b>	5'108	9'658	2'877	2009	73	149

Table 7: Distribution of names and classes in all volume indices

## 4.2.2 Extraction of other information

In order to complete the database, I needed to consider four more pieces of information per entry: a) page and line indication, b) cross-referencing, c) sub- and super-ordination, and d) editorial notes/descriptions. As mentioned earlier in section 4.1, the page and line numbers may appear in different forms in the index. By a manual inspection I found that the following formatting variations are possible:

- 110<sup>117</sup>
- 247<sup>23, 32</sup>
- 152<sup>33ff.</sup>
- 62<sup>23-31</sup>
- 268<sup>4</sup>–270<sup>25</sup>
- 269<sup>15ss</sup>

My initial assumption was that the line numbers could be very valuable for the NER task, since they help to locate the names at the correct position in the text. This would lead to a precise recognition, as well as to a simplified evaluation. However, I had to discard this idea due to two main reasons: on the one hand, the indicators are not always exact in the sense that they may specify a range of lines or pages. The examples given above – with the exception of the first two – do not specify all the single lines in which a name can be found. For instance, it is not possible to deduce the successive pages or lines from the abbreviations 'ff.' and 'ss.'. On the other hand, the line enumeration in the volume bodies was removed, as explained in section 3.2.3: in the XHTML version of the volume, the line numbers appeared at arbitrary positions in the text, and there was no proper way to find the original line breaks. Thus, I decided to use only the page information, and ignore all line indicators<sup>6</sup>. To this end, I developed a set of regular expressions for the extraction of all page numbers (which assumed various shapes in the XHTML). In the case of page indicators with a range, I automatically added all in-between pages<sup>7</sup>.

The next step was the correct mapping of cross-references. A referenced entry is mostly given in italics, and therefore enclosed within <i>-tags. In a first iteration of the index files, I marked all entries containing a cross-reference. This was possible via rules and regular expressions based on the occurrence of specific strings (such as 'cf.', 's.', 's.a.:', 'vgl.')<sup>8</sup>. In a second iteration, I added all names and variants given

---

<sup>6</sup>I retained the line numbers in the form of page/line tuples, but did not employ them in the NER task. I made sure all page numbers were correctly extracted by a manual evaluation.

<sup>7</sup>For instance, I inserted page 269 between 268 and 270 in the fifth example above.

<sup>8</sup>It was necessary, however, to exclude special cases like 's. auch Sachregister und Glossar' or 'cf. Index des matières', which refer to the glossary and not to another entry.

in the marked entries to their cross-referenced entries. In doing so, I ensured that each entry contained a complete list of name variants.

As explained before, entries may have a list of subordinate entries which are related to it semantically or historically. The distinction between main and subordinate entries was possible by searching for either the initial dash or the `title` tag (cf. figure 10 in section 4.1). Since subordinate entries do not have a `<dfn>`-tag, I simply added it by transforming the `<title>`-tag. I decided to do this because some subordinate entries did not contain any names or variants at all; at times, they only consisted of an editorial note and a page indicator. This means that, ultimately, all subordinate entries contained at least the name given in the main entry. For the sake of completeness, I made the distinction between main and subordinate explicit in the database, but it was irrelevant for the NER task. All entries were stored and looked up in the same manner. Likewise, comments and descriptions in italics were ignored in the recognition task; I only used them for accessing cross-references.

### 4.2.3 Post-processing and output

Before definitely storing all extracted elements into a database, I removed characters and strings that I considered as "noise". Among others, the removal list included square brackets, triple-dots, question and exclamation marks, multiple white spaces, as well as irrelevant punctuation occurring at the beginning or end of a name. Moreover, I removed strings which remained at the end of the extraction process, most of which were fragments of editorial notes such as 'parfois', 'oder', 'Nr.', 'Gde.', 'BE', or 'auch'.

An intermediate format of the database was a nested **Python** dictionary which contained all entries per volume index. For each entry, I stored the following information: definition (i.e. entry name in `<dfn>`), class, name variants, page and line numbers, description, super-ordinate entry, and (optional) cross-references. Additional output formats included a pickled<sup>9</sup> version of the dictionary, a tabulator-separated CSV file, and a plain text file. While the former database format does not deliver insight into its contents (and was therefore merely used for the recognition task), the latter two files simplified manual evaluation and error analysis. In addition, they may be useful for potential future projects involving data visualization and statistics, or even the creation of online IE and QA systems.

---

<sup>9</sup>Pickling converts a **Python** object into a compressed format, so that it can be reconstructed in another **Python** script. Cf. <https://docs.python.org/2/library/pickle.html>.

## 4.2.4 Remaining issues

Overall, the correct and complete extraction of names plus corresponding information was relatively difficult. The way in which the indices are structured differs depending on the language, the volume, and the editors. For some reason, the French indices tend to shuffle subordinate entries together instead of creating separate entries<sup>10</sup>, as can be seen in figure 11 below. This resulted in extremely big database items.

Lausanne, Lausanen, Laussanna, Losanna, Losanne, Losen, Losenn, Loßenn, adj. lausanisch, lausannensis, lausannois 4<sup>47</sup>, 8<sup>20 ss</sup>, 9<sup>4</sup>, 10<sup>23 ss</sup>, 20<sup>37</sup>, 21<sup>1 ss</sup> passim – *l. ville* 6<sup>38</sup>, 9<sup>24</sup>, 13<sup>19</sup>, 20<sup>41</sup>, 22<sup>2, 17</sup>, 23<sup>14 ss</sup> passim, statt, stad L. etc. 60<sup>5</sup>, 121<sup>7</sup>, 298<sup>18</sup>, 659<sup>5 ss</sup>, 660<sup>15</sup> – artisans: horlogers 626<sup>23, 26</sup>; imprimeurs 674<sup>2, 11</sup> – bâtiments: Chapitre (siège du chapitre épiscopal situé à la Cité, à l'époque bernoise siège de la Cour du jadis chapitre) 22<sup>18, 43</sup>, 30<sup>1, 39</sup>; Chasteau de Saint-Maire demeure de l'évêque, dès 1536 du bailli 23<sup>26</sup>, 71<sup>37</sup>, 369<sup>1</sup>; Clergie bâtiment situé à la Cité

Figure 11: Excerpt from a problematic entry in the index of VD C/1

Furthermore, the entries do not only list names, but may contain many other textual elements. For instance, the entry *Lausanne* in figure 11 contains descriptive strings that stem from the editors (e.g. 'Clergie bâtiment situé à la Cité', 'artisans: horlogers'). Without manual investigations, these strings are very hard to automatically distinguish from "real" names. Apart from that, the following issues need to be addressed in future work:

**Distinction of names and phrases:** Entries may contain phrases related to a name (e.g. 'ze Kriens verkunden', 'gen Ure faren'). Although the phrases actually occur in the text, they cannot really be treated as NEs.

**Contents in italics:** The formatting in the indices – as well as the XHTML markup – is not used consistently. This means that in some cases, name variants are enclosed in the <i>-tags, where I could not distinguish them from irrelevant contents (such as descriptions and notes).

<sup>10</sup>For example, the entry *Fribourg* in the index of FR I/2.6 spans three printed book pages, corresponding to a huge single entry in the XHTML.

**Cross-references:** The referencing abbreviations are sometimes used for indicating (vaguely) related entries (e.g. 'Frédéric III, cf. Habsbourg'). In that case, the name variants should not be added to the referenced entry.

**Missing page numbers:** If an entity occurred very frequently in a volume, the editors omitted the page and line indicators (e.g. *Luzern* in LU I/2).

**Common and ambiguous words:** The entries may contain names that are, at the same time, common words such as 'Hof', 'Sohn', or 'ville'. As the line numbers could not be used, this resulted in wrong matches during the NER process.

I believe that most of these problems can only be solved by manual investigations and extensive post-editing. Automated processes, as described in this section, are highly dependent on the consistency of the input, which was unfortunately not given in the SSRQ data<sup>11</sup>. However, a manual review of the resulting database showed that the methods used in the extraction task are promising, leading to correct output in most of the entries. I would also like to argue that the use of approximate string matching techniques, as described in the next section, was able to compensate for many of the shortcomings of the extraction process.

### 4.3 Recognition and annotation of entities

Once the database compilation had been completed, the names and page information could be used to detect the NEs in the volume bodies. For this end, I implemented a detection procedure in **Python** which takes as input the created database and corpus files (XML files, as described in chapter 3). Note that, in fact, the task of the NER procedure as such is NE detection only, as it locates relevant strings in the text. The NE classification was carried out based on the classes given in the database entries. I first reordered and sorted the **Python** dictionary according to page numbers, and then searched through the complete XML corpus of every volume. Thus, the search was not limited to the source text, but included all paragraphs – i.e. including titles and all contents composed by the editors. The output of the algorithm was, again, based on the model of the *Text+Berg digital* project, in which annotations were made in a stand-off XML file. The *Text+Berg* annotation file contains an entry for every NE occurring in the corpus, and gives information about its position and class embedded in XML element attributes.

---

<sup>11</sup>Generally, the extraction performed better in the German indices, as they tended to be less variably structured.

### 4.3.1 Procedure

The recognition process was divided into two components: a **direct** search and an **approximate** search component. Approximate string matching was essential in this task for three reasons. First of all, as mentioned before, historical texts do not have a standardized orthography, which means that names appear in numerous different spelling variants. Although the indices normally list the most frequent variants, the list is never complete. Second, it should be taken into account that the indices hardly ever mention inflected forms, such as genitive, plural, or adjectival forms (e.g. 'Verenen', 'Laurentii', 'Hofmanns', 'Lausannensis', 'Zürcher', 'Capelgassen'). Third, both the tokens in the corpus and in the database may be erroneous to some extent. In section 3.2.3 I explained that footnote reference letters were not removed from the text, and that within `<table>` elements, these letters were directly attached to the tokens. This means that some tokens had an additional last letter which is not part of the word. Moreover, the complete removal of unwanted strings in the index entries was not always possible<sup>12</sup>.

Basically, the procedure operated in two nested iterations per volume. The first iteration parsed<sup>13</sup> the whole XML corpus on page, paragraph, sentence and token level, looked up entries and all name variants relevant for the current page (as specified in the **Python** dictionary), and stored all name variants that could be matched. The remaining entries, i.e. the ones that could not be found anywhere (neither by direct nor approximate matching) in the corpus after the first iteration, were stored separately. This process was repeated in the second iteration, with the important difference that it only searched the **first** paragraph of each page. I decided to do this because, as explained in section 3.2.6, numerous paragraphs were accidentally moved down to the following page in the XHTML. In other words, if a NE was not detectable on the relevant page in the first corpus iteration, it could mean that it occurred in a moved paragraph on the next page<sup>14</sup>. This method was very effective as it was able to find a significant portion of missing NEs. For example, in the first corpus iteration of volume FR I/2\_6, I could not find any matches for 86 different database entries. After the second iteration, only five entries remained, which means that more than 90 % of all entries contained NEs occurring in displaced paragraphs.

---

<sup>12</sup>This sometimes resulted in the comparison between corpus tokens and "noisy" strings which were actually not NEs.

<sup>13</sup>Using *ElementTree/ lxml*.

<sup>14</sup>To make sure that the correct pages were searched, I incremented the page numbers of the missing entries in the temporary database by 1.

### 4.3.2 Direct and approximate search

In the direct search component, I tried to find strings that are one-to-one matches. For every page in the corpus, I extracted only the relevant entries from the (re-ordered) database. All name variants given in these entries were then searched for in the current page. This search was always based on the number of tokens in the variant: for instance, if a variant consisted of three tokens, I progressively dissected the text into strings of three tokens<sup>15</sup>. This made sure that only strings of the same token length were compared to each other<sup>16</sup>. If there was a one-to-one correspondence between all characters in both strings, I stored them as direct matches.

I conducted several experiments with different similarity measures and thresholds in order to find the best technique for the approximate search component. I also estimated the usefulness of all measures that were mentioned in section 2.1.2.2 for this specific detection task. I discarded the idea of using phonetic encoding techniques, since almost all of them are designed to process English input, and none of them are adapted to historical texts<sup>17</sup>. Moreover, a phonetic code alone does not quantify similarity; further measures are required to define which codes should be considered similar. Testing the *Levenshtein* distance demonstrated that, despite its advantages, the measure did not put enough weight on the beginning of the strings. Ultimately, given the fact that the *Jaro-Winkler* metric has been proven to work well with proper names, I decided to base the approximate search algorithm predominantly upon this measure. The **Python** module I used for approximate string matching was *Jellyfish*<sup>18</sup>.

The most difficult aspect of engineering the algorithms was finding an optimal threshold for the *Jaro-Winkler* metric, while bearing the trade-off between precision and recall in mind. By using a high threshold of at least 0.9, several matches were not captured, i.e. lead to False Negatives (e.g. *Agatha* – ‘Agathe’, or *Barnabas* – ‘Barnabe’). On the other hand, lowering it to 0.8 resulted in False Positives (e.g. *Peter* – ‘empter’, *Hilarius* – ‘hinaus’). After numerous repeated test runs with different thresholds, I found that 0.825 was a very balanced ratio. Unfortunately,

---

<sup>15</sup>I used the NLTK tokenizer to tokenize both the variant strings and the sentences.

<sup>16</sup>This token-based approach was also easy to perform because the corpus had already been tokenized. Reassembling the tokens to a string and then performing a full-text search would have been more time-consuming, especially since **Python**’s regular expressions treats diacritical symbols as a word boundary.

<sup>17</sup>I did not make any experiments with the *Meyer-Wilde* algorithm for German, since the replacement rules seemed to be very simplistic (cf. (Kempken, 2005, 11)). However, it would be interesting to test its performance with Early New High German input in future work.

<sup>18</sup>*Jellyfish* contains ten different algorithms for doing approximate and phonetic string matching. Cf. <https://pypi.python.org/pypi/jellyfish>.

the encoding framework of *Jellyfish*'s calculation function tended to split single diacritical symbols into two separate symbols, e.g. the symbol ó was split into the letter o and the diacritical mark '. Assuming this might lead to wrong comparisons, I implemented a normalization rule that removed all diacritical symbols from the strings. Note that this rule only applied if the *Jaro-Winkler* ratio of the strings was below the threshold; if the threshold could be reached by removing the diacritics, the strings were considered as matches.

Furthermore, very short strings were extremely problematic, as it was easier for them to pass the threshold. The person name *Hag*, for instances, was always falsely matched with the string 'Tag'. Therefore, I defined further restrictions for strings shorter than four characters: the *Jaro-Winkler* similarity needed to be higher than 0.84, and the *Hamming* distance (cf. section 2.1.2.2) lower than 2. As a last step, I added a list of exceptions based on the German and French stop word lists included in *NLTK*. The tokens in these list (mostly function words) were not treated as names and thus ignored in the string matching. The following pseudo code summarizes the steps of the approximate search algorithm:

---

**Algorithm 1** Decide if name variant and current text string are matches

---

**Ensure:** *text\_string* **not in** *stopwords* & *variant* **not in** *stopwords*

```
if length(variant) > 4 then
  if jaro_sim > 0.825 then return variant
  else
    if jaro_sim_normvariant > 0.825 then return variant
    end if
  end if
else
  if jaro_sim > 0.84 & hamming_dist < 2 then return variant
  end if
end if
```

---

It is true that the algorithm could be refined by incorporating more similarity measures or techniques such as n-gram or LCS comparisons. I also conducted some experiments with combining the *Jaro-Winkler* similarity with the *Levenshtein* distance: the results showed that further restricting the search by setting the maximal *Levenshtein* distance to 3 lead to slightly better precision, whereas recall worsened (e.g. the historical holiday name *Symon et Judas* could not be matched to the string 'Simonis et Jude' anymore). In most of the other experiments, I found that the combination of different techniques just complicated and slowed down the matching process, and did not lead to significant improvements.

### 4.3.3 Stand-off annotation

Every NE detected by either the direct or the approximate search process was stored in a stand-off XML file, following the example of the *Text+Berg digital* XML. This file was created once for every volume and contained the following elements: match ID, NE class, position in the text (given in the form of a token span<sup>19</sup>), the ID of the NE in the database, and a confidence value. The confidence value is simply the (rounded up) *Jaro-Winkler* distance; in case of a direct match, the value is 1. The match IDs were assigned continuously, i.e. depending on the order of occurrence. Figure 12 shows an excerpt from the stand-off file created for AG II/10.

```
<ne id="327" class="person" span="p5-s5-t57" idx="AG_2_10_IDX0331" confidence="1" />
<ne id="328" class="person" span="p5-s5-t57" idx="AG_2_10_IDX1096" confidence="0.933" />
<ne id="329" class="person" span="p5-s5-t58" idx="AG_2_10_IDX0876" confidence="1" />
<ne id="330" class="person" span="p5-s6-t11, p5-s6-t12" idx="AG_2_10_IDX0771" confidence="0.846" />
<ne id="331" class="person" span="p5-s6-t18" idx="AG_2_10_IDX0787" confidence="0.867" />
<ne id="332" class="person" span="p5-s8-t6" idx="AG_2_10_IDX0804" confidence="1" />
<ne id="333" class="person" span="p5-s8-t15" idx="AG_2_10_IDX0787" confidence="0.867" />
<ne id="334" class="person" span="p5-s8-t19" idx="AG_2_10_IDX0787" confidence="0.963" />
```

Figure 12: Example of a stand-off NE annotation file in XML format

Note that I generated a new `<ne>` entry for every name variant that was found by the two search algorithms. Thus, overlaps are possible, since each variant string could match several times within the same text string. For example, token 57 in the excerpt above was matched two times, once in the direct search, and once in the approximate search. Of course, one might argue that this generates a certain kind of redundancy. However, my motivation for creating the stand-off file in this way was completeness. Every file stores all possible matches, thus enabling a filtering according to confidence value.

The main advantage of a stand-off annotation file is that it can be inspected separately from the corpus, and thus easily extended or used for further applications. For instance, the token position given in the `<span>`-tag may be employed to later visually highlight the NEs in the running text of the volumes. Apart from the stand-off annotation file, I additionally created a verticalized version of the corpus with inline annotations in the form of (B)IO-tags. This version contained plain text (i.e. was free of markup) and was later used for Machine Learning experiments, as described in section 5.

<sup>19</sup>For example, `span='p1-s20-t43, p1-s20-t44'` means that the NE has 2 tokens, beginning at token 43 and ending at token 44 in sentence 20.

## 4.3.4 Evaluation

### 4.3.4.1 Precision

As a by-product of the stand-off annotation, I created a subset file for every volume which stored only matches with a confidence value below 1. This allowed me to carry out an evaluation of the results generated by the approximate search component, and to see how well the matching algorithm – as outlined in the previous section – performed. I investigated these evaluation files of two randomly selected volumes (one per language), and manually checked whether the matched strings are actually corresponding<sup>20</sup>.

At an early stage of the evaluation, it became evident that a large amount of False Positives (especially in the French volume, VD C/2) came about due to only one single character mismatch. Since the algorithm compared strings of the same token number, this resulted in matches such as *La Mauguettaz* - 'Mauguettaz ,' or *Kappel brugg* - ') Kappelbrugg'. I therefore decided to evaluate in two steps: first with a low error tolerance, in which these mismatches were not counted as errors, and after with a higher error tolerance, where I counted them as errors. The results of the evaluation are displayed in table 8 below.

	Predicted	Tolerance	TP	FP	Precision (%)
LU I/2	5223	low	4792	431	91.76
		high	4843	380	92.73
VD C/2	1597	low	1408	189	88.17
		high	1528	69	95.68

Table 8: Evaluation of the approximate search algorithm for two volumes

Overall, the precision was reasonably high in both volumes (92.73 % for German and 95.68 % for French, allowing +1 token mismatches). It should be kept in mind, however, that the French index was smaller: the approximate search in volume LU 1/2 generated a total of 5'223 matches, whereas it found 1'597 matches in VD C/2. My claim is that precision is not mainly dependent on the language, but on the quality of the indices. Generally, the French name variants were much longer (the index of VD C/2 contained multi-token variants, e.g. 'villes et communaultez du Pays de Vaud').

Moreover, the token-based string comparison seemed to be quite problematic. For instance, the variant *stattschriber* was matched to the string 'schriber', but

<sup>20</sup>Workload of approx. 5 hours.

not to 'statt schriber'. A full-text search or regular expressions may solve this problem. I also believe that removing punctuation after tokenization would lead to less mismatches. Nevertheless, considering the fact that the approximate search was mainly built upon the *Jaro-Winkler* metric, the results were quite promising. Table 9 lists a number of good and bad examples of matches with the corresponding (rounded up) *Jaro-Winkler* similarity.

	String from index	String in volume text	Jaro
	<b>LU I/2</b>		
<b>Correct matches</b>	<i>Ysengass</i>	Eisengasse	0.86
	<i>Symon et Judas</i>	Symonis et Jude	0.89
	<i>Leodegarius</i>	Leodegarstag	0.86
<b>Wrong matches</b>	<i>Peterman</i>	iederman	0.83
	<i>der Wechsler</i>	den wechsell	0.86
	<i>Münchs wib</i>	Münch nit	0.84
	<b>VD C/2</b>		
<b>Correct matches</b>	<i>Saint-Jean-</i>	Saint-Jehan-Baptiste	0.86
	<i>Saint-Barthélémi</i>	Saint-Bartholomier	0.87
	<i>Lausanne</i>	lausanneoise	0.84
<b>Wrong matches</b>	<i>Suisse</i>	puisse	0.89
	<i>ville de Nyon</i>	ville de Moudon	0.85
	<i>ville</i>	fille	0.87

Table 9: Examples of correct and wrong matches, taken from two volumes

The advantage of the approximate search algorithm was that it was able to capture variational morphology, as can be seen with *Lausanne* - 'lausanneoise'. In cases like 'Eisengasse' and 'Bartholomier', the algorithm recognized the strings as spelling variants. Furthermore, it was sometimes able to compensate for gaps in the indices: for example, the fragment 'Saint-Jean-' was correctly matched with 'Saint-Jehan-Baptiste', and the compound 'Leodegarstag' – which was not listed in the index entry – was correctly matched with the name *Leodegarius*. In other cases, the strings were too similar to be distinguished (e.g. 'wib' and 'nit', 'Nyon' and 'Moudon'). Potential solutions include putting even more weight on the beginning of strings, and extending the list of stop words.

In this evaluation I focused on the string matching ability of the approximate search. Given the absence of the line numbers, I was not able to deal with grounding of NEs. Thus, the results in table 8 did not take NE ambiguity into account. For instance, there were multiple entries in the LU I/2 index with the entry name *Peter*, all referring to different individuals<sup>21</sup>. In the evaluation, I checked whether the person name *Peter* is a spelling variant of, say, *Petter*, as defined by the index; but I did not verify if the ID that matched the name was actually the "right" person. The search operated on whole (single) pages, without taking line indicators into account. If it was possible to reconstruct line numbers, one could restrict the scope of the search, thus limiting the possibility of wrongly associated IDs. For the remaining ambiguity, i.e. occurrence of multiple individuals on the same line, the idea of "one sense per discourse" might be interesting<sup>22</sup>.

#### 4.3.4.2 Recall

Unfortunately, since line numbers could not be taken into account, calculating the recall turned out to be a very laborious task. The elicitation of the exact number of False Negatives (i.e. name variants that could not be matched) requires the manual look-up of all variants in the indices and then counting them in the volume bodies, based on the line numbers indicated. After, the occurrences need to be compared to the output of the algorithm. As this would go beyond the scope of my project, I decided to perform the evaluation on the level of index entries instead of the level of name variants.

The first part of the evaluation focused on the number of queries. I defined a query to be the looking up of a single entry on a given page. For example, according to the index database of volume AG II/10, there were two different entries (not name variants!) to be found on page 6: 'IDX0142' and 'IDX0300'. This corresponds to two queries.

For every volume, I counted the following: 1) total number of queries to be made, 2) number of successful queries, and 3) number of failed queries. A successful query means that an entry had at least one name variant that could be matched on the relevant page. The counting of 1), total number of queries to be made, was done automatically via the reordered NE database. This generated a **Python** dictionary which could be used for manual evaluation, as exemplified in the following snippet:

---

<sup>21</sup>Note that, however, the editors often conflated several different individuals under the same entry. For correct NE grounding, these need to be assigned to separate IDs first.

<sup>22</sup>Cf. Gale et al. (1992).

```

FR-I-2_queries = {
'Page 1': [IDX0238, IDX0226, IDX0143], # 3 queries
'Page 2': [IDX0238, IDX0298, IDX0226, IDX0459], # 4 queries
'Page 3': [IDX0143], # 1 query
'Page 4': [IDX0226, IDX0459, IDX0143], # 3 queries
...}

```

I achieved the automated counting of 2) and 3) by modifying the search components so that they kept track of the matches. In the second step of the evaluation, I developed a function which automatically counts the number of different entries (i.e. IDs in the database), for which the algorithms did not find any matches at all. The results of the two evaluation steps are given in table 10 below.

	Number of queries			Number of entries		
	Total	Successful	Failed	Total	Matched	Unmatched
<b>AG II/10</b>	2'649	2'528 (95.4 %)	121	1'119	1'045 (93.4 %)	74
<b>BE II/9</b>	2'686	2'566 (95.5 %)	120	1'615	1'528 (94.6 %)	87
<b>LU I/2</b>	2'042	1'955 (95.7 %)	87	953	918 (96.3 %)	35
<b>FR I/2.6</b>	977	972 (99.5 %)	5	471	467 (99.2 %)	4
<b>VD C/1</b>	1'993	1'813 (90.7 %)	180	619	542 (87.6 %)	77
<b>VD C/2</b>	666	630 (94.6 %)	36	331	304 (91.8 %)	27

Table 10: Evaluation of recall on entry level in all volumes

The table lists the total number of queries in each volume, as well as the number of successful and unsuccessful queries. In AG II/10, for instance, the algorithm looked up entries 2'649 times, and in 2'528 cases this returned matches. In 121 queries, the algorithm could not find any match. The number of entries refers to the relevant IDs in the database. For AG II/10, the database contained 1'119 different entries, 1'045 of which could be matched at least once anywhere in the corpus. 74 entries could not be matched at all, neither by the direct nor by the approximate search, and even when considering all spelling variants.

Except for VD C/1, the search algorithms reached a query and entry recall of over 90 % in all volumes. They performed best in volume FR I/2.6, and worst in VD C/1. It is not exactly clear why in FR I/2 the search achieved outstanding results; it can be assumed that, in turn, the precision was slightly worse than for the other volumes. My explanation for the relatively low results in VD C/1 is the complexity of the volume index, which made it difficult to properly extract spelling variants.

As already mentioned, the manual evaluation of False Negatives is very time-consuming. In order to conduct an error analysis, I therefore decided to randomly select 105 cases of unsuccessful queries per language<sup>23</sup>, and determine the reason why the query failed. I looked the entries up in the (printed version of) the volume indices, investigated the relevant pages, and distributed the error sources into five categories. The results of this evaluation are given in figure 13.

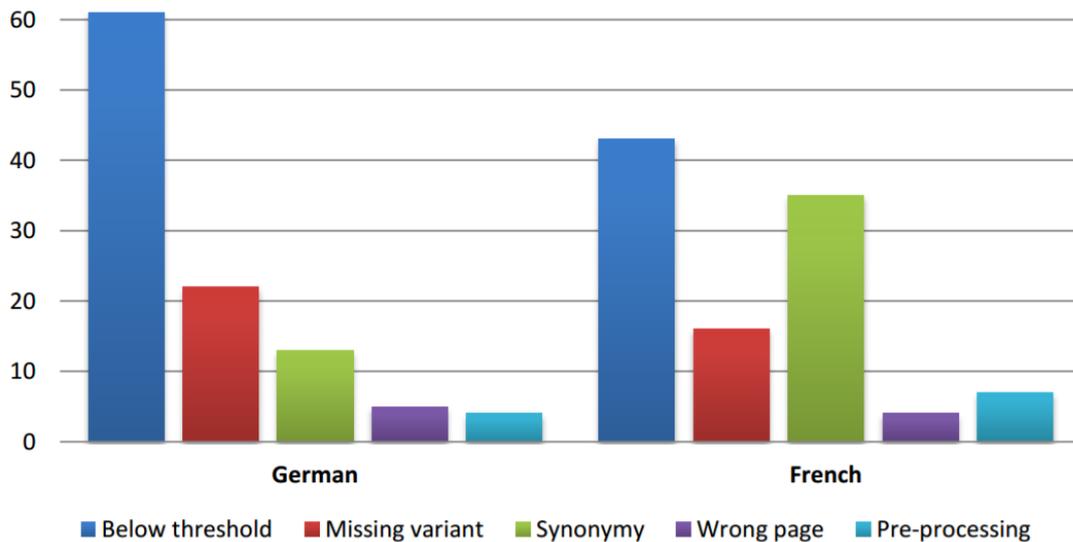


Figure 13: Sources of error for False Negatives

By investigating the queries I found that the following issues lead to False Negatives, which will need to be addressed in the future:

**Below threshold:** A significant number of unsuccessful queries were due to the *Jaro-Winkler* measure specified in the algorithm. For instance, the name *Conversio Sancti Pauli* could not be matched to the string 'conversionem Pauli'. Frequently, the threshold was missed by only 0.03 (e.g. when trying to match the saint's name *Gallus* with its inflected form 'Galli')<sup>24</sup>. These cases could have been matched by lowering the threshold (which would, however, lead to new False Positives). It could be beneficial to set the thresholds for individual tokens of a string, instead of defining one for the whole string.

**Missing variant:** Some entries contained the name variant embedded within `<i>`-tags. As explained before, the search algorithm ignored strings in these tags, as they were regarded as editorial comments.

<sup>23</sup>35 queries per volume, except for French (where I picked 5 of FR I/2.6 and 65 of VD C/1); evaluation workload of approx. 6 hours.

<sup>24</sup>The similarity measure is case-sensitive. In very short strings, the differences in lower- and upper-casing easily lead to a rapid drop in the ratio.

**Synonymy:** This was a problem especially in the French volumes. Page/line indicators in the indices often pointed to strings which were not listed in the entry, but denoted the same entity. For example, one entry contained the name *Frédéric empereur*, but in the given pages this person was referred to as 'prince'. Another example was the name *Schindler* which always appeared in the text as 'der Vogt'. Naturally, the volume editors constructed the index entries in such a way that they are understandable and deducible by human readers. Automated processes, however, would need more refined rules and sophisticated algorithms<sup>25</sup>.

**Wrong page:** In fact, the page indicators given by the editors were not always correct. Sometimes, they miscounted by one page, so that the NE could not be located at the given position.

**Pre-processing:** These queries would have been successful if the name variant extraction and production task had performed better (this includes dealing with parentheses in names such as *Niederwil(-Tägerig)* or expanding the abbreviation 'St.' to 'Sankt'.).

As stated before, this evaluation gives an insight into the recall on entry level (not on level of spelling variants). However, taking the results of the approximate search into account, it allows for a rough estimation of the overall accuracy<sup>26</sup>. Of course, the (rare) case of wrong one-to-one matches need to be taken into account as well. In summary, the NE detection process was focused more on precision than on recall. The error analysis in this section underlined the important role consistent input plays in automated processes, and that careful pre-processing is essential in order to prevent subsequent faults. In the future, more experiments will be carried out so as to find the ideal similarity measures, thresholds and rules for the approximate search algorithm.

---

<sup>25</sup>E.g. co-reference resolution, cf. Krug et al. (2015), Ebling et al. (2011).

<sup>26</sup>E.g. by calculating the average number of variants per entry, and the average number of occurrences of a variant per page.

## 5 Machine Learning Approach

This chapter addresses two experiments that were conducted using systems based on CRF models. The first experiment deals with tagging the pre-processed SSRQ data with *Stanford's* off-the-shelf NER tool, while the second is training an own CRF classifier by combining the tags generated by *Stanford NER* with the ones produced by the rule-based algorithms.

### 5.1 Applying modern language NER systems

After having completed the work steps delineated in the previous two chapters, the result is a cleaned XML corpus with externally annotated NEs. Since the annotations were made automatically by handcrafted rules and have not been post-edited so far, the corpus could be referred to as being "pre-annotated". As noted earlier, the algorithm applied in the NE detection process was centered more on correctness (precision) than on completeness (recall), which left certain NEs unrecognized. It can also be assumed that a volume index, being a manually produced resource, contains a certain number of gaps. For instance, I have explained that frequent occurrences of a NE such as *Luzern* were not always indicated by page/line numbers. I also expect that volume editors may – either accidentally or deliberately – omit the mentions of names and references in the index entries. On account of this, it would be desirable to find a way of detecting remaining NEs, and thus compensate for the aforementioned shortcomings. One way of doing so is the use of another NER tool originally designed for modern texts. In fact, "when working with a historical variant of an extant language, a frequent approach is to start off with an existing tagger for the modern variant" (Piotrowski, 2012, 91). In this context, I was especially interested in the question whether an off-the-shelf modern language NER system would find names and spelling variants which were not mentioned in the indices, or which could not be recognized by the algorithm.

### 5.1.1 *Stanford NER*

There is a rich array of open-source NE recognizers for modern languages, such as *GATE/ANNIE* (Cunningham et al., 2002), NLTK's NE classifier (Bird et al., 2010, 281ff.), the *Stanford NER* (Finkel et al., 2005), or *Cogcomp-NER* (Ratinov and Roth, 2009). I ultimately decided to use *Stanford NER*<sup>1</sup> due to the following reasons: first, the software is well documented and, being implemented in **Java**, available with numerous extensions (including interfaces for **Perl**, **PHP**, **Ruby**, and **Python**). Second, there are several pre-trained models ready for use to tag English, German, Chinese, French and Spanish input. Third, *Stanford NER* is often regarded as the state-of-the-art system because of its robustness across domains, and reportedly good performance: for English, the system reached F-scores ranging from 85.5 % (on the CoNLL-2003 dataset) to 92.3 % (CMU Announcements dataset) (Finkel et al., 2005, 368). The *Stanford NER* software is based on linear chain CRF sequence models using a wide range of features.

#### 5.1.1.1 Classifiers

The two serialized classifiers for tagging **German** input were developed by Faruqui and Padó (2010). Both models were trained on the German training set from the CoNLL-2003 Shared Task<sup>2</sup> using generalization data from two large corpora: the HGC (*Huge German Corpus*, also known as *Stuttgart University Newspaper Corpus*<sup>3</sup>) and *deWac* (*German Web Corpus*<sup>4</sup>). The generalizations were made by forming distributional similarity lexicons from 175 million tokens of each corpus. While the HGC is a collection of news-wire texts, the *deWac* corpus was compiled by scraping off content from the web. It thus contains data from various genres and has been suggested to be used with all documents other than news texts. For NE-tagging the German data, I therefore chose to apply the *deWac* model.

The **French** classifier was trained on a *Wikipedia*-based corpus project described by Nothman et al. (2013). In this so-called WP2 corpus, meta-data and links between encyclopedia articles were exploited for creating NE annotations. Evaluating the automatically labeled corpus against a gold-standard of manually annotated data demonstrated that "transforming *Wikipedia* into training data [...] provides a free

---

<sup>1</sup><https://nlp.stanford.edu/software/CRF-NER.html>

<sup>2</sup>Cf. Tjong Kim Sang and De Meulder (2003) and <https://www.cnts.ua.ac.be/conll2003/ner/>.

<sup>3</sup>Cf. <https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/hgc.en.html>.

<sup>4</sup>Cf. Adam et al. (2010).

and high-yield alternative to the laborious manual annotation required for NER” (Nothman et al., 2008, 131). A part of the project involved manually annotating 4’800 English Wikipedia articles, and then using inter-language links to project the labels onto eight other languages, including French. The resulting French corpus consisted of around 3.5 million tokens, which was subsequently used to train the CRF model.

Both the German *deWac* and the French WP2 classifier produce BIO-tags for the four classes person (PER), location (LOC), organization (ORG), and miscellaneous (MISC). In the *Stanford NER* execution command, different settings can be specified, as well as the models, input file, and output format. There are five choices of different output formats available including XML markup and tab-separated values. Moreover, it is possible to tag input that is already tokenized<sup>5</sup>. In order to use the *Stanford NER* tagger on my corpus, I first extracted the markup from the XML corpus files for all volumes, so as to obtain the volume bodies in tokenized plain text format. Then, I concatenated all three volumes for the two languages, which resulted in two large plain text files (one for German and one for French). Per default, the *Stanford NER* output is produced in the `slashTags` format, in which each token is followed by a slash and the tag. The following excerpt from the German corpus file shows `slashTags`-formatted output produced by the tagger (ellipses inserted by me):

```
Friedensabkommen/O zwischen/O Thierstein/I-LOC und/O Basel/I-LOC  
sowie/O Kiburg/I-LOC und/O Burgdorf/I-LOC 1378/O Dezember/O 29/O [...]  
Und/O stost/O die/O vorgenant/O [...] an/O den/O Toggelbrunnen/I-LOC  
und/O ander/O obgenanten/O von/O Burgdorff/I-PER hoeltzer/O ./O Aber/O  
die/O Winterhalten/O stost/O an/O Niklisbrunnen/I-LOC hinder/O dem/O  
Tütlisperg/I-LOC und/O an/O der/O Cartuser/I-MISC höltzer/O ./O
```

Interestingly, both with the German and French model, the tagger did not generate any B (beginning) tags in any of the corpus files. For reasons of time I had to abstain from conducting research regarding the cause of this lack. However, I assume that since the German and French models were developed for the *Stanford NER* version 1.2 (the newest release is 3.8.0), they use simplified tags only. Having said that, the distinction between an entity (I) and a non-entity (O) was still explicit, which is why I chose to proceed with the IO-tagged output.

---

<sup>5</sup>This can be done by setting the parameter `”tokenizeNLs=true”`. Otherwise, *Stanford NER* applies an own tokenization algorithm to tokenize the input, which differs from the one used by NLTK.

### 5.1.1.2 Building an IO-tagged corpus

In order to compare the *Stanford NER*-tagged corpus to the annotations generated by the hand-crafted algorithms, I modified the output function to produce an additional version of the corpus with IO-tags. Thus, whenever a match could be found, it was not only added to the stand-off annotation file, but also to the plain text corpus (in the form of the corresponding tag). Since the *Stanford NER* models operate with a different set of NE classes, I first needed to transform the classes from the indices. For instance, the models do not have a label for saints' names or terms. I therefore mapped the index classes to the *Stanford NER* labels as follows:

- `person` → I-PER
- `place` → I-LOC
- `term` → I-ORG
- `saint` → I-MISC

While `person` and `place` are used in the same way as I-PER and I-LOC, it could be argued that `term` is not exactly corresponding to the class I-ORG (as `term` also includes treaties and documents). It is true that mapping `saint` to I-MISC is an abstraction, and that it could also be subsumed by the tag I-PER. However, it is a very small class, and I preferred adhering to a fourfold classification.

As explained earlier, the search algorithms could match name variants to the same string multiple times, i.e. in the cases of nested entities or overlaps. Nested entities occurred whenever a longer string was matched which contained a substring that was matched as well. For instance, the string 'Zuo dem Brunnen' was first detected by the one-to-one search. There, it could be directly matched to the identical name variant. After that, the place name *Brunnen* was additionally detected within that string. Overlapping refers to cases where the same string was first detected by the direct search, and later by the approximate search.

However, for the creation of a tagged file in `slashTags` format I decided to resolve nesting and avoid any overlaps. This was partly methodologically motivated, since "shallow" annotations on one level are easier to generate and maintain. It also had a pragmatic motivation, since the output created by *Stanford NER* did not include the B-tag. Moreover, if the goal is to train an own CRF model based on nested annotations, it would be necessary to add two or more columns of labels to the tokens. This "complicates the treatment of the data, because most machine learning classifiers are not designed to attribute more than one class to each instance" (Aguilar et al., 2016, 69).

In order to obtain a shallow annotation, I made sure that longer strings were tagged before short strings, and that the same tokens could not be matched multiple times. Thus, in the example of 'Zuo dem Brunnen', the substring 'Brunnen' could not be matched anymore. The NE tags assigned by the search algorithms were gradually added to the corpus, based on the following hierarchy:

1. Direct search, strings longer than one token
2. Direct search, strings with exactly one token
3. Approximate search, strings longer than one token
4. Approximate search, strings with exactly one token

By using this hierarchy, I made sure that one-to-one correspondences had priority over fuzzy matches, and that strings consisting of more than one token were tagged first. For adding the tags in each language, I created a plain text version of the corpus and stored it in the form of a **Python** dictionary, where the token IDs were used as key and the NE tags as value. Initially, each token ID had the default value O. Every step in the hierarchy above corresponded to one full iteration over the tokens in the dictionary. Once a name variant could be matched to a string, the token values were changed from O to the corresponding class tag. In the next iteration, the same tokens could not be matched again. The result was one tokenized, IO-tagged corpus per language that was directly comparable to the output generated by the *Stanford NER* classifiers.

### 5.1.1.3 Comparison and evaluation

Table 11 below shows the number of tags generated by *Stanford NER* as well as by the rule-based algorithms. *Stanford NER* produced an extremely large amount in both text files: the total number of tags (on token level) is 35'225 in German, and 61'629 in French. In contrast, the rule-based approach tagged tokens 16'247 times in the German text, and 13'254 times in the French text. In other words, the total amount of all tags for German more than doubled; for French, *Stanford NER* produced about four times more tags than the algorithms, which signifies an increase of around 365 %. Furthermore, the CRF classifiers produced a substantial number of NE tags for tokens that were not at all tagged by the algorithms (referred to as 'Stanford only' in the table). The number of corresponding tags, i.e. the tokens which were assigned to the same NE tag by both *Stanford NER* and the algorithms, was relatively small (7'489 corresponding tags in German, and 5'711 in French). In a nutshell, *Stanford NER* agreed to approximately 43–46 % of all the NE tags suggested by the rule-based algorithms.

	German	French
<b>Total <i>Stanford</i> tags:</b>	35'225	61'629
<b>Total rule-based tags:</b>	16'247	13'254
<b>Corresponding tags:</b>	7'489	5'711
<b><i>Stanford</i> only:</b>	25'556	53'921

Table 11: Number of tags generated by *Stanford NER* and by the search algorithms

I also compared the class tag distribution produced by *Stanford NER* to the one by the algorithms. Table 12 illustrates that in German and French, both approaches most frequently used the PER and LOC tags, followed by ORG. Paradoxically, the rule-based search for French only tagged 5 tokens with MISC, whereas *Stanford NER* used this tag for 27'197 tokens. The MISC class appeared to be the most problematic in general, and the *Stanford* tagger seemed to "overuse" this label in case of insecurity. For this reason, I decided to analyze the MISC class separately.

	Class	Rule-based	<i>Stanford</i>
<b>German</b>	PER	10'071	20'604
	LOC	4'522	8'377
	ORG	912	5'440
	MISC	742	804
<b>French</b>	PER	2'385	8'584
	LOC	10'746	24'028
	ORG	118	1'819
	MISC	5	27'197

Table 12: Distribution of NE class tags

Correctly tagging the SSRQ data is, as I would claim, an extremely difficult task for a modern NER system. Apart from the spelling variation and absence of standardized orthography, the volume contents are multilingual. Big chunks of text in the French volumes are written in a historical variant of German or in Latin, which is why the *Stanford NER* performed worse than in the German volumes. For multilingual documents, it would be desirable to have a language-independent classifier. As can be seen in the excerpt in section 5.1.1.1 above, the CRF models correctly recognized NEs in the parts written in modern German, but failed to classify NEs such as *Winterhalten* (place name). On the other hand, it recognized *Cartuser* (a religious order), a NE which is not mentioned in the volume index.

In order to gain insight into the results of the *Stanford* classifiers and to perform an error analysis, I randomly selected<sup>6</sup> 150 cases of non-corresponding tags per language (50 of each class, except for MISC). These non-corresponding tags included all cases in which the rule-based algorithms disagreed with *Stanford NER*: i.e. whenever they generated a different NE tag for a token, or none ('O'). For all three classes in each language, I manually annotated the 50 selected tokens to create a small gold-standard, and then counted the correct and wrong tags produced by *Stanford NER*. Table 13 summarizes the results of the evaluation.

'Compensations' is a subset of the true positives, and refers to cases where *Stanford NER* was able to compensate for missing tags (NEs not captured by the algorithm). In other words, I counted those tokens for which *Stanford NER* produced a correct class tag, but which were not detected by algorithms at all. 'Borderline' is a subset of the false positives. It denotes tags which did not correspond to my gold-standard, but were not exactly wrong either.

Class	German			French		
	PER	LOC	ORG	PER	LOC	ORG
<b>True Positives</b>	21	41	7	18	16	8
Compensations	21	22	6	16	13	5
<b>False Positives</b>	29	9	43	32	34	42
Borderline	18	0	13	4	2	8
<b>Precision (%)</b>	42	82	14	36	32	16

Table 13: Precision of *Stanford NER* evaluated for three classes

Table 13 illustrates that – as expected – precision was generally low, except for the German LOC class which achieved 82 %. False Positives in this class occurred only due to ambiguous phrases, such as 'im Urbar' (referring to a an urbarium, i.e. a historical document). In the German PER class, there were various borderline cases of place names being tagged as person names: this happened if the place was preceded by a person name and the preposition 'von', e.g. 'Walther von Burgdorff'<sup>7</sup>. The ORG class performed worst, since it was assigned to numerous uppercase tokens. Examples are common nouns such as 'Zolldirektion', 'Gemeinde', 'Schule', and acronyms such as 'BvB'. I counted some of them to the borderline category because the tokens actually denote organizations, but do not refer to individual entities.

<sup>6</sup>I did not completely randomize the selection: since NEs can consist of several tokens, I ensured that multi-token NEs stayed together.

<sup>7</sup>If the PER tags of these place names were not considered as errors, precision would increase from 42 % to 78 %.

In the French text, *Stanford NER* performed especially bad because of multilingual contents and other unknown segments. The PER tag was frequently assigned to German nouns (e.g. 'Guggisberg', 'Vogt'), and sometimes to place names. Most False Positives in the ORG and LOC class occurred with historical spelling variants of function words (often upper-cased, e.g. 'Cecy', 'Ainsy'), Latin tokens ('Postremo', 'Ciceronis'), and Roman numerals. Borderline cases included ambiguous NEs such as 'Saint-Gal' (both place and saint's name) or 'Officia Politica' (which was tagged ORG).

Comparisons between the MISC tags distributed by *Stanford NER* and the rule-based approach were difficult to make. The algorithms applied this class for saints' names only, while the *Stanford NER* classifiers tended to use it for strings that could potentially be NEs. In the German corpus, the classifiers mostly used the class for adjectival forms: out of 50 MISC tags, 39 were attributed to tokens such as 'Zürcher', 'rheinischen' or 'Luzerner'. However, I was not able to identify any pattern or logic in the distribution of the MISC tags in the French texts, and the results were extremely noisy. Among the tagged tokens were function words, acronyms, Roman numerals, German nouns, and all-caps strings.

The low percentages seem to suggest that modern NER systems are not at all suited for tagging historical texts. However, I would like to underline the category 'Compensate' in table 13. *Stanford NER* classified many tokens which remained untagged by the rule-/index-based search; in fact, 74.8 % of all correctly tagged tokens were not found by the algorithms. With some investigations it became evident that index editors frequently stopped listing page/line indicators after a certain amount of pages, or if the name occurs too frequently. Also, if the name appears in titles or footnotes, the indicators were often omitted. Various NEs found by *Stanford NER* were not given in any indices at all, including *Heilbrunn*, *Augustin*, and *Montbret*. Lastly, it was able to capture a number of NEs that were not tagged by the algorithms, since they did not pass the threshold in the approximate search.

Overall, the comparisons between rule-based and ML-based output were not too surprising. Traditionally, rule-based systems make use of relevant lists, and are thus "custom-tailored" to a specific corpus or task. ML systems, on the other hand, are prone to a certain degree of over-generalization with unknown data. My conclusion is that despite their low precision, the *Stanford* classifiers were indeed able to counterbalance some of the shortcomings of the rule-based approach, and at least partly compensate for the lower recall. Having established the up- and downsides of both approaches, I was aiming at combining them in an ideal way so as to produce output which can be used to build an own, optimized NER model.

## 5.2 CRF classifier training with *Wapiti*

As stated before, *Stanford NER* is a classifier predicated on CRF – a typical statistical modeling method used for **sequential** data. Sequential means that the order of tokens in the text is crucial for the correct labeling. Hence, the goal of ML is to calculate probabilities based on the training data, and using them to predict the best label for the current sequence. Probabilities are calculated according to given features, which in turn are extracted from the ”whole” sequence of evidences (i.e. not only in relation to the current token). CRFs find applications in NLP tasks such as shallow parsing and POS-tagging, but also in NER. There is a wide range of open-source ML systems available that allow the training of own classifiers (including the *Stanford NER* software). However, for building a CRF model using my pre-processed SSRQ data, I decided to use the open-source ML toolkit *Wapiti* developed by LIMSI-CNRS<sup>8</sup> (Lavergne et al., 2010). We have already tested this toolkit on the SSRQ texts in an earlier seminar project with promising results.

*Wapiti* is a platform-independent ML package for the implementation of linear-chain CRFs, which allows the labeling of sequential data with discriminative models<sup>9</sup>. *Wapiti*’s main assets rest upon the fact that it includes the most important training and optimization algorithms, as well as various regularization methods to avoid over-fitting. The standard training algorithm is quasi-newton optimization (L-BFGS) with L2<sup>10</sup> regularization. Another feature of the toolkit is the ability for multi-threading and N-best-tagging. According to (Constant and Tellier, 2012, 505), *Wapiti* is one of the best known and most efficient implementations of linear CRFs.

### 5.2.1 Data and features

In the previous chapter, I have discussed the results of *Stanford NER* and the rule-based algorithms, and claimed that the approaches have complementary strength and weaknesses. Since the rule-based algorithms are based on the indices, their results were more precise, whereas *Stanford NER* produces more tags and achieved better recall. I therefore developed a so-called **silver-standard** version of the data by automatically combining the output of the two approaches in each language.

---

<sup>8</sup>*Laboratoire d’Informatique pour la Mécanique et les Sciences de l’Ingénieur*;  
cf. <https://wapiti.limsi.fr/>.

<sup>9</sup>*Wapiti* also supports maximum entropy Markov models and softmax/multinomial logistic regression.

<sup>10</sup>By default, the value for L2 regularization is 0.0001.

For this purpose, I unified the IO-tagged corpora to a tab- and newline-separated file with four columns, as illustrated in table 14 below.

Token	MyTag	SNERTag	SilverTag
Allso	O	O	O
hatt	O	O	O
Petter	O	I-PER	I-PER
Måder	O	I-PER	I-PER
,	O	O	O
amman	O	I-MISC	I-MISC
ze	O	O	O
Langentan	I-LOC	I-PER	I-LOC
,	O	O	O
geredt	O	O	O

Table 14: Tokenized, tab-separated data format with four columns

Each line is composed of the token, the tag given by the rule-based approach ('MyTag'), and the tag given by *Stanford NER* ('SNERTag'). The fourth column contains the silver-standard tag ('SilverTag'), which is either 'MyTag' or 'SNERTag', depending on the following rule:

---

**Algorithm 2** Decide which tag to use for silver-standard

---

```

for line in 4c_corpus do
  if MyTag = O & SNERTag ≠ O then return SNERTag
  else
    if MyTag ≠ O then return MyTag
    end if
  end if
end for

```

---

In other words, the rule-based NE tags always had priority over the *Stanford* tags, since they are expected to be more precise. The *Stanford* tag only applied in case there was no rule-based tag (i.e. 'MyTag' = O). Logically, if both 'MyTag' and 'SNERTag' were O, then so was 'SilverTag'. The resulting four-column format could be directly used as input for training and testing with *Wapiti*. For each language, I split the silver-standard file into a three sets: the first 80 % of the file were used for training, the following 10 % for development, and the last 10 % for testing<sup>11</sup>.

---

<sup>11</sup>Since *Wapiti*'s sequence error rate operates on sentence level, I manually restored sentences at file beginning or end in case the file splitting process ripped them apart.

In the training phase, I built three different classifiers per language using the default settings for *Wapiti*'s training mode. *Wapiti* automatically performed fine-tuning on the given development set. In every run, a different template of feature patterns was used. Two of them extracted evidence from the tokens (i.e. from column 1 in the corpus), whereas the third one also included evidence from the tags (i.e. columns 2 and 3):

**Template B.1** defined a set of (case-sensitive) unigram features within a context window of two tokens<sup>12</sup>, as well as a binary feature regarding the upper- or lower-casing of the current token. It also considered the first and last three characters of the current token.

**Template B.2** contained all aforementioned features, but completed them with 12 additional ones. These included features regarding first and last two characters in the current token, last four characters in the current token, and upper-casing of previous and current token. Moreover, there were five binary features returning 'True' if the current or previous token contained punctuation.

**Template B.3** was composed of all features from template 1, but additionally transformed the NE tag evidence (within a context window of 2) into features.

The labeling mode was called with the parameters `-check` (since the test set was already labeled and in 4-column format) and `-post` (for activating posterior decoding instead of the classical Viterbi decoding, which allows to output normalized score for sequences and labels).

## 5.2.2 Evaluation

### 5.2.2.1 Silver-standard comparison

In the evaluation phase, I decided to first compare the results of the trained classifiers to the silver-standard test set in each language. I excluded, however, the third template B.3 at this point, as its features were based on extracted evidence from the tags. As mentioned, these tags were consistently distributed based on algorithm 2 above, and the classifier – when presented with columns 2 and 3 – would almost always be able to predict the silver-standard tags<sup>13</sup>. On the other hand, testing the

---

<sup>12</sup>Corresponding to five different unigram features: current token [0], preceding token [-1], token before preceding, [-2]; following token [+1] and token after following [+2]. Cf. appendix B.

<sup>13</sup>In other words, evaluating the model trained with template B.3 against the silver-standard test set will only reveal how well it can "imitate" the heuristic. To verify this, I conducted a test run and found that, indeed, F-scores were around 99 to 100 % for all classes.

classifiers built with templates B.1 and B.2 shed light on their ability to predict silver-standard tags based on the token information only. The results for both of these two classifiers were extremely similar, and for some classes even identical. Moreover, both had a token error rate of around 4 %. For this reason, table 15 below shows the performance of classifier B.2 only, giving precision, recall and F-score for all labels.

	German			French		
	Pr	Rc	F1	Pr	Rc	F1
<b>O</b>	98	99	98	95	99	97
<b>I-PER</b>	86	70	77	81	61	69
<b>I-LOC</b>	40	21	27	70	46	55
<b>I-ORG</b>	64	82	72	51	64	57
<b>I-MISC</b>	85	6	12	54	43	48

Table 15: Label statistics of the *Wapiti* model B.2; numbers in percentages

The very high F-scores for the O label imply that the model was good at predicting what is **not** a NE; however, correctly classifying them seemed to be more complex. As expected, the MISC class was most problematic, and resulted in the lowest precision and recall, while PER and ORG were easier to classify. Interestingly, the LOC class achieved lower than ORG, mostly due to lower recall. Apparently, it was relatively difficult to predict the silver-standard tags by only the string information from the tokens. The fact that both B.1 and B.2 were similar indicates that generally, features considering letters and punctuation in a token does not contribute much to performance.

### 5.2.2.2 Gold-standard comparison

Simply put, testing the trained models on the silver-standard test sets demonstrated their predicting ability when using evidence from the token sequences. However, in order to perform a meaningful evaluation, I decided to create two small gold-standard test sets by manually annotating a selected corpus excerpt in both languages. Since the annotation is a highly time-consuming task<sup>14</sup>, I limited the excerpts to a token number equivalent to ten book pages (that is, ten pages in the printed volumes). For both German and French, I randomly selected a volume and

<sup>14</sup>Without fundamental knowledge of the historical language varieties, there is always the factor of uncertainty, and it was necessary to frequently consult the indices for verification. Annotation workload in total was approximately 6 hours.

a starting page. From this page onward, I extracted ten consecutive pages, and annotated them while adhering to the line-separated 4-column format. The resulting excerpts contained 5'647 tokens in German, and 4'994 in French<sup>15</sup>. I tested all three models on the gold-standard test set. Unfortunately, due to the small size of the gold-standard, the only NE classes that occurred were PER, LOC and MISC in both languages. MISC was very rare (less than 10 occurrences), which is why table 16 gives results for PER and LOC only. The percentages for the O class were extremely high again (between 96 and 100 % F-score in both German and French), so I decided not to include them in the table.

		German			French		
		Pr	Rc	F1	Pr	Rc	F1
<b>B.1</b>	<b>I-PER</b>	88	71	79	82	85	83
	<b>I-LOC</b>	33	10	16	38	80	51
<b>B.2</b>	<b>I-PER</b>	<b>91</b>	67	77	76	89	82
	<b>I-LOC</b>	39	10	16	40	78	53
<b>B.3</b>	<b>I-PER</b>	<b>91</b>	85	<b>88</b>	<b>86</b>	87	<b>87</b>
	<b>I-LOC</b>	63	<b>86</b>	73	25	<b>99</b>	40

Table 16: Statistics for two labels of all *Wapiti* models; numbers in percentages

The first model trained on features from template B.1 already achieved reasonably good results for the PER class, with F-scores of around 80 % in both German and French. Precision and recall were similar, except for German, where it was slightly lower (71 %). The LOC class was very problematic, especially in German. In French, recall for this class was acceptable (80 %), but precision was not (merely 38 %). Comparing these results to the ones of the model trained with template B.2 revealed that the numerous additional token features from B.2 did not contribute much to performance. Even though there were more features considering the characters and punctuation in token sequences, there were no significant improvements in the overall results. Precision and recall for the LOC labels slightly improved, and precision of the German PER label even climbed over 90 %. However, the features from template B.2 were generally not better suited to predict the gold-standard. My claim is that the spelling and orthographic variation in historical texts is simply too extensive, so that no inferences can be made from only looking at token strings. Upper-/lower-casing and characters of a token are highly variable, and thus do not allow much inspection regarding its NE class. Models B.1 and B.2 had an average token error rate of about 4.4 % in German and 5.7 % in French.

<sup>15</sup>A test set traditionally comprises around 5 to 10 % of the training size, which is of course much bigger than my selection.

In the evaluation of model B.3 against the gold-standard, I was interested in the predictive power of the classifier if presented with the tags by *Stanford NER* and the search algorithms. In that case, the classifier did not only make the predictions based on token information, but also based on the decisions made by the two NER systems. As expected, this led to better results over all classes, and in both languages. PER was again easiest to classify, leading to F-scores of 88 % in German, and 87 % in French. Although the classifiers achieved very good recall for LOC in both languages (86 % and 99 %), precision was rather low, resulting in F-scores of 73 and 40 % respectively. It could be argued that person names are indeed more predictable than other NEs. For instance, the context in which person names appeared seem to be more revealing and consistent, and both *Stanford NER* and the rule-based algorithm tended to agree about whether a string is a person name or not. Probably due to the ambiguity of person and place names (as in 'Walther von Burgdorff'), the LOC class was less transparent, and thus harder to classify. Lastly, I believe that the multilingual aspect of the data diminished overall performance of the classifiers.

Even though the gold-standard test set was very small, it was possible to obtain promising intermediate results and some important insights. The classifiers were not only trained on a CRF system designed for tagging modern languages, but partly on a system relying on handcrafted rules; naturally, both approaches have their flaws. It should be kept in mind that all processes which lead to the final result – including the corpus compilation from the XHTML files, the extraction and processing of NEs in the indices, as well as the detection and annotation task – incorporated potential errors from the previous processes. Therefore, one should refrain from making high demands on a ML classifier built upon these data.

It should also be taken into account that the classifiers could not resort to other linguistic features such as morphological information, lemma or POS, and evidence was only extracted from sequences of strings. One option to improve the classifiers' performance is trying to incorporate linguistic information into the corpora. However, "since older forms of the language [...] are increasingly different from the modern language, language tools need to be adapted or built from scratch in order to deal successfully with these historical language stages" (Borin and Forsberg, 2011, 42). Another option would be the normalization of the data before training. In future projects, I would like to explore these possibilities, while aiming at building more generalized models able to perform NER in historical texts of different size, genre, and language varieties.

# 6 Conclusion

## 6.1 Summary

The work described in this thesis pursued the following three main objectives: first, pre-processing and compiling a multilingual collection of digitized historical texts, second, extracting NEs and relevant information from secondary sources, and third, investigating different systems for the automatic recognition and annotation of the entities in the pre-processed corpus.

The first objective was achieved by parsing, re-structuring and cleaning up XHTML files of selected SSRQ volumes written in medieval German and French. In the course of this task, I have highlighted the general complexities of automatically digitized and converted historical data, and illustrated how OCR software can be used for text reconstruction. I have also described the process of adapted sentence splitting and tokenization in order to achieve a well-structured and valid XML output format. Although the resulting data set is, by definition, not a representative corpus (in the absence of an explicitly pre-defined time span, genre and spatial dimension), it is still a comprehensive and multi-purpose collection of cultural heritage documents. Such documents have "numerous usages in the various branches of historical research: some of these are related to the task of revealing general trends and patterns about a historic period or personality while others focus on discovering specific [...] pieces of information" (Berzak et al., 2011, 207). Now that the digitized primary sources are cleaned, tokenized and enumerated, it is possible to utilize them for various other research projects in the humanities.

The second and most substantial part of my thesis was focused on the extraction and production of names, variants and page indicators from the volume indices. This involved defining particular regular expressions and rules so as to filter only relevant information from the index entries. The task of producing spelling variants out of fragmented or ill-formatted strings emphasized once more the importance of consistent input data. Furthermore, I demonstrated the usage of approximate string matching techniques with the goal to correctly detect the resulting NEs in

the texts. Due to the fact that in the historical texts, orthography and the language itself was not standardized, finding ideal thresholds for similarity turned out to be very complicated. I have therefore concluded that the lexicon- and rule-based NER task was aimed at better precision at the expense of recall. The final products of this process included an extensive standoff-annotation file in XML format, as well as a NE database reflecting the contents of the indices. (Linde and Mitmann, 2013, 244) carried out a similar project for pre-annotating historical texts, and described it in the following terms:

[This project] is distinguished by the extent to which it automates existing data. Not only are the texts themselves digitized on the basis of existing sources, but the digitization of [...] glossaries, and the exploitation of their data in the annotation of medieval texts, constitute a time-saving innovation that helps focus on the specific problems of annotation. [...] The approach outlined here allows for an effective and efficient creation of large text corpora.

I would like to argue that this statement holds true for the approach in my thesis as well, and that the applications of the resulting database go beyond NER and annotation. For instance, the NE database holds the potential for the creation of Semantic Web Ontologies<sup>1</sup> and Open Linked Data projects, data visualization, or historical linguistic research (e.g. concerning spelling variation in proper names).

Ultimately, in the third and last part of my thesis, I conducted experiments dealing with the application of ML systems to the pre-processed data. I first tested performance of a modern language NE tagger, *Stanford NER*, when applied to the German and French texts. This revealed that *Stanford NER* was complementary to the rule-based algorithms insofar as it achieved higher recall, and produced more tags. The second experiment involved training own classifiers for each language with the CRF toolkit *Wapiti* on the combined data. I developed three different models based upon three sets of features, and partially tested them on silver- and gold-standard data sets. I have pointed out that classification is increasingly difficult for a ML model if the evidence can be extracted from tokens only, and no other linguistic information is available. Figure 14 depicts a flowchart including the major processes and deliverables that were established in the course of this thesis.

---

<sup>1</sup>In fact, the *Swiss Law Sources Foundation* has already launched online thesauri for historical persons, places and organizations; cf. <https://www.ssrq-sds-fds.ch/persons-db/>.

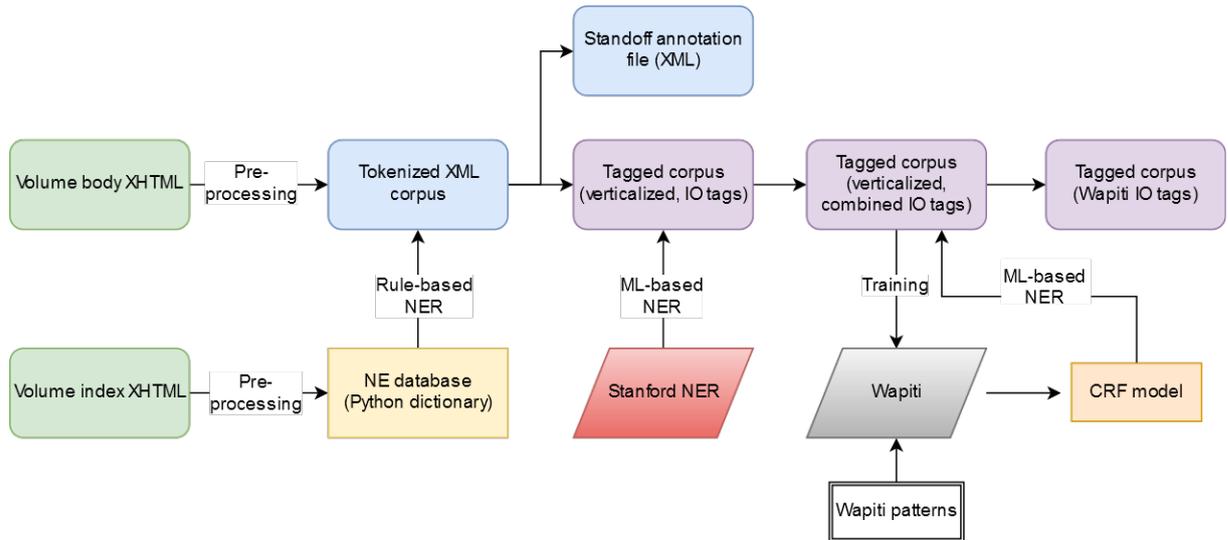


Figure 14: Thesis project pipeline

## 6.2 Outlook

This thesis project has laid the groundwork for future investigations and improvements in the context of historical data in general, and specifically for the SSRQ data. One of the next goals is to further upgrade the XHTML cleanup and name extraction processes, as well as to refine the approximate search algorithms. Related to that, I also want to research into the possibilities and limitations of the different string similarity measures. An important topic which needs to be addressed in the future is normalization: it would be, for example, desirable to follow the lead of Hauser et al. (2007) who have identified and collected the most common phonological, graphical and dialectal variations of Early New High German. It would also be beneficial to analyze the impact of multilinguality and language mixing in historical texts, especially in the context of ML. Once the NER pipeline has been optimized, I would like to proceed to the task of NE grounding, disambiguation and co-reference resolution.

# References

- K. Adam, R. Siva, P. Jan, and P. Avinesh. A corpus factory for many languages. *Proceedings of LREC 2010*, 2010.
- S. T. Aguilar, X. Tannier, and P. Chastang. Named entity recognition applied on a data base of medieval latin charters. the case of chartae burgundiae. In *Proceedings of the 3rd HistoInformatics Workshop on Computational History co-located with Digital Humanities conference (DH 2016)*, pages 67–71, July 2016. URL [http://ceur-ws.org/Vol-1632/paper\\_9.pdf](http://ceur-ws.org/Vol-1632/paper_9.pdf).
- A. Auer and T. Fairman. *New Methods in Historical Corpora*, volume 3, chapter 4: Letters of artisans and the labouring poor (England, C. 1750-1835), pages 77–93. Narr, Tübingen, 2013. ISBN 978-3-8233-6760-4.
- D. Benikova, C. Biemann, and M. Reznicek. Nosta-d named entity annotation for german: Guidelines and dataset. In *LREC*, pages 2524–2531, 2014.
- Y. Berzak, M. Richter, C. Ehrler, and T. Shore. *Language Technology for Cultural Heritage. Selected Papers from the LaTeCH Workshop Series*, chapter 11: Information Retrieval and Visualization for the Historical Domain, pages 197–212. Theory and Application of Natural Language Processing. Springer, Berlin/Heidelberg, 2011.
- S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O’Reilly UK Ltd., 2010. ISBN 0596516495.
- M. Bollmann. (semi-)automatic normalization of historical texts using distance measures and the norma tool. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2), Lisbon, Portugal*, 2012.
- L. Borin and M. Forsberg. *Language Technology for Cultural Heritage. Selected Papers from the LaTeCH Workshop Series*, chapter 3: A Diachronic Computational Lexical Resource for 800 Years of Swedish, pages 41–61. Theory

- and Application of Natural Language Processing. Springer, Berlin/Heidelberg, 2011.
- N. Bubenhofer, M. Volk, F. Leuenberger, and D. Wüest. Text+berg-korpus (release 151v01). XML-format, 2015. Digitale Edition des Jahrbuch des SAC 1864-1923, Echo des Alpes 1872-1924, Die Alpen, Les Alpes, Le Alpi 1925-2014, The Alpine Journal 1969-2008.
- M. Burghardt and C. Wolff. Stand off-annotation für textdokumente: Vom konzept zur implementierung (zur standardisierung?). *Proceedings of the Biennial GSCL Conference*, pages 53–59, 2009.
- K. Byrne. Nested named entity recognition in historical archive text. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 589–596. IEEE, 2007.
- K.-U. Carstensen, C. Ebert, C. Ebert, S. Jekat, H. Langer, and R. Klabunde. *Computerlinguistik und Sprachtechnologie: Eine Einführung*. Springer-Verlag, 2010.
- P. Christen. A comparison of personal name matching: Techniques and practical issues. In *Proceedings of the IEEE Sixth Data Mining Workshop (ICDMW 06)*, pages 290–294, Australian National University, 2006. IEEE.
- M. Constant and I. Tellier. Evaluating the impact of external lexical resources into a crf-based multiword segmenter and part-of-speech tagger. In *8th International Conference on Language Resources and Evaluation (LREC'12)*, pages 646–650, 2012.
- H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.
- S. Ebling, R. Sennrich, and D. Klaper. Digging for names in the mountains: Combined person name recognition and reference resolution for german alpine texts. In *Language and Technology Conference*, pages 189–200. Springer, 2011.
- A. Ernst-Gerlach. *Retrievalmethoden für historische Korpora mit nicht standardisierten Schreibweisen*. PhD thesis, Universität Duisburg-Essen, Fakultät für Ingenieurwissenschaften und Informatik, 2013.

- A. Ernst-Gerlach and T. Pilz. Search methods for documents in non-standard spelling. Talk at the Workshop on Historical Text Mining, Lancaster, U.K., July 2006. URL [http://www.is.inf.uni-due.de/bib/pdf/talks/Ernst\\_Pilz\\_06ta.pdf](http://www.is.inf.uni-due.de/bib/pdf/talks/Ernst_Pilz_06ta.pdf).
- F. Farooq and Y. Al-Onaizan. Effect of degraded input on statistical machine translation. In *2005 Symposium on Document Image Understanding Technology*, pages 103–108. UMIACS, University of Maryland, Nov. 2005. URL <https://lamp.cfar.umd.edu/meetings/Proceedings/SDIUT05.pdf>.
- M. Faruqui and S. Padó. Training and evaluating a german named entity recognizer with semantic generalization. In *Proceedings of KONVENS 2010*, Saarbrücken, Germany, 2010.
- J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.
- B. A. Forouzan. *Data Communications and Networking*. McGraw Hill Higher Education, 4. edition, 2007.
- W. A. Gale, K. W. Church, and D. Yarowsky. One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language*, pages 233–237. Association for Computational Linguistics, 1992.
- C. Grover, S. Givon, R. Tobin, and J. Ball. Named entity recognition for digitised historical texts. In *LREC*. Citeseer, 2008.
- F. Hartweg and K.-P. Wegera. *Frühneuhochdeutsch: Eine Einführung in die deutsche Sprache des Spätmittelalters und der frühen Neuzeit*. Walter de Gruyter, 2005.
- A. Hauser, M. Heller, E. Leiss, K. U. Schulz, and C. Wanzeck. Information access to historical documents from the early new high german period. In *Dagstuhl Seminar Proceedings*, pages 147–154. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2007. URL <http://drops.dagstuhl.de/opus/volltexte/2007/1057/pdf/06491.HellerMarkus.Paper.1057.pdf>.
- S. Höfler and M. Piotrowski. Building corpora for the philological study of swiss legal texts. *JLCL*, 26(2):77–88, 2011.

- R. Holley. How good can it get? analysing and improving ocr accuracy in large scale historic newspaper digitisation programs. *D-Lib Magazine*, 15(3/4), Mar. 2009.
- N. Indurkha and F. J. Damerou. *Handbook of Natural Language Processing, Second Edition*. Chapman and Hall/CRC, 2 edition, 2010. ISBN 978-1-4200-8593-8.
- S. Kempken. *Bewertung historischer und regionaler Schreibvarianten mit Hilfe von Abstandsmaßen*. phdthesis, University of Duisburg-Essen, Dec. 2005.
- S. Kempken, W. Luther, and T. Pilz. Comparison of distance measures for historical spelling variants. In *IFIP International Conference on Artificial Intelligence in Theory and Practice*, pages 295–304. Springer, 2006.
- T. Kiss and J. Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.
- G. Kondrak. N-gram similarity and distance. In *International Symposium on String Processing and Information Retrieval*, pages 115–126. Springer, 2005.
- L. Kornelsen, U. Lucke, D. Tavangarian, M. Waldhauer, and N. Ossipova. Strategien und werkzeuge zur erstellung multimedialer lehr-und lernmaterialien auf basis von xml. In *DeLFI*, pages 31–42, 2004.
- M. Krug, F. Puppe, F. Jannidis, L. Macharowsky, I. Reger, and L. Weimer. Rule-based coreference resolution in german historic novels. *on Computational Linguistics for Literature*, page 98, 2015.
- G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *Proceedings of NAACL-2016, San Diego*, June 2016.
- T. Lavergne, O. Cappé, and F. Yvon. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. Association for Computational Linguistics, July 2010. URL <http://www.aclweb.org/anthology/P10-1052>.
- J. L. Leidner, G. Sinclair, and B. Webber. Grounding spatial named entities for information extraction and question answering. In *Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references*, volume 1, pages 31–38. Association for Computational Linguistics, 2003.

- D. Lin and X. Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 2, pages 1030–1038. Association for Computational Linguistics, 2009.
- S. Linde and R. Mitmann. *New Methods in Historical Corpora*, volume 3, chapter 16: Old German reference corpus: digitizing the knowledge of the 19th century. Automated pre-annotation using digitized historical glossaries, pages 235–246. Narr, Tübingen, 2013.
- C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.
- C. Meyer and G. Wilde. Nicht wörtlich genommen—schreibweisentolerante suchroutinen in dbase implementiert. *c't Magazin für Computer und Technik*, 10:126–131, 1988.
- C. Neudecker. An open corpus for named entity recognition in historic newspapers. In N. Calzolari, editor, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, May 2016. European Language Resources Association (ELRA). ISBN 978-2-9517408-9-1.
- J. Nothman, J. R. Curran, and T. Murphy. Transforming wikipedia into named entity training data. In *Proceedings of the Australian Language Technology Workshop*, pages 124–132, 2008.
- J. Nothman, N. Ringland, W. Radford, T. Murphy, and J. R. Curran. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151–175, 2013.
- R. Perkuhn, H. Keibel, and M. Kupietz. *Korpuslinguistik*. LIBAC/UTB. Fink, Paderborn., 2012.
- T. Pilz, A. Ernst-Gerlach, S. Kempken, P. Rayson, and D. Archer. The identification of spelling variants in english and german historical texts: manual or automatic? *Literary and Linguistic Computing*, 23(1):65–72, 2008.
- M. Piotrowski. Document conversion for cultural heritage texts: Framemaker to html revisited. In *Proceedings of the 10th ACM symposium on Document engineering*, pages 223–226. ACM, 2010.
- M. Piotrowski. Natural language processing for historical texts. *Synthesis Lectures on Human Language Technologies*, 5(2):1–157, 2012.

- M. Piotrowski and C. Senn. Harvesting indices to grow a controlled vocabulary: towards improved access to historical legal texts. In *Proceedings of the 6th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 24–29. Association for Computational Linguistics, 2012.
- L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics, June 2009. URL <http://cogcomp.cs.illinois.edu/papers/RatinovRo09.pdf>.
- D. Rechtsquellenstiftung des Schweizerischen Juristenverbandes, editor. *Die Rechtsquellen der Stadt Burgdorf und ihrer Herrschaften und des Schultheißenamts Burgdorf*, volume SSRQ BE II/9 of *Sammlung Schweizerischer Rechtsquellen*. Verlag Sauerländer Aarau, Switzerland, 1995. Prepared by Anne-Marie Dubler.
- D. Rechtsquellenstiftung des Schweizerischen Juristenverbandes, editor. *Stadt und Territorialstaat Luzern: Satzungen und andere normative Quellen (1426–1460)*, volume SSRQ LU I/2 of *Sammlung Schweizerischer Rechtsquellen*. Schwabe Verlag Basel, Switzerland, 2004. Prepared by Konrad Wanner.
- D. Rechtsquellenstiftung des Schweizerischen Juristenverbandes, editor. *Die Freien Ämter III: Die Ämter Meienberg und Merenschwand*, volume SSRQ AG II/10 of *Sammlung Schweizerischer Rechtsquellen*. Schwabe Verlag Basel, Switzerland, 2009. Prepared by Anne-Marie Dubler and Jean Jacques Siegrist.
- C. N. d. Santos and V. Guimarães. Boosting named entity recognition with neural character embeddings. In *Proceedings of The Fifth Named Entities Workshop (NEWS)*, 2015. URL <https://arxiv.org/pdf/1505.05008.pdf>.
- O. Scrivner and S. Kübler. Tools for digital humanities: Enabling access to the old occitan romance of flamenca. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, pages 1–11, 2015.
- S. Sekine, K. Sudo, and C. Nobata. Extended named entity hierarchy. In *LREC*, 2002.
- R. Smith, D. Antonova, and D.-S. Lee. Adapting the tesseract open source ocr engine for multilingual ocr. In *Proceedings of the International Workshop on Multilingual OCR*, page 1. ACM, 2009.

- R. W. Smith. History of the tesseract ocr engine: what worked and what didn't. In *DRR*, 2013.
- E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, volume 4, pages 142–147. Association for Computational Linguistics, 2003.
- N. Ueffing and H. Ney. *Bayes Decision Rules and Confidence Measures for Statistical Machine Translation*, chapter 7, pages 70–81. Springer, Berlin, Heidelberg, 2004.
- M. Volk and S. Clematide. Learn-filter-apply-forget. mixed approaches to named entity recognition. In *6th International Workshop on Applications of Natural Language for Informations Systems*, volume 1, pages 153–163, Madrid, Spain, 2001.
- M. Volk, N. Bubenhofer, A. Althaus, and M. Bangerter. Classifying named entities in an alpine heritage corpus. *KI*, 23(4):40–43, 2009.
- M. Volk, N. Bubenhofer, A. Althaus, M. Bangerter, L. Furrer, and B. Ruef. Challenges in building a multilingual alpine heritage corpus. In *LREC*, 2010.

# Curriculum Vitae

## Personal Details

Yvonne Gwerder

Rösslimatt 10

6423 Seewen SZ

yvonne.gwerder@uzh.ch

## Education

- 2010–2014 B.A. studies in English Literature and Linguistics and Computational Linguistics at the University of Zurich
- 2014–2015 Translational studies at the SAL School of Applied Linguistics, Zurich
- since 2014 M.A. studies in Multilingual Text Analysis at the University of Zurich

## Work experience

- 2012 Research assistant in the bilingual corpus project *Bilingwis*, Institute of Computational Linguistics, University of Zurich
- 2014 Research assistant for Sentiment Analysis, Institute of Computational Linguistics, University of Zurich
- 2015 Research assistant for the normalization and annotation of historical texts, Institute of Computational Linguistics, University of Zurich
- 2015 Terminology internship at *Roche Diagnostics International*, Rotkreuz
- 2016 Student assistant for database pre-processing and deduplication, Swiss Economic Institute (KOF), ETH Zurich

# A Python Scripts

- [A1] `corpus_preproc_*.py`  
Corpus pre-processing for the single volumes, as outlined in chapter 3. There is a separate script for each volume (`*` being a placeholder for the volume codes `ag`, `be`, `lu`, `fr`, `vd`, and `vd2`).
- [A2] `clean_register.py`  
Removal of duplicate IDs in the indices, as outlined in section 4.1.1.
- [A3] `extract_variants_de.py`  
Script for extracting and generating name variants in the German indices.
- [A4] `extract_variants_fr.py`  
Script for extracting and generating name variants in the French indices.
- [A5] `lookup_de.py`  
Script for NER in German volumes; creates stand-off annotation and IO-tagged corpus.
- [A6] `lookup_fr.py`  
Script for NER in French volumes; creates stand-off annotation and IO-tagged corpus.
- [A7] `cat_files.py`  
Concatenation of OCR-processed pages in order to evaluate them with *ocreval-Union*. It is called with the additional command line argument `'de'` or `'fr'`.
- [A8] `ocr_de.py`  
OCR-processing of all PNG files for volume LU I/2.
- [A9] `ocr_fr.py`  
OCR-processing of all PNG files for volume VD C/2.

**[A10] `splitstanford.py`**

Script for creating a tab- and newline separated text file from the `slashTag`-formatted output produced by *Stanford NER*. It is called with the additional command line argument `'de'` or `'fr'`.

**[A11] `combine.py`**

Script for combining the rule-based and *Stanford NER* tags to a 4-column file. It is called with the additional command line argument `'de'` or `'fr'`.

**[A12] `splitfiles_wapiti.py`**

Splitting the output of [A11] into a training, development and test set ready for use with *Wapiti*. It is called with the additional command line argument `'de'` or `'fr'`. The default values for splitting are 80 % for the training set, and 10 % each for development and test set.

## B *Wapiti* Feature Templates

### B.1 Template 1

```
# current token (case-sensitive)
U:word=%x[0,0]

# token features (context window +/-2)
U:word X=%x[0,0]
U:word L=%x[-1,0]
U:word LL=%x[-2,0]
U:word R=%x[1,0]
U:word RR=%x[2,0]

# upper case of token (binary)
*:starts-upper X=%t[0,0,"^\u"]
# lower case of token (binary)
*:starts-lower X=%t[0,0,"^\l"]

# suffix of 3 characters in token
U:suf-3 X=%m[0,0,"...$"]
# prefix of 3 characters in token
U:pre-3 X=%m[0,0,"^..."]
```

### B.2 Template 2

```
# current word (case-sensitive)
U:word=%x[0,0]

# token features (context window +/-2)
U:word X=%x[0,0]
```

```
U:word L=%x[-1,0]
U:word LL=%x[-2,0]
U:word R=%x[1,0]
U:word RR=%x[2,0]

# suffix of 3 characters in token
U:suf-3 X=%m[0,0,"...$"]
# prefix of 3 characters in token
U:pre-3 X=%m[0,0,"^..."]

# upper case of token (binary)
*:starts-upper X=%t[0,0,"^\u"]
# lower case of token (binary)
*:starts-lower X=%t[0,0,"^\l"]

# Unigram-Feature: first character of current token
U:Pre-1 X=%m[0,0,"^.?"]
# first two characters of current token
U:Pre-2 X=%m[0,0,"^.??.?"]
# last two characters of current token
U:Suf-2 X=%m[0,0,".?.?.$"]
# last four characters pf current token
U:Suf-4 X=%m[0,0,".?.?.?.?.$"]

# Unigram-Feature: True if previous token is upper-cased.
U:Caps? L=%t[-1,0,"\u"]
# True if current token is upper-cased.
U:Caps? X=%t[0,0,"\u"]
# True if current token starts with upper-case letter.
U:BegC? X=%t[0,0,"^\u"]

# True if previous token is punctuation mark.
U:Punc? L=%t[-1,0,"\p"]
# True if current token is punctuation mark.
U:Punc? X=%t[0,0,"\p"]
# True if following token is punctuation mark.
U:Punc? R=%t[1,0,"\p"]
# True if current token contains a punctuation mark.
U:InsP? X=%t[0,0,".\p."]
```

## B.3 Template 3

```
# current token (case-sensitive)
U:word=%x[0,0]

# token features (context window +/-2)
U:word X=%x[0,0]
U:word L=%x[-1,0]
U:word LL=%x[-2,0]
U:word R=%x[1,0]
U:word RR=%x[2,0]

# suffix of 3 characters in token
U:suf-3 X=%m[0,0,"...$"]
# prefix of 3 characters in token
U:pre-3 X=%m[0,0,"^..."]

# upper case of token (binary)
*:starts-upper X=%t[0,0,"^\u"]
# lower case of token (binary)
*:starts-lower X=%t[0,0,"^\l"]

# rule-based features (context window +/-2)
U:tag X=%X[0,1]
U:tag L=%X[-1,1]
U:tag LL=%X[-2,1]
U:tag R=%X[1,1]
U:tag RR=%X[2,1]

# stanfordNER features (context window +/-2)
U:tag X=%X[0,2]
U:tag L=%X[-1,2]
U:tag LL=%X[-2,2]
U:tag R=%X[1,2]
U:tag RR=%X[2,2]
U:tag L/0=%X[-1,2]/%X[0,2]
```

## C Modules and Tools

This list contains **Python** modules (which are not part of the standard library), as well as other tools that were used for the work described in this thesis.

- Python modules:

- *nltk* (<https://www.nltk.org/>)
- *editdistance* (<https://pypi.python.org/pypi/editdistance>)
- *jellyfish* (<https://pypi.python.org/pypi/jellyfish>)
- *pyocr* (<https://pypi.python.org/pypi/pyocr>)
- *PIL* (<https://pypi.python.org/pypi/Pillow>)
- *natsort* (<https://pypi.python.org/pypi/natsort>)

- Tools:

- *PDF Shaper* (<https://www.pdfshaper.com/>)
- *ocrevalUAtion* (<https://github.com/impactcentre/ocrevalUAtion>)
- *Stanford NER* (<https://nlp.stanford.edu/ner/>)
  - \* German model: *'dewac\_175m\_600.ser.gz'*  
([https://www.nlpado.de/~sebastian/software/ner\\_german.shtml](https://www.nlpado.de/~sebastian/software/ner_german.shtml))
  - \* French model: *'french.4class.wp2.ser.gz'*  
(available on the UZH r2d2 server)
- *Wapiti* (<https://wapiti.limsi.fr/>)



## **Selbstständigkeitserklärung**

Hiermit erkläre ich, dass die Masterarbeit von mir selbst ohne unerlaubte Beihilfe verfasst worden ist und ich die Grundsätze wissenschaftlicher Redlichkeit einhalte (vgl. dazu: <http://www.uzh.ch/de/studies/teaching/plagiate.html>).

Schwyz, 28. Juni 2017

.....  
Ort und Datum

.....  
Unterschrift