Bochumer Linguistische Arbeitsberichte 13

Automatic Normalization for Linguistic Annotation of Historical Language Data

Marcel Bollmann

Bochumer Linguistische Arbeitsberichte

Herausgeber: Stefanie Dipper & Björn Rothstein

Die online publizierte Reihe "Bochumer Linguistische Arbeitsberichte" (BLA) gibt in unregelmäßigen Abständen Forschungsberichte, Abschluss- oder sonstige Arbeiten der Bochumer Linguistik heraus, die einfach und schnell der Öffentlichkeit zugänglich gemacht werden sollen. Sie können zu einem späteren Zeitpunkt an einem anderen Publikationsort erscheinen. Der thematische Schwerpunkt der Reihe liegt auf Arbeiten aus den Bereichen der Computerlinguistik, der allgemeinen und theoretischen Sprachwissenschaft und der Psycholinguistik.

The online publication series "Bochumer Linguistische Arbeitsberichte" (BLA) releases at irregular intervals research reports, theses, and various other academic works from the Bochum Linguistics Department, which are to be made easily and promptly available for the public. At a later stage, they can also be published by other publishing companies. The thematic focus of the series lies on works from the fields of computational linguistics, general and theoretical linguistics, and psycholinguistics.

Band 13 (Dezember 2013)

Herausgeber: Stefanie Dipper
Sprachwissenschaftliches Institut
Ruhr-Universität Bochum
Universitätsstr. 150
44801 Bochum

Björn Rothstein
Germanistisches Institut
Ruhr-Universität Bochum
Universitätsstr. 150
44801 Bochum

Erscheinungsjahr 2013
ISSN **2190-0949**

Marcel Bollmann

Automatic Normalization for Linguistic Annotation of Historical Language Data

---

2013

Bochumer Linguistische Arbeitsberichte

(BLA 13)

Contents

List of Tables

List of Figures

Abstract

This paper deals with spelling normalization of historical texts with regard to further processing with modern part-of-speech taggers. Different methods for this task are presented and evaluated on a set of historical German texts from the 15th–18th century, and specific problems inherent to the processing of historical data are discussed. A chain combination using word-based and character-based techniques is shown to be best for normalization, while POS tagging of normalized data is shown to benefit from ignoring punctuation marks. Using these techniques, when 500 manually normalized tokens are used as training data for the normalization, the tagging accuracy of a manuscript from the 15th century can be raised from 28.65% to 76.27%.

1 Introduction[1]

In the domain of corpus linguistics, part-of-speech (POS) annotation is probably the most commonly found type of annotation for large, modern language corpora. It can serve as a starting point for the full syntactic parsing found in treebanks such as the Penn treebank (Marcus et al., 1993), the TIGER corpus (Brants et al., 2002), and Tüba-D/Z (Telljohann et al., 2004). Apart from this, POS annotation can be used for a wide range of practical applications, such as information retrieval, semantic disambiguation (e.g., Wilks and Stevenson, 1998), or distance computations (e.g., Dipper and Schrader, 2008). Gold standard POS annotation also plays an important role for training and evaluation of automatic POS taggers, which in turn can provide valuable assistance in the creation of further annotated resources. Automatic POS tagging of modern language data is a well explored field, commonly achieving accuracies around 97% (Schmid, 1995; Brants, 2000; Schmid and Laws, 2008).

The typical domain of larger corpora such as Tüba-D/Z or TIGER is newspaper texts. One reason for this is the common availability of such texts in relatively large amounts. Another reason is that newspapers are typically written in a neutral, formal style of the standard variety of a language, as opposed to, e.g., literary works, which may be more heavily influenced by the personal writing style of one author. In recent times, there has been a growing interest in more specialized types of corpora, e.g., corpora of SMS texts (Walkowska, 2009) or Twitter messages (Petrović et al., 2010). Corpora of historical data are another example of these specialized resources; projects to create historical corpora have been started for several different languages, e.g., German (Scheible et al., 2011a), Spanish (Sánchez-Marco et al., 2010), or Slovene (Erjavec, 2012). While the creation of a corpus of modern standard language can be heavily aided by tools like POS taggers, the same cannot be said for non-standard varieties, as performance of these tools is typically much worse. As an example, Scheible et al. (2011b) present a study on Early Modern German texts and report an average tagging accuracy of 69.6% for this type of data.

While syntactic and semantic differences between historical and modern language varieties are common, differences in spelling are probably the biggest obstacle for natural language processing (NLP) tools. This is because spelling in historical texts is not only different from modern spelling, but also often inconsistent due to the lack of fixed spelling conventions. These inconsistencies can also arise within a single text written by one author. A common approach to this problem, which is also employed in this paper, is to "normalize" these historical spellings to their modern counterparts. While this method has generated a lot of interest by various research groups lately (e.g.,

[1] This paper is a slightly modified and updated version of my Master's thesis from December 7th, 2012, which was supervised by Prof. Dr. Stefanie Dipper. A brief summary of the main results has also been previously published as a workshop paper (Bollmann, 2013).

Baron et al., 2009; Jurish, 2010; Bollmann et al., 2011; Hendrickx and Marquilhas, 2011; Adesam et al., 2012), no clear "best" algorithm for automatic normalization has been identified yet. Also, comparisons between the various normalization algorithms are rarely performed (Reynaert et al., 2012, is a recent example, though).

Furthermore, there has been less focus on the effect of various normalization methods on POS tagging. While Dipper (2010) and Scheible et al. (2011b) describe an increase in tagging performance when using a normalization layer (rather than the original data), the normalization was created manually in both cases and can therefore be assumed to be correct. However, when using automatic methods, it is not obvious whether tagging accuracy is directly linked to normalization accuracy. This is especially true for strongly inflecting languages like German: minor differences in inflection that arise during normalization are counted as errors there, but can still generate the correct POS tag. Similarly, the POS tag of wrongly normalized words might conceivable be inferred from their syntactic context if the surrounding words have been normalized correctly. Different normalization methods might have different advantages in this regard which cannot be derived from normalization accuracy alone.

This thesis aims to do two things: firstly, to compare different approaches to normalization and evaluate them on different types of historical texts; and secondly, to evaluate the task of POS tagging on historical language data, with a particular focus on the effect that normalization has on tagging accuracy. It will show that there are specific characteristics of normalized texts which affect POS tagging and should be carefully considered when using such an approach. The language of the texts used for the evaluation is German from the Early New High German (ENHG) to the Early Modern period, with sample texts ranging from the 15th to 18th centuries. The modern standard variety of New High German (NHG) is sometimes also referred to as "modern German" here.

Spellings of ENHG and NHG wordforms are compared frequently; for easier identification, the historical wordform and its modern (or, sometimes, automatically normalized) counterpart are often separated by an arrow (e.g., *old* $\rightarrow$ *new*). If glosses are used, the first line in italics will always be the historical text, while the following lines contain gold standard annotations; if an example deviates from this structure, it will be labelled accordingly.

The structure of this paper is as follows: Section 2 presents all corpora that are used in the evaluation, while Section 3 presents different approaches to normalization and evaluates them on historical data. Section 4 discusses part-of-speech tagging by highlighting specific problems of tagging historical data and evaluating tagging performance on original and normalized historical texts. Section 5 presents related work, and Section 6 concludes and discusses possible lines of future research.

2 Corpora

This section describes the corpora used for the normalization and tagging experiments. Section 2.1 presents the Luther corpus that was used for the evaluation of normalization methods. Section 2.2 describes the modern German corpus used to train the POS tagger and to perform tagging experiments on modern data. The remaining sections discuss the corpora used to evaluate normalization, POS tagging, and the combination of both in the context of actual research scenarios: the Anselm corpus in Section 2.3; and the GerManC-GS corpus in Section 2.4.

2.1 Luther corpus

Gold standard data of normalizations of historical texts is not easily available. Such data is required, however, for training and evaluating automatic normalization methods. An earlier study described in Bollmann et al. (2011) tried to address this problem by creating test data from two different versions of the Luther bible. The resulting Luther corpus is briefly described here.

For this corpus, two different versions of the bible were used: the original Early New High German version by Martin Luther from 1545, and a revised modern version of it.[2] Both versions were aligned on the basis of verses and split up randomly in development, evaluation, and training parts. Bollmann et al. (2011) describe the alignment process in more detail; the Luther corpus used in this thesis is identical to the one described there except for one change: 1:n alignments, such as *soltu* → *sollst du* 'should you', have been removed.

One reason for removing 1:n alignments is that they raise the complexity of the normalization process; it is not trivial, for example, to decide how such normalizations could be generated with a simple Levenshtein algorithm (cf. Sec. 3.1.3). If the space character between the words is treated just like any other character, the number of possible normalization candidates increases by a large margin, which is likely to increase the error rate significantly. Considering that 1:n alignments only make up a tiny fraction of the corpus (e.g., 0.65% of the evaluation part), the disadvantages likely outweigh the benefits. The more important reason, however, is that 1:n alignments do not occur in the Anselm corpus, which is one of the primary application scenarios considered in this thesis. Here, in cases like *soltu*, the modern word boundaries are already marked in the transcription (e.g., *solt|u*). Therefore, these wordforms can be split up before the normalization process, avoiding the above-mentioned problems. The GerManC corpus features a similar tokenization scheme (Scheible et al., 2011b).

With this modification, the Luther corpus used here consists of 218,504 tokens in the training part and 109,258 tokens in the evaluation part. This is comparatively large for a

[2] Available from `http://www.sermon-online.de/`.

corpus of annotated[3] historical data: the manually annotated GerManC-GS corpus, for example, only consists of 57,845 tokens divided across 24 different texts with possibly different writing styles, while the average length of a text in the Anselm corpus is around 8,000 tokens.

Despite being written in 1545, language and spelling in the Luther bible is already relatively close to modern German. In the evaluation part of the corpus, 65.13% of all tokens in the ENHG version are already equivalent to their aligned modern counterparts, setting a relatively high baseline for a normalization algorithm. This is not too surprising, though, considering that Luther's 1545 bible translation was highly influential for the subsequent development of New High German (Besch, 2000). Typical spelling differences to NHG are 'v' in place of (modern) 'u' (as in *vnd* → *und* 'and') and duplication of consonant letters, often 'f' (as in the preposition *auff* → *auf* 'in/on/at').

2.2 TIGER/Tüba corpus

The TIGER/Tüba corpus has been created as a test and training corpus for the evaluation of part-of-speech tagging in Section 4. As the name suggests, it results from the combination of the TIGER corpus (Brants et al., 2002) and version 6 of the Tüba-D/Z treebank (Telljohann et al., 2004).

Although most POS taggers already provide a language model for German, there are a few problems with using them in the context of this work. First, models for different taggers have often been trained on different tagsets, requiring an individual post-processing step to map them to the tagset used in the evaluation texts. If the model uses a smaller tagset, an unambiguous mapping might not even exist, which could lead to distorted results. Second, while many taggers provide a pre-compiled language model, they do not provide the data that was used to train it. Having access to the training data, however, enables us to perform more sophisticated experiments, such as testing the effect of removing all punctuation, a potential drawback that resembles the situation found in historical data. Finally, tagging performance across different taggers becomes more comparable when the training data is the same in each case.

A slight problem arises from the fact that the texts in TIGER and Tüba-D/Z mostly follow traditional German orthography before the 1996 orthography reform. The gold standard normalizations in the Anselm corpus do not consequently follow either the original or the reformed orthography, though (GerManC-GS appears to follow old orthography rules). The most significant change of the reform consists of replacing 'ß' with 'ss' after short vowels; however, vowel length cannot be determined from the wordform alone. Therefore, to reduce potential errors that could arise from incompatible spellings, the corpora were modified to replace all 'ß' spellings with 'ss' regardless of their context; the same has been done with all other input data for the POS tagger. This

[3] In this case, the annotation is the modernization of the 1545 Luther text.

results in a partially artificial, but more consistent spelling style, while not negatively impacting tagging performance.[4]

Both TIGER and Tüba-D/Z contain POS annotation in the style of the Stuttgart-Tübingen-TagSet (STTS) (Schiller et al., 1999). There are several minor differences, however, mostly concerning the ordering of morphological attributes and naming conventions of tags (e.g., pronominal adverbs are tagged as PROP in Tüba-D/Z, but PROAV in TIGER). Therefore, a pre-processing step was performed to map both corpora to a uniform tagset. This mapping is unambiguous and introduces only minor changes to the tags. The combined corpus still conforms to the STTS guidelines[5] and consists of 1,864,816 tokens in 106,288 sentences.

2.3 Anselm corpus

The Anselm corpus (Schultz-Balluff and Dipper, 2013) consists of different versions of the text "Interrogatio Sancti Anselmi de Passione Domini" ('Questions by Saint Anselm about the Lord's Passion'). It is created in the context of an ongoing, interdisciplinary research project which aims to provide a digital, annotated edition of these texts. This includes enriching the data with part-of-speech annotation. Using this data, the performance of normalization and POS tagging can be evaluated in the context of an actual research scenario.

In total, there are more than 50 German manuscripts and prints of the Anselm text, written in various German dialects between the 14th and 16th centuries. Versions can differ considerably in length, averaging about 8,000 tokens. The texts are religious in nature and written in a question–answer style, with Saint Anselm of Canterbury asking questions to the Virgin Mary about the Passion of Jesus Christ. A gold standard annotation has been created for two of the manuscripts so far, which are consequently used here: a manuscript of 4,783 tokens kept in Melk, Austria, written in an Eastern Upper German dialect; and another manuscript of 5,399 tokens from Berlin, written in an Eastern Central German dialect. Both manuscripts are dated to the 15th century.

(1) *Owe allerlibeſte vrouwe dyn lybes kynt unſer meiſter iſt gevangen*
Oh weh allerliebste frau dein liebes kind unser meister ist gefangen
'Alas, dearest woman, your dear child, our master, has been captured'

(2) *O allerliebſte fraw · dein chind vnſer maiſter iſt geuan̄gn̄*
Oh allerliebste frau dein kind unser meister ist gefangen
'Oh dearest woman, your child, our master, has been captured'

Example 1 shows an excerpt from the Berlin manuscript, while Example 2 shows the same passage from the Melk text; the second line always gives the modern spelling of

[4]This is not surprising, considering that 'ß' and 'ss' are different spellings for the same consonant and rarely induce a difference in meaning. Nevertheless, ten-fold cross-validation has been performed on the TIGER/Tüba data with and without the replacement of 'ß', and found no difference in average POS tagging accuracy.

[5]A minor deviation is that prepositions do not contain annotations of morphological case, as this information was available in the TIGER corpus, but not in Tüba-D/Z.

each wordform (ignoring capitalization). Some characteristic spelling differences can already be observed here, e.g., the frequent use of the letter ‘y’ in the Berlin text, which can represent modern ‘i’, ‘ie’, or ‘ei’. The Melk manuscript uses the letter ‘y’ much less, but has the frequent spelling ‘ch’ for modern ‘k’, which does not appear in the Berlin text. Also note the very different spellings for modern *Frau* ‘woman’, *vrouwe* vs. *fraw*, which are both used frequently in the respective texts to refer to the Virgin Mary. These examples already highlight some of the difficulties of spelling normalization, as the individual spelling characteristics can vary substantially between texts.

2.3.1 Preprocessing

Transcriptions of the manuscripts are diplomatic, i.e., they seek to preserve as many features of the original manuscript as possible. As transcriptions are done in plain text, special characters or character sequences are used to encode such features. Very common encodings are, e.g., a dollar sign (\$) for a ‘long s’ (ſ) and a backslash followed by a hyphen (\-) for a nasal bar (̄). It is not trivial to process this data with normalization algorithms: in a naive approach, multi-character transcription symbols would be treated as separate “letters” by the algorithm. Converting the transcriptions to Unicode characters is a possible solution here. However, another problem remains: combining Anselm data with data from other sources that use different transcription conventions might prove difficult. If, for example, the ‘long s’ is not transcribed separately (and the plain ‘s’ character is used instead), normalization algorithms trained on Anselm texts (which always encode ‘long s’) will likely perform worse on this type of data, and vice versa. As the automatic processing of historical texts typically suffers from a sparse data problem, it is desirable to be able to include data from as many similar resources as possible, though.

To avoid the above-mentioned issues, all transcriptions are converted to plain alphabetic characters before further processing. ‘Long s’ is converted to plain ‘s’ during this step. Superscriptions and ligatures are treated as separate characters, while diacritics are removed except for the umlaut characters (ä, ö, ü) that still exist in modern German. Most of these conversions are straightforward; some transcription marks do not have a unique replacement, though. The nasal bar and the ‘r’ abbreviation (transcribed as an apostrophe) are typical examples, and occur relatively frequently in the texts. The former can be used in place of ‘(e)n’, ‘(e)m’, or ‘e’ preceding a nasal, while the latter typically replaces ‘(e)r’. Several of these variants can be observed in the Melk manuscript:

(3) (a) *meȳ* → *mein* (nasal bar for ‘n’)
(b) *iudn̄* → *juden* (nasal bar for ‘e’ before ‘n’)
(c) *vm̄b* → *um* (nasal bar has no modern representation)

(4) (a) *v’chaufft* → *verkauft* (abbreviation for ‘er’)
(b) *ma’ia* → *maria* (abbreviation for ‘r’)

Variants (3b) and (4a) occur the most often in this manuscript. Also, it can be observed that the nasal bar usually represents ‘e’ only when placed above ‘n’, and often has no representation in modern spelling when placed above ‘m’. Therefore, for the Melk manuscript, the following simplification rules are used: ‘n̄’ is simplified to ‘en’; ‘m̄’ is simplified to ‘m’; all other uses of the nasal bar introduce an extra letter ‘n’ after the letter they are placed on; and the ‘r’ abbreviation is always simplified to ‘er’. In the Berlin manuscript, the nasal bar is simplified to ‘n’ in all cases, as this appears to be the predominant use here. Also, both symbols are used much more frequently in the Melk manuscript (403 nasal bars, 39 ‘r’ abbreviations) than in the Berlin text (30 nasal bars, 2 ‘r’ abbreviations). While these simplification rules still introduce some errors, they appear to be the most sensible rules for these manuscripts without making individual decisions for each wordform.

Additionally, the letter ‘ß’ is always converted to ‘ss’. This only affects normalization and modernization, as ‘ß’ in its modern form does not occur in the transcriptions. The former, however, were found to be inconsistent regarding their use of the reformed German orthography. As the letter ‘ß’ is always replaced by ‘ss’ in the input data for the POS tagger anyway (cf. Sec. 2.2), the change was already made in the normalization and modernization layer here.

Finally, all characters in the transcriptions are converted to lowercase. The Anselm manuscripts lack consistent capitalization: in the Melk text, capital letters are often—but not always—used with proper nouns, but also in some contexts without a clear pattern, as seen in Example (5). They are not regularly used for common nouns. Also, while words at the beginning of a sentence are sometimes capitalized, this is not a definite rule.

(5) *von* *Rechter* *chrankchait*
von rechter Krankheit
‘(because) of grave disease’

These factors make it appear unlikely that capitalization provides a reliable clue for either the normalization or the tagging process. Therefore, this information was dropped by always lowercasing all characters.

Whenever examples from the texts are shown during discussion of normalization and POS tagging, only the lowercased, simplified version of the transcription will be given.

2.3.2 Annotation

Both the Melk and the Berlin manuscript have been manually normalized and POS tagged by student assistants. POS tagging follows the STTS tagset (Schiller et al., 1999) but does not include morphological information (such as case, number, gender). The normalization follows the guidelines published in Bollmann et al. (2012), which distinguish two layers: “strict” normalization and modernization. Normalization in the stricter sense

	$.	$,	other
virgule (/)	126	17	24
dot (.)	27	3	10
middle dot (·)	11	2	7
other	203	258	4,089

(a) Melk text

	$.	$,	other
alinea (¶)	177	21	20
middle dot (·)	98	200	158
virgule (/)	1	1	0
other	81	166	4,474

(b) Berlin text

Table 1: Correlation between original and modern punctuation marks in the Anselm texts; column labels refer to STTS tags, i.e., '$.' is sentence-final punctuation (including colons and semi-colons), '$,' is the modern comma.

represents the closest modern wordform for a given historical wordform, while modernization changes wordforms with regard to semantics and correct modern inflection. The latter also replaces extinct wordforms with modern equivalents, while the normalization layers maps them to "artificial" lemmas. The hypothesis here is that the normalization layer is easier to generate with automatic normalization algorithms (especially if they do not take word context into account to help determine proper inflection), while the modernization layer achieves a higher tagging accuracy due to being closer to a grammatical NHG sentence.

Additionally, both texts were enriched with information about modern punctuation and sentence boundaries. Table 1 gives an overview of the correlation between punctuation marks found in the manuscripts and the modern punctuation marks added by the annotator. In the Melk text, the original punctuation most often coincides with modern sentence-final punctuation, with the virgule being used the most. However, 55.31% (203) of all modern sentence boundaries are not marked in the original manuscript at all. In place of the modern comma, no punctuation mark appears in the manuscript in 92.14% (258) of the cases. The virgule symbol appears very frequently in this transcription, however only in the first 72% of the text—in the remaining part, not a single virgule symbol is found. It could not be determined whether this is an original feature of the manuscript or due to an unfinished transcription process. If it is the latter, modern sentence-final punctuation could potentially show a higher coincidence with the virgule symbol on the finished transcription.

In the Berlin text (Table 1b), the virgule is almost never used. Instead, a certain type of paragraph mark (¶), the alinea, is frequently used along with the middle dot (·). Here, the tendency is a bit clearer for modern punctuation marks to coincide with a symbol in the manuscript: modern sentence boundaries are most often marked with an alinea (49.58%), while the modern comma is most often represented by a middle dot (51.55%). On the other hand, considering Table 1b by row, the middle dot only

No.	Date	File name	Tokens
1	1677	LeichSermon	2,585
2	1730	JubelFeste	2,523
3	1770	Gottesdienst	2,292

Table 2: GerManC-GS texts used for evaluation; information taken from Scheible et al. (2011a).

GerManC-GS	STTS
NA	NN
PAVREL	PAV
PTKREL	PRELS
PWAVREL	PWAV
PWREL	PRELS

Table 3: Mappings for tags unique to the GerManC-GS corpus

corresponds to a modern comma 43.86% of the time, while having no modern equivalent in 34.65% of cases (158 occurrences; column 'other'). While this distribution is notably different from that of the Melk text, it is still very unreliable for creating a mapping of original to modern punctuation.

2.4 GerManC-GS corpus

The GerManC corpus aims to be a representative corpus of historical, written German from 1650–1800 (Scheible et al., 2011a), classified as Early Modern German (EMG) by the authors. It contains texts from different periods within that time frame, different dialectal regions, and different genres (such as "newspaper", "letter", or "narrative"). The GerManC-GS corpus aims to be a subcorpus of GerManC that has been enriched with gold standard annotations of normalization, lemmatization, and part-of-speech tags. For this purpose, three texts of different time periods have been selected for each genre. The subcorpus is created specifically to test and to help improve existing NLP tools on historical data (Scheible et al., 2011a, p. 125).

In this thesis, the texts of the genre "sermon" from the GerManC-GS corpus will be used (referred to as "Sermon texts" from now on).[6] This genre was chosen because it is of a religious nature, similar to the Anselm texts and the Luther bible. Table 2 gives some general information about the three texts. On average, they are less than 2,500 tokens in size, making them the shortest texts in this evaluation. Also, with the oldest text being written in 1677, they are considerably newer than both the Anselm texts (which are dated to the 15th century) and the Luther bible (from 1545). Therefore, the language of these texts is expected to be considerably closer to New High German, which in turn should make them easier to process with existing tools.

Examples (6–8) show an excerpt from each of the three Sermon texts. They give the impression that spelling is not as varied as with the Anselm texts, however, some

[6] I would like to thank Silke Scheible for kindly providing me with the texts.

inflectional (*kǒstlich* → *köstliches* 'delicious') and syntactic differences can still be found (*dir danken* would actually be *dir zu danken* 'to thank you' in NHG).

(6) *Und darumb ist Ihnen ihr Schmertz desto grǒsser worden*
Und darum ist Ihnen ihr Schmerz desto größer geworden
'And that is why their pain became so much greater'

(7) *Mein GOTT ! es ist ein kǒstlich Ding , Dir dancken*
Mein Gott ! Es ist ein köstliches Ding , dir danken
'My Lord! It is an exquisite thing to thank you'

(8) *gieb meinen Worten das Feuer , das die Herzen entzůndet*
gib meinen Worten das Feuer , das die Herzen entzündet
'Give my words the fire to ignite hearts'

The gold standard part-of-speech annotation follows the STTS tagset without morphological attributes. However, some additional tags have been introduced to cover features specific to Early Modern German: NA marks adjectives that are used as nouns, while the other tags (PAVREL, PTKREL, PWAVREL, PWREL) mark various parts of speech when used to introduce a relative clause. As the tagging evaluation is performed using a tagger trained on standard STTS, these tags are mapped back according to the scheme in Table 3. This mapping only affects 80 tokens from all three Sermon texts combined.

Contrary to the Anselm corpus, the original transcriptions are given in the form of Unicode characters. Superscripts of characters are commonly found, though in general, fewer "special" characters are used than in the Anselm data. Still, to be able to perform comparable evaluations with both types of data, the same conversions are applied to the Sermon texts (cf. Sec. 2.3.1): all texts are lowercased and converted to plain alphabetic characters. In analogy to the Anselm texts, examples discussed in the remainder of this thesis will use this simplified version of the texts only.

The normalization of spelling variants in GerManC-GS aims to achieve a compromise between historical accuracy and usefulness for modern NLP tools, and is described as "a type of pre-lemmatisation" (Scheible et al., 2011a, p. 126). Spelling variants are normalized to a common head variant, typically representing the modern spelling of the word; however, extinct wordforms are also normalized by applying systematic spelling modernizations (Scheible et al. (2011a) give the example of the verb ending *-iren* → *-ieren*). All in all, this normalization scheme is closer to the "normalization" layer of the Anselm corpus than the "modernization" one, though not exactly identical. The lemmatization layer is not used in this thesis.

3 Normalization

The task of normalization as it is performed here is defined as the mapping of historical wordforms to their modern counterparts. Section 3.1 first presents various normalization methods that can be used to achieve this, while Section 3.2 evaluates them on the corpora discussed in the previous section. Finally, Section 3.3 presents an approach to combine normalization algorithms and evaluates such combinations for the previously discussed methods.

3.1 Methods

This section describes the normalization methods, sometimes referred to as "normalizers", that are used for the evaluation. Sec. 3.1.1 presents a simple wordlist mapper; Sec. 3.1.2 presents a rule-based approach to normalization; Sec. 3.1.3 discusses how Levenshtein distance (and other distance measures) can be used for normalization; and Sec. 3.1.4 describes the automatic learning of a weighted Levenshtein distance.

All normalization methods presented here have been previously discussed in Bollmann (2012), except for the learning algorithm presented in Sec. 3.1.4. In the context of this thesis, a stronger focus will be put on the individual strengths and weaknesses of the algorithms, especially in the light of the individual characteristics of the Anselm and GerManC texts.

There is one characteristic that all of the presented methods have in common: they operate context-free, i.e., they only process one wordform at a time without taking neighboring wordforms into account. Naturally, this is a disadvantage when dealing with ambiguous wordforms that should be normalized differently depending on context. A common example from Early New High German is the form *jn*, which can represent both the preposition *in* 'in' and the pronoun *ihn* 'him'. All methods discussed below will (ideally) always choose either one or the other, but cannot discriminate between these cases, which limits the maximum accuracy they can achieve. As an example, this upper bound is found to be between 90% and 95% for the Anselm texts (cf. Table 4 on p. 25).

A method to make use of context information to improve normalization is presented by Jurish (2010), which also includes a combination of different normalization algorithms. The reasons for not considering such an approach here mainly lie in the raised complexity of such a system: as few comparisons between methods exist, it is desirable to evaluate the strengths, weaknesses, and typical accuracies of the individual normalization methods first before including them in a more complex framework. As some of the discussed algorithms have not yet been thoroughly evaluated on their own, evaluation is restricted to the simpler context-free approach here.

3.1.1 Wordlist mapping

From a conceptual point of view, arguably the simplest method of normalization is to perform word-by-word substitutions according to a pre-defined list. This method is called "wordlist mapping" here, as it maps entries from a list of historical wordforms to their modern counterparts. As a result, it has no concept of characters or spelling variation.

Mappings can be created manually or learned from training data. In the implementation used here, the wordlist mapper trivially learns pairs of historical and modern wordforms from aligned data, and also keeps track of how often each pair was seen during training. During normalization, if a historical wordform can be mapped to more than one modern wordform, the mapping with the highest frequency is chosen.

Despite its simplicity, the wordlist mapping approach is actually a very effective strategy for automatic normalization, as the evaluation in Section 3.2 confirms. A wordlist that has been trained on a sufficiently varied corpus of historical data can possibly contain many different spelling variants, particularly of function words. Therefore, even if the vocabulary of the text to be normalized is quite different from the one the wordlist has been trained on, if many common function words are covered by the list, this will still result in many correct normalizations. The effect of correctly normalizing function words should not be underestimated; they are not only very frequent in any kind of text, but also among the hardest words to normalize for other algorithms. This is because they tend to be short, and therefore more easily confusable with other words:

(9) *eyme* → *einem* 'a/an' (dative case)

Example (9) is taken from the Berlin text. The wordform *eyme* here has only two letters in common with its desired normalization *einem* (or three, if transposition of letters is allowed), which in this case means that more than half of the input word must be changed in order to match the modern form. It is not obvious how a context-free algorithm that operates on a character level would arrive at the correct word, considering that there are several other candidate wordforms which are more "similar" to the input string, such as *eimer* 'bucket' or *reime* 'rhymes' (see also Secs. 3.1.3 and 3.1.4 for clearly defined notions of "similarity").

This example also highlights another advantage of using wordlists: it can cover spellings that are highly idiosyncratic (provided that at least one instance of the spelling has been learned during training) and/or differ considerably from modern spelling conventions. Also, it is well-suited for handling abbreviations, which are very common in historical manuscripts and cannot be sensibly processed on a character level alone, such as in Example (10).

(10) *ihūs xpūs* → *jesus christus* 'Jesus Christ'

Due to the very nature of the method, a wordform that has not been previously seen cannot be normalized at all using wordlists. This is a drawback which implicates that wordlist mapping should probably not be used alone, but rather as one component of a combination of normalization methods. This approach is explored further in Section 3.3.

3.1.2 Rule-based normalization

The rule-based normalization algorithm was first presented in Bollmann et al. (2011); the identical algorithm will be used here. The main idea of this approach is to extract character rewrite rules from a training set of aligned data. Rewrite rules can operate on one or more characters and take their immediate context into account. They can be written in a form similar to phonological rules:

(11) $v \rightarrow u$ / # _ n ('v' is replaced by 'u' between a word boundary and 'n')

(12) $j \rightarrow ih$ / # _ r ('j' is replaced by 'ih' between a word boundary and 'r')

(13) $we \rightarrow \varepsilon$ / u _ # ('we' is deleted between 'u' and a word boundary)

Example (11) shows a rule typical for the Luther bible, where it is learned from the very frequent word pair *vnd* → *und* 'and'. Rule contexts always consist of exactly one character or a word boundary (#), whereas both the left- and right-hand side may contain multiple characters as well as none (Examples (12) and (13)), the latter case being symbolized by the epsilon symbol (ε). This feature of the rule-based method resulted from an ambiguity inherent in the learning algorithm for the rules. It is based on the algorithm for calculating Levenshtein distance (Levenshtein, 1966), where instead of only counting the number of edit operations that are performed, the actual operations themselves are recorded. However, this can lead to ambiguous alignments. Example (14) shows the two possible alignments for the word pair *jre* → *ihre* 'her/their':

(14) (a)
j		*r*	*e*
i	*h*	*r*	*e*

(b)
	j	*r*	*e*
i	*h*	*r*	*e*

Here, both alignments can be generated by the Levenshtein algorithm, as the number of edit operations is the same for both (14a) and (14b). However, from a linguistic point of view, the replacement $j \rightarrow i$ is more plausible than the alternative $j \rightarrow h$. Also, during the development of this method, many cases like (14) involved digraphs ('ih') or trigraphs. Therefore, to circumvent the alignment problem, neighboring replacement rules are always merged, in this case resulting in rule (12). In Section 3.1.4, an alternative approach to this problem is discussed.

For normalization, rules are applied to a historical input word from left to right. If there are several applicable rules for a given position within a word, rules are chosen in such a way that the average probability of rules used to generate a normalization

is maximized. Rule probabilities are calculated from the frequencies of the rules during training. However, they are not the only factor in deciding which rule to apply: additionally, a (modern) lexicon lookup is performed. Only words that are covered by this lexicon can ever be generated by the rules. In the implementation used here, this dictionary is dynamically updated during training if the training data contain previously unknown modern wordforms.

(15) (a) $d \rightarrow d \,/\, n \_ \#$ ('d' is not changed between 'n' and a word boundary)
(b) $\varepsilon \rightarrow \varepsilon \,/\, r \_ e$ (nothing is inserted between 'r' and 'e')

A peculiarity of the rule-based approach is the introduction of identity rules and epsilon identity rules, as shown in Example (15). Identity rules state that a character should not be changed in a given context, while epsilon identities are taken to mean that no insertion has taken place. These type of rules are intended to "compete" with other, non-identity rules during the actual normalization process. As rules are selected based on their probabilities, if an identity rule has a sufficiently higher probability, it will be chosen over other, non-identity rules. This is the only mechanism in this approach for deciding whether a given character in the input word should even be modified or not. Again, Section 3.1.4 presents an alternative way to handle this problem.

Note that the rule-based normalization can fail to produce a result in either of two cases: (1) there is no applicable rule for a given position in a word; or (2) no combination of rules leads to a word covered by the modern lexicon. The first case can occur if there is very little training data, or the data to be normalized contains spellings that are very different from those seen during training. The second case can occur for the same reasons, but also when the vocabulary of the test data is not covered by the modern lexicon. Both cases result in leaving the historical wordform unchanged.

3.1.3 Levenshtein distance

Levenshtein distance is a string-based distance measure originally described in Levenshtein (1966). The Levenshtein distance—also called edit distance—between two strings is defined as the minimum number of edit operations required to transform one string into the other. Allowed edit operations are the insertion, deletion, and substitution of characters.

In the context of normalization, Levenshtein distance can be used by comparing a historical input string to all wordforms found in a modern lexicon. The wordform with the lowest distance to the input string is then used as the normalization. In theory, this approach is not restricted to Levenshtein distance, but can be used with any string-based distance measure.

The normalization algorithm based on Levenshtein distance is the only normalization method compared here which requires no training data, and in fact cannot be trained at all. For this approach to produce meaningful results, it is necessary that the spelling in

the historical text is already very close to modern spelling. However, even then, wrong normalizations are very likely to occur. This is because every character is treated the same way: for the wordform *got*, the algorithm is just as likely to return the correct *gott* 'God' as it is to return *rot* 'red' or *gut* 'good', because they all have an edit distance of 1 to the input string.

Therefore, standard Levenshtein distance should probably not be seen as a serious normalization method, but rather as a baseline for other distance-based measures. The evaluation in Section 3.2 supports this view.

3.1.4 Weighted Levenshtein distance

While standard Levenshtein distance is a very raw measure for comparing strings, there exist several refinements for different application scenarios. One of the most versatile extensions is to allow edit operations to have individual weights. In this version, instead of simply counting the number of edit operations, the sum of the weights of the individual operations is used. From all possible sets of edit operations that transform one string into the other, the sums of the weights are considered; the minimum of these sums is then defined as the distance between the two strings. This variant of Levenshtein distance will be called "weighted Levenshtein distance" (WLD) here. If all weights are set to 1, WLD is equivalent to the standard Levenshtein distance discussed above. The method of using this distance for normalization is the same as for standard Levenshtein distance, too.

Properties Weights are typically set between 0 and 1, with 1 being the default cost for insertions, deletions, and substitutions if no individual weight is defined. Weights allow for a more refined assessment of spelling differences: the substitution $j \rightarrow i$ can be given a relatively low weight, as this is a common spelling variation in some ENHG texts, whereas a substitution that should never occur (such as $j \rightarrow x$) can be left at the default cost. This way, the first substitution will always be preferred, while the latter will only be considered in rare circumstances, if at all.

Note that even with weighted Levenshtein distance, different variants are conceivable: for example, WLD can be either directed or undirected. In the undirected version, substituting 'a' with 'b' always has the same weight as substituting 'b' with 'a'; in the directed version, they can be assigned different weights. Spelling variations between historical and modern texts are typically not symmetric, though. While the letter 'u' can sometimes represent a modern 'f'—Example (16) shows this and is taken from the Luther bible—it is unusual to find an example for the opposite direction. Even with characters that are used interchangeably to some extent, like 'u' and 'v', one direction is usually more common than the other: while $u \rightarrow v$ is possible, as in Example (17), it is much more common to find the replacement $v \rightarrow u$. These observations should be

reflected in the edit weights. Hence, it seems reasonable to always use directed WLD when working with historical texts.

(16) *zweiuel* → *zweifel* 'doubt'

(17) *douon* → *davon* 'therefrom'

A distinguishing property of WLD is that only actual modifications to the input string can be assigned weights. In other words, leaving a character unchanged always has a "cost" of zero. This is a fundamental difference to the approach of the rule-based method (cf. Sec. 3.1.2), where identity and non-identity rules have equal status, and the preference for any given character to be left unchanged must be learned from the training data first. Therefore, normalization with WLD always has a bias towards similar wordforms, which is not necessarily the case with the rule-based approach. In fact, if an input wordform is already covered by the modern lexicon, it will never be changed by the WLD approach (as the distance of a wordform to itself is always zero), whereas this is still possible with the rule-based method.

Levenshtein distance and its modifications typically operate on single characters. It is possible, though, to extend the measure to *n*-gram substitutions. As an example, for the word pair in Example (17), a weight could be defined for the edit operation *ou* → *av*. Note that this weight can be independent of that of the two single-character edit operations combined. Furthermore, it is not necessary that both sides of the edit operation have the same number of characters. The resulting distance measure is still well-defined using the same definition as given above; allowing *n*-gram weights simply extends the number of possible edit operations that have to be considered.

With this modification, WLD does not only cover both common single-letter variants and multi-character substitutions, but also offers more flexibility than the rule-based approach. This is because *n*-gram weights can also be defined for substitutions where both sides share one or more characters, e.g., *vn* → *un*. This resembles the context information found in the rule-based approach, but has the advantage that the amount of context that is included can be freely varied. Therefore, WLD with *n*-grams can assign weights to very specific replacements as well as to more general ones. The rule-based method, on the other hand, is not able to capture generalizations (such as *v* → *u*) unless the replacement is learned in all of its possible contexts. There is one drawback of the current WLD implementation, though: it is not possible to refer to word boundaries in *n*-gram substitutions. While it is conceivable to extend the approach to allow for this (by treating the word boundary as just another character), this feature has not yet been implemented here.

To sum up, weighted Levenshtein distance (WLD) as it is used here extends Levenshtein distance by assigning weights to edit operations. The default cost of single-character edit operations is 1, while the identity of characters (i.e., not changing anything) always has a weight of 0. Weights may be also assigned to *n*-gram edit operations.

Furthermore, a directed WLD measure is used, so rule weights do not need to be symmetric.

Implementation and learning of weights In Bollmann (2012), first experiments were performed with a WLD measure where weights had been defined manually. For this, the historical data had been manually inspected for characteristic spelling variations first. A more elegant solution is to learn weights automatically from training data, as this approach can profit from already existing gold standard normalizations without requiring further manual work. Several algorithms have been proposed for this purpose (e.g., Ristad and Yianilos, 1998; Hauser and Schulz, 2007); the implementation used here roughly follows the approach outlined in Adesam et al. (2012).

Before any weights can be learned, the word pairs used for training have to be aligned on a character level. This can be achieved using a Levenshtein algorithm, but requires the resolution of ambiguity, which was already mentioned in the discussion of the rule-based approach (cf. p. 18) and illustrated in Example (14). Where the rule-based method circumvents the problem by merging neighboring rules, another approach is taken for the WLD method. Here, the ambiguity is resolved by using iterated Levenshtein distance alignment (Wieling et al., 2009). The main principle underlying this alignment algorithm is to calculate statistical dependence between characters. In the example *jre* → *ihre*, two possible alignments for the character 'j' exist; which alignment is chosen ultimately depends on the other contexts where 'j' occurs. If there are other word pairs where 'j' can be unambiguously aligned to 'i' (e.g., *jn* → *in*), but none that unambiguously connect 'j' to 'h', the statistical dependence between 'j' and 'i' will be considerably higher. This leads to the alignment *j* → *i* being preferred in ambiguous cases, too. Statistical dependence is calculated using pointwise mutual information (Church and Hanks, 1990); edit distance weights are then adjusted accordingly, and alignments are re-calculated using these new weights. The whole process is repeatedly iteratively until the alignments have converged.

The formula to calculate WLD weights from the final set of alignments is again based on Adesam et al. (2012), but was refined empirically by testing on the development part of the Luther corpus (cf. Sec. 2.1) to produce more sensible results. If LHS and RHS are the left- and right-hand sides of a substitution, respectively, the basic formula is:

$$(18)\quad -\frac{1}{d}\log p_\alpha(\text{RHS}|\text{LHS})$$

The function p_α calculates the (conditional) probability of the characters with additive smoothing (using $\alpha = 0.5$), with the exception that infrequent characters on the left-hand side are additionally penalized. The factor d is used to scale down the resulting value in order to bring it in line with the default cost of 1. Adesam et al. (2012) use $d = 10$; with the Luther bible, $d = 7$ was found to be better. Finally, weights that are

higher than the number of characters on the left-hand side were cut, as they will rarely if ever be used during normalization. Primarily, this was a consideration of runtime performance of the algorithm (see below).

With regard to *n*-gram weights, two seperate variants are considered: firstly, using only unigram weights (WLD_{uni}); and secondly, extracting unigram, bigram, and trigram weights (WLD_{tri}). For determining bigrams and trigrams, "empty" alignments are also counted. Consider again the alignment of *jre* → *ihre* in Example (14a), repeated here including the epsilon symbol for the empty position:

(19)
j	*ε*	*r*	*e*
i	*h*	*r*	*e*

The modern character 'h' has no alignment in the historical input string. However, the empty position in the input string is still counted for the extraction of *n*-grams. As an example, for bigram extraction, this results in the following alignments to be considered:

(20) (a)
[*j*	*ε*]	*r*	*e*
[*i*	*h*]	*r*	*e*

(b)
j	[*ε*	*r*]	*e*
i	[*h*	*r*]	*e*

(c)
j	*ε*	[*r*	*e*]
i	*h*	[*r*	*e*]

From the alignments in Example (20), weights will be learned for the substitutions *j* → *ih* and *r* → *hr*. The alignment in (20c) has identical characters on the left- and right-hand side, and is therefore not assigned a weight, as identities are required to always have a weight of zero. This example shows that counting empty positions can result in edit operations with differing numbers of characters on both sides, similar to the multi-character rules from Examples (12) and (13).

One exception is made when extracting weights: "pure" insertions, i.e., edit operations with an empty left-hand side, are not allowed. This is a drawback especially for the unigram-only variant, as no insertion weights can be learned at all. Here, the insertion of a character always has the default cost of 1. This restriction was found to significantly improve normalization results, as insertions tend to be applied too often otherwise. Note that insertions can still occur with bigrams and trigrams, as in Example (20), if the left-hand side of the edit operation has fewer characters than the right-hand side.

Finally, a few things should be said about the implementation of the normalization algorithm. Normalization using WLD is defined as finding the entry in a modern lexicon with the lowest distance to the input string. Of course, calculating the distance for all entries in the lexicon is computationally expensive and therefore not feasible. The implementation used here is built using finite-state technology. The main idea is that any WLD parametrization can be represented as a weighted finite-state transducer (Mohri and Sproat, 1996). If this transducer is composed with a finite-state acceptor representing the modern lexicon, this results in a finite-state automaton that takes an input string and

outputs all words in the modern lexicon along with their distance. Applying a shortest-path algorithm to this automaton then yields the entry with the lowest distance (Mohri et al., 2000, p. 15).

While this method is exact, it is also relatively slow. The time to normalize a single token varies a lot and also depends on the amount of different weights that have been defined/learned. However, it was not uncommon to see durations between 500 ms and 3,000 ms on the test setup[7] for normalization of just one token, which is probably too slow for productive use. Using heuristics to reduce the number of calculations is probably the most promising solution; Adesam et al. (2012) use a filtering approach for this purpose. While there has been theoretical work on more efficient algorithms for generalizations of Levenshtein distance (Mitankin et al., 2011), no efficient implementation for WLD exists based on this theoretical model.[8]

3.2 Evaluation

This section presents an evaluation of the normalization methods discussed above. First, some general information about the evaluation procedure and the texts used for the evaluation is given. Section 3.2.1 then proceeds to give a quantitative evaluation of the different methods across the texts, while Section 3.2.2 examines the individual strengths and weaknesses of the methods by analyzing examples and looking at what was actually learned. Section 3.2.3 tries to assess the question of how much training data is required for good normalization results.

All normalization methods except for the wordlist mapper require a modern lexicon to work. The same modern lexicon is used for all evaluations here. It consists of all simplices that can be generated using the finite-state morphology DMOR (Schiller, 1996) and is additionally enriched with the vocabulary of the full modernized Luther bible.

For all texts except Luther, normalization is evaluated without punctuation and foreign words (e.g., passages or phrases written in Latin). These tokens are filtered out using the gold standard part-of-speech annotation to detect them. While the Luther test corpus does not contain punctuation marks, foreign words have not been removed as no POS annotation for the corpus is available. The filtering is done to obtain more accurate results: punctuation marks should not be changed by the normalization and are therefore trivial to normalize, and the spelling of foreign words is unlikely to be relevant for the spelling of words in the main language. At least for the Anselm corpus, POS annotation is not required to perform this filtering step: punctuation marks can be detected relatively easily from the characters used, and foreign words are already explicitly marked in the transcriptions. Therefore, the filtering can also be applied to

[7] Intel i7-870 @ 2.93 GHz, 16 GB RAM, Linux 3.6.4

[8] In fact, the main author of the cited paper doubts that such an implementation is feasible for weighted Levenshtein distance due to the high memory requirements (P. Mitankin, personal communication, October 8, 2012).

Text	Tokens	Baseline	Maximum
Luther (eval)	109,258	65.13%	94.65%
Berlin (norm)	4,700	23.40%	95.23%
Berlin (mod)	4,700	21.34%	92.04%
Melk (norm)	4,541	39.46%	93.26%
Melk (mod)	4,541	35.87%	90.42%
LeichSermon	2,178	73.09%	98.99%
JubelFeste	2,137	79.41%	99.53%
Gottesdienst	1,953	83.92%	99.74%

Table 4: Size, baseline (accuracy without normalization) and maximum achievable accuracy per text, after filtering punctuation marks and foreign words; "norm" and "mod" refer to the normalization and modernization layer of the Anselm corpus, respectively.

these texts without prior POS annotation.

Table 4 shows the size of all texts after filtering along with their baseline and maximum accuracies. The baseline is the percentage of identical tokens without any normalization step. It shows how similar the spelling of a historical text already is to its modernized (or manually normalized) version. Baseline scores differ vastly, but not surprisingly: the Anselm texts (Berlin and Melk) are the oldest ones in the evaluation and have the lowest baseline, the Berlin text going as low as 21.34%, showing that it is very far from modern German spelling. The Sermon texts, on the other hand, are the most recent ones and show baselines as high as 83.92%. Also, the baseline steadily increases from the oldest (LeichSermon, 1677) to the newest text (Gottesdienst, 1770).

Maximum accuracy is the highest accuracy that can be achieved using context-free normalization methods. As discussed at the beginning of Section 3.1, if a historical wordform is aligned with more than one modern wordform over the course of a text, one of these wordforms is inevitably normalized incorrectly, thereby limiting the maximum possible accuracy. Again, there is a tendency for older texts to have a lower score here: 90.42% is the maximum for the modernization layer of Melk, while the Gottesdienst text can, in theory, be normalized almost perfectly (99.74%). However, the high percentages of the Sermon texts may at least partly arise from their comparatively small size: the shorter a text is, the less likely conflicting normalizations are to occur. The Anselm texts show that maximum accuracy is not directly tied to the baseline score: while the Berlin text has the lowest baseline, its maximum accuracy is slightly higher than that of the Melk text. This may be an indicator that although Berlin is the most different from modern spelling, its spelling might be slightly more consistent than that of Melk.

	Baseline	Mapper	Rules	Leven	WLD$_{uni}$	WLD$_{tri}$
Luther (eval)	65.13%	**92.60%**	90.40%	80.19%	83.79%	87.72%
Berlin (norm)	23.05%	*62.05%*	**63.17%**	37.17%	51.14%	60.71%
Berlin (mod)	21.12%	*58.05%*	**58.90%**	34.71%	47.55%	56.14%
Melk (norm)	39.32%	63.15%	64.14%	54.10%	65.40%	**69.34%**
Melk (mod)	35.91%	57.81%	60.41%	50.16%	60.55%	**64.56%**
LeichSermon	72.71%	*76.46%*	**78.67%**	70.80%	75.63%	*76.40%*
JubelFeste	79.47%	85.22%	**88.52%**	84.06%	86.07%	*88.15%*
Gottesdienst	83.41%	86.58%	90.50%	87.89%	93.05%	**95.46%**

Table 5: Normalization accuracy per text after training on the first 500 tokens and evaluating on the rest, ignoring punctuation and foreign words (except for Luther: training and evaluation on separate parts); best result for each row is highlighted in bold, differences to the best result that are not statistically significant ($p > 0.05$) are marked in italics.

Finally, for the Anselm texts, the scores of the modernization layer are significantly lower than those of the normalization layer. This is only natural, as the normalization layer is defined as being closest to the original text. Because the modernization layer also adjusts inflection, it is conceivable that a historical wordform (or rather: a type) without inflectional endings is mapped to several modern wordforms (types) with different inflectional endings, depending on context. This would explain the differences in maximum accuracy between the layers.

3.2.1 Quantitative analysis

All normalization algorithms evaluated here, except for the Levenshtein method, require training data before they can be used for normalization. As the individual spelling characteristics of the texts are so different (cf. Secs. 2.3 and 2.4), it seems reasonable to first use in-domain training data only; i.e., all normalization algorithms are trained only on a portion of the same text they are evaluated on. For the Luther corpus, separate training and evaluation parts are used (see Sec. 2.1). For all other texts, the first 500 tokens are used for training, and the remainder of the texts for evaluation.

Table 5 presents the normalization results. It shows that for all texts except Luther, either the rule-based method (column "Rules") or the trigram variant of weighted Levenshtein distance ("WLD$_{tri}$") works best. The wordlist mapper always performs slightly (about 1–4 percentage points) worse than the rule-based method, except for the Luther bible, where it is the best normalization method. Also, its difference to the

best method is not statistically significant[9] for Berlin and LeichSermon. Levenshtein distance ("Leven") has the worst accuracy in all cases except for Gottesdienst, where it is better than wordlist mapping. Still, this confirms the view that it is not well-suited for the normalization task, but rather as a baseline for the weighted Levenshtein variants, which always perform better here. WLD that uses only unigrams ("WLD_{uni}") is always worse than the version including bigram and trigram weights, sometimes considerably so (e.g., 51.14% vs. 60.71% for the Berlin text).

For the Anselm texts, the normalization and modernization layers are again evaluated separately. The results are not surprising; accuracies for the modernization are consistently and almost evenly (about 3–5 percentage points) worse than for the normalization. Again, this only resembles the fact that the normalization layer is closer to the original wordform. The wordlist mapper is the only method which is insensitive to the extent of spelling difference, and could therefore be expected to perform better on modernization than other methods. This is not the case, which is likely due to the same reason as the difference in maximum accuracy (cf. p. 26): the adjustment of inflectional endings causes a type in the historical text to be aligned with several more types in the modernization layer, forcing a higher number of errors.

Among the texts used in this evaluation, the Luther corpus takes a special position. It is significantly larger than all the other texts, especially when considering the portions used for training: while only 500 tokens are used for the other texts, the Luther bible's training part consists of 218,504 tokens. As it is typically best to have as much training data as possible, the size of the training portion should have a noticeable effect on normalization accuracy. In fact, even though the Sermon texts all have a higher baseline than Luther, only the third one achieves a higher accuracy (94.49% vs. 92.60%). Still, considering the huge size difference, this is already a remarkable result.

The fact that the best result for Luther is achieved by the wordlist mapper—contrary to all Anselm and Sermon texts—could also be attributed to the large amount of training data. From the evaluation, this cannot be claimed with certainty, as even when increasing the size of the training parts for Anselm and Sermon (see Sec. 3.2.3), using the wordlist mapper never results in the best score. However, as the Luther training portion is even larger than all of the other texts combined, it cannot be ruled out that the wordlist mapper is actually the best overall method with large amounts of training data. The results can definitely be seen as an argument in favor of pooling resources of gold standard normalizations in order to create a larger and better training corpus.

Apart from the factor of size, the Luther corpus is also the only one which does not represent an actual research scenario, but was constructed specifically to evaluate normalization methods. Being a bible translation, it is also likely to be very carefully translated and printed, thereby showing more systematic spelling variations, but less

[9]For these and all further tests of statistical significance, chi-squared tests on the token counts (with a confidence level of 95%) have been used.

spelling inconsistencies. Manuscripts (such as Berlin and Melk), on the other hand, are much more prone to clerical errors. Additionally, spelling in manuscripts can also be influenced by external factors such as the size or layout of the page, while the layout of printed texts is usually planned out in advance. All of these factors might contribute to the comparatively higher accuracy scores reported for Luther.

Finally, the Luther corpus has another advantage that causes the results to be slightly biased: the complete vocabulary of its modern version has been added to the modern lexicon. Hence, each of the correct wordforms can theoretically be produced by the automatic normalization, while this is not necessarily the case for the other texts. Table 6 shows the ratio of "unknowns" for each text, i.e., tokens in the gold standard normalization that are not included in the modern lexicon. Unknowns make up between 1.81% and 5.19% of the other texts, with the exception of LeichSermon, which has a significantly higher percentage of unknown tokens (12.40%). Looking at the normalization of LeichSermon, the reason for this quickly becomes clear: it contains an unusually high amount of numerical tokens, typically referring to bible verses, as in Example (21).

Text	**Unknowns**	
Luther (eval)	0	0.00%
Berlin (norm)	174	3.70%
Berlin (mod)	130	2.77%
Melk (norm)	146	3.22%
Melk (mod)	82	1.81%
LeichSermon	270	12.40%
JubelFeste	111	5.19%
Gottesdienst	54	2.76%

Table 6: Unknowns per text (tokens in the gold standard normalization not covered by the modern lexicon)

(21) *psal. ix. 13. luc. 2. 52. sir. 39. 2. 3. 4. 5. tob. 10. 5. [. . .]*
psalmen 9 13. lukas 2. 52. sirach 39. 2. 3. 4. 5. tobit 10. 5. [. . .]
'Psalms 9,13, Luke 2,52, Sirach 39,2–5, Tobit 10,5, . . . '

Numerals are not explicitly handled; they do not occur at all in the Anselm texts or the Luther corpus. While there are numerals in the other Sermon texts, too, they do not appear as frequently as in LeichSermon, where they account for 165 of the 270 unknowns. As no exhaustive list of numerals is included in the modern lexicon, the Levenshtein algorithm will try to normalize them to other wordforms, leading to nonsense normalizations. This problem is specific to the distance-based algorithms, though: not only can the wordlist mapper and the rule-based method learn the proper mappings, but they will leave the original token unchanged if no appropriate mapping or rule was learned. Example (21) shows that for using numerals with Levenshtein-based algorithms, they either have to be explicitly added to the lexicon (which is impractical, considering that their number is theoretically unbounded) or handled by an exception rule in the algorithm.

The high ratio of unknowns in the LeichSermon text also explains the seemingly dubious result that Levenshtein distance is actually performing worse than the baseline

here (70.80% vs. 72.71%). As the baseline accuracy is equivalent to that of a "normalization" method which leaves all words unchanged, this means the Levenshtein algorithm introduces more unwanted changes than correct ones. As a Levenshtein-based algorithm will never change a wordform that is already in the modern lexicon, the only explanation for this result is that the normalization of LeichSermon contains many wordforms that are not covered by the lexicon, which is consistent with the data shown in Table 6.

From the evaluation results in Table 5, the scores for Luther have already been reported in Bollmann (2012), except for those of the WLD measure.[10] Instead, the study evaluated a similar version of WLD with the difference that weights had been defined manually rather than learned automatically. Weights were created after manual inspection of the first 500 tokens of the Luther training corpus. This procedure resulted in 88.33% accuracy, whereas the automatically trained parametrizations only achieve 87.72% at best (Table 5, Luther with WLD_{tri}). These results suggest that the learning process of the edit distance weights, e.g., the scaling and smoothing factors (cf. p. 22 ff.), can still be improved.

3.2.2 Qualitative analysis

The last section compared the performance of normalization methods by their overall accuracy. However, this is not the only factor in deciding how useful an automatic normalization is for further processing with NLP tools. Especially in those cases where the methods produce incorrect results, these wrong normalizations can range from complete mistakes to only having different inflection from the correct wordform. Furthermore, it is conceivable that the set of correctly normalized wordforms is also different between the algorithms. This section takes a closer look at the actual normalizations and the parametrizations learned by each method.

Note that when discussing examples, the Anselm texts do not contain the 'ß' character (cf. Sec. 2.3). The following examples will therefore sometimes use a slightly artificial modern spelling, using 'ss' in places where modern 'ß' would be correct. This is not a mistake, but due to the technical considerations explained in Sections 2.2 and 2.3.

Wordlist mapping The wordlist mapper is probably the simplest case to analyze. Due to its nature, it cannot produce partially correct normalizations, as it only operates on whole wordforms. Therefore, a correct normalization can only result from two scenarios: either the correct mapping for the input wordform has been learned and is correctly applied, or there is no known mapping for the input string, but the normalized form is equal to the historical one. The latter case certainly occurs more or less by chance and is not a "feature" of the normalizer itself, but also contributes to the final accuracy

[10] Note that while the Melk text had also been evaluated in Bollmann (2012), the results cannot directly be compared as the evaluation here is based on a newer version of the transcription.

Mapping	Count
jn → ihn	839
jn → in	38
jn → ihm	5
jn → indes	2
jn → es	1
auff → auf	1,746
auff → herauf	22
auff → an	1
ding → dinge	30
ding → ding	11

(a) Luther

Mapping	Count
yn → in	2
yn → ihn	2
yn → ihm	1
yn → ihnen	1
das → das	5
das → dass	5
komen → kommen	1
komen → gekommen	1
spricht → spricht	1
spricht → heisst	1

(b) Berlin (modern)

Table 7: Examples for historical types mapped to multiple modern types, considering only the training parts of the texts

score. Incorrect normalizations, on the other hand, can result when no mapping has been learned even though the input wordform should be changed, but also from applying learned mappings when not appropriate. This is typically the case with historical types that have multiple alignments in the normalization, as only the most frequent mapping will ever be applied.

Table 7 shows examples for such ambiguous mappings that have been learned by the wordlist mapper. It suggests that especially for short function words, multiple alignment types are likely to exist. This is the case with the token *jn* in Luther, for example, that is usually mapped to either the pronoun *ihn* 'him' (accusative case) or the preposition *in* 'in'. Other alignments, e.g., with the pronoun *ihm* 'him' (dative case), also occur, but are relatively rare. It is not impossible that they partly stem from mistakes during the alignment process of the two bible versions, which was not done by hand, but rather automatically (Bollmann et al., 2011). From this parametrization, the wordlist mapper will always normalize *jn* to *ihn*. If we assume that a similar distribution of *in* vs. *ihn* can be found in the evaluation part of Luther, which is half the size of the training part, then this should introduce about 20 incorrect normalizations in the evaluation. While this is certainly not much, if there are many cases of these ambiguous mappings, the mistakes can quickly add up. From the Luther training part, a total of 14,905 mappings with 12,242 unique source types are learned. This means that there is a total of 2,663 mappings which are learned, but never applied.

Typically, if a mapping is ambiguous, one of the modern normalization candidates

has a much higher frequency than the others. For example, *auff* has been mapped to the preposition *auf* 'at/on' 1,746 times, while the second most frequent mapping to the adverb *herauf* 'on' was seen only 22 times. In the modernization of the Berlin manuscript, though, the wordform *das* is mapped an equal number of times to both *das* and *dass*. While the former can be an article, a demonstrative pronoun, or a relative pronoun, the latter is always a subordinating conjunction. Hence, this is a difference which will have consequences for part-of-speech tagging. In cases like these, where there is a tie between two possible mappings, it is more or less up to chance which mapping the normalizer will apply: the mapping that has been learned first will be chosen. If these situations occur often or with wordforms of high frequency, many mistakes will be introduced this way. This is a strong argument in favor of using context information to aid the normalization process, an approach which will not be explored in the context of this thesis, though.

Different types of ambiguous mappings can be observed in Table 7, too. The different mappings for *jn* and *yn* usually stem from distinctions in spelling that are not yet made (or not made consistently) in Early New High German. On the other hand, the different mappings for *ding* result from inflectional differences. While *ding* 'thing' is a singular form in modern German, it is more often mapped to the plural form *dinge*. Example (22) shows an alignment from which this mapping would be learned; it is an excerpt from the Old Testament, 2 Kings, 8,13.

(22) *Was ist dein Knecht [...], daß er solch groß Ding tun sollte?*
Was ist dein Knecht [...], daß er so große Dinge tun sollte?
'What is your servant [...], that he should do such great thing(s)?'

The 1545 Luther translation uses the phrase *solch groß Ding*, which is syntactically singular, but can actually refer to more than one entity; this is reflected in the modernized bible version, which has the plural form. In the Berlin manuscript, the wordform *spricht* is an example for an ambiguous mapping due to semantic differences. Where it is mapped to modern *heisst* (which would actually be written *heißt*), it is used in the phrase *daz spricht* following a passage in Latin. In this context, *spricht* does not have the modern meaning of 'speak', but is rather used in the sense of 'this means', which in modern German is expressed as *das heißt*. As the modernization layer of the Anselm corpus also changes wordforms based on semantic differences, this discrepancy arises.

These types of ambiguity in the learned wordlist have different consequences than the cases of *jn* or *das*, though. The latter will result in part-of-speech tags of very different categories, e.g., preposition versus pronoun for *jn*, depending on the chosen normalization. Inflectional differences, such as *ding*/*dinge*, do not change the lexical class of the word, though, as both are nouns. Similarly, if *spricht* is replaced by *heisst* due to semantic differences, as both are verbs, the part of speech still remains the same. Consequently, while the first type of ambiguity will necessarily introduce errors during

the tagging process, the other types will not. Finally, there are cases like the mapping of *komen* in Table 7b which are somewhere in between: the modern *kommen* 'to come' is either a finite or infinitive form of the verb, while *gekommen* is its participle form. These are assigned different tags in the STTS tagset, but are still both verb forms. The evaluation of POS tagging will try to account for this similarity (cf. Sec. 4.1).

To conclude the discussion of Table 7, the observed frequency of the mappings presented for the Berlin text is very low, ranging between 1 and 5 occurrences. In fact, there are only 15 mappings learned from the modernization of the Berlin text with a frequency higher than 5. The most common mapping, in this case, is that of *do* → *da* 'as/there', which was seen 20 times in the training portion. These numbers illustrate that a training corpus of 500 tokens is actually tiny for wordlist mapping (and probably for other algorithms, too), as even a single occurrence of a mapping can greatly influence the outcome of the normalization.

(23)

ORIG:	*do*	*chom*	*czuhant*	*iudas*	*in*	*den*	*garten*
MAPPER:	da	chom	czuhant	judas	ihn	den	garten
NORM:	da	kam	zehant	judas	in	den	garten

'There, Judas immediately came into the garden'

Example (23) shows an excerpt from the Melk manuscript normalized with the wordlist mapping approach. The adverb *do* is normalized correctly, as is the proper noun *iudas*. Both words are relatively common in the Melk text. The wordforms *chom* and *czuhant* were not previously learned and are therefore (incorrectly) left unchanged. Note that *czuhant* is an extinct form meaning 'immediately'; as neither *czuhant* nor its artificial lemma *zehant* exist in modern German, either form is not likely to be useful for part-of-speech tagging. Still, the mapping to a consistent artificial lemma could be helpful for other NLP applications, e.g., alignment of the different Anselm manuscripts.

The mapping of *in* is another interesting case. It is the same in both the original and the gold standard normalization, but was changed by the wordlist mapper to *ihn*. Again, this is the previously discussed example of preposition versus pronoun. Apparently, in the training sample, the historical *in* was more often normalized as the pronoun *ihn*, leading to this erroneous mapping. A contributing factor is that the wordlist mapper does not perform any lexicon lookup to see if the wordform is already covered, and will therefore change even valid modern words if the appropriate mappings have been learned.

Rule-based method Contrary to the wordlist method, the rule-based approach learns rewrite rules on a character level. It also keeps track of characters that did not change and learns them in the form of identity rules. As the majority of characters in a word typically does not need to be changed, identity rules always make up the largest part of the learned rules. This is true even for the Berlin text, which has the lowest baseline to begin with: 723 of 900 rule types (or 80.33%) learned from the first 500 tokens

Rule	Freq.
v → u / #_n	31
o → a / d_#	21
y → ie / d_#	12
y → ei / s_n	12
v → f / #_r	12
y → ei / m_n	11
y → ie / l_b	10

(a) Berlin (norm)

Rule	Freq.
v → u / #_n	40
o → a / d_#	19
ch → k / #_i	9
p → b / #_a	7
iu → jü / #_n	7
e → ng / i_#	7
c → ε / #_z	7

(b) Melk (norm)

Rule	Freq.
ß → ss / a_#	5
y → i / e_n	4
ue → ü / m_s	3
ue → ü / l_c	3
y → i / e_#	2
ue → ü / w_r	2
ue → ü / f_l	2

(c) Gottesdienst

Table 8: Top 7 non-identity rules learned by the rule-based normalizer from the first 500 tokens of Gottesdienst and the Anselm texts (normalization layer)

are identity rules. Furthermore, the 22 most frequent rule types learned from the text are identity rules, too. The numbers for the Melk text are similar (82% identity rules, 16 most frequent rule types are identities).

In order to give an impression of the actual spelling variations that are learned by the rule-based approach, Table 8 lists the top seven non-identity rules for some of the texts. Naturally, the frequency counts are higher than those of the wordlist mapper in Table 7b for the Berlin text (even though the normalization layer is used here). This is because for each word pair, while it provides only one word-to-word mapping, several rules can be extracted from it. However, the frequencies are still relatively low, dropping quickly for the Melk text to only 7 instances for the fourth most frequent replacement rule. The Gottesdienst text is used as one of the examples (Table 8c) because it is the text with the highest baseline. Consequently, identity rules make up almost 95% of all rules learned. As the large majority of words does not change at all, the most frequent non-identity rule was learned only 5 times. This illustrates an important point: the closer a text already is to modern spelling conventions, the more training data is needed to reliably learn the variations it contains. While it may sound counterintuitive at first, the reason lies within the composition of the training data. If a text is already close to modern spelling, it contains fewer training pairs which actually differ from each other. However, only these pairs can provide any information about potential variant spellings.

For the Anselm texts, many different types of spelling variation are found among the most frequent rules. The two most frequent ones chiefly result from the same pairs of words, *vnd* → *und* and *do* → *da*, respectively. These words commonly appear in the Melk text as well as the Berlin manuscript, which is why the top two rules are the same for both. The other common rules actually provide a good impression of the general spelling characteristics of the texts. For Melk, the third rule from the top commonly stems from the word pair *chind* → *kind* 'child'; the spelling variant *ch* → *k*

can also be found in other words, though, such as those in Examples (24) and (25b). Examples (25–28) show common word pairs resulting in rules 4–7 from Table 8b.

(24) (a) *chrafft* → *kraft* 'strength'
(b) *chrankchait* → *krankheit* 'disease'

(25) (a) *pat* → *bat* '[he] asked/begged'
(b) *parmherczichait* → *barmherzigkeit* 'mercy/compassion'

(26) *iungern* → *jüngern* 'disciples'

(27) *gie* → *ging* '[he] went'

(28) *czu* → *zu* 'to'

Spelling characteristics of the Berlin text are quite different. The fifth rule of Table 8a ($v \rightarrow f$ / # _ r), for example, stems from the very common word pair shown in Example (29). The spelling that is captured by the rule, i.e., word-initial *vr* → *fr*, occurs a total of 90 times in the full Berlin text. On the contrary, it is only found three times in Melk and not a single time in any of the other texts. These examples show that examining the rules which are automatically extracted from the texts can indeed provide insights about the individual spelling characteristics they contain.

(29) *vrouwe* → *frau* 'woman'

Another noticeable feature of Table 8a is that four of the highest ranked rules modify the character 'y'. This suggests that 'y' is an extremely common character in the Berlin text, confirming the impression given by Example (1). Indeed, analysis of the Berlin manuscript reveals that it contains the letter 'y' a total of 1,301 times. Judging from Table 8a, its most common occurrences are in the feminine definite article *dy* and the pronoun *syn* (→ *sein* 'his'). The frequency of rules replacing 'y' also demonstrates another characteristic feature of the rule-based method. Evaluation of the complete list of rules shows that in all cases except three, the replacement of 'y' involves the letter 'i'. The rule-based method, however, is not designed to capture this generalization, as it always requires the additional context information to perform the substitution. Therefore, even if the letter 'y' was always replaced by 'i' in the training data whenever it occurred, if it appears in a context in which it was not previously seen, it will not get replaced by the rule-based method during normalization. This is a potential drawback of the approach, as it entails that a higher amount of training data is required in order to capture spelling variants in all possible contexts.

Example (30) shows how this property of the rule-based method manifests itself in practice. It shows an excerpt of the JubelFeste text along with its gold standard normalization, while Example (31) gives the automatic normalizations of both the rule-based method and WLD_{tri}.

(30) *das zweyte was david zur freymuehtigkeit unsers lob-gesanges erfordert*
das zweite was david zur freimütigkeit unseres lobgesangs erfordert
'The second (thing) which David requires for frankness of our song of praise'

Edit	Weight
v → u	0.100
c → ε	0.143
p → b	0.167
o → a	0.184
z → s	0.189
f → ε	0.245
h → k	0.267
u → ü	0.282
y → i	0.291
a → e	0.318

(a) WLD_{uni}

Edit	Weight
v → u	0.100
vn → un	0.100
vnd → und	0.112
c → ε	0.143
p → b	0.169
do → da	0.170
z → s	0.193
o → a	0.211
ai → ei	0.211
z → ss	0.240

(b) WLD_{tri}

Edit	Weight
c → ε	0.143
ch → k	0.263
ch → h	0.366
ch → ε	0.411
ch → che	0.439
ch → gk	0.439
c → g	0.449
ch → cht	0.596
c → t	0.606
c → k	0.606

(c) WLD_{tri}

Table 9: WLD weights learned from the first 500 tokens of Melk (norm); (a) and (b) show the 10 lowest weights learned by WLD_{uni} and WLD_{tri}, respectively; (c) shows all of the WLD_{tri} weights for replacing 'c' or 'ch'.

(31) ORIG: *das zweyte [...] zur freymuehtigkeit unsers lob-gesanges*
RULES: das zweyte [...] zur freimütigkeit unsers lob-gesanges
WLD_{tri}: das zweite [...] zur freimütigkeit unsers lobgesang

Here, the word *freymuehtigkeit* has apparently been learned in the first 500 tokens of the text, as it is normalized correctly by both methods. The word *zweyte*, on the other hand, has not been previously learned. Both wordforms show the same spelling variation *ey* → *ei*, though, which the rule-based approach does not capture, as the context is different both times: in *freymuehtigkeit*, the 'y' has the right context 'm', while it is 't' in *zweyte*. The WLD_{tri} method, however, was able to apply this generalization after training on the same data set.

(32) *chrewcz* → *kreuz* 'cross'

A similar example from the Melk text is (32); here, the spelling variation in question is *cz* → *z*, which often occurs in the word pair *czu* → *zu*. This was already shown above in Table 8b and Example (28). Again, the rule-based method is not able to normalize Example (32) correctly (and simply returns the unchanged wordform) as the contexts are different, while the WLD_{tri} method produces the correct modern word.

(Weighted) Levenshtein algorithm The above examples already highlighted a few differences between the rule-based method and weighted Levenshtein distance. Table 9 shows some example weights extracted by the WLD methods from the normalization

layer of the Melk text. The most frequent rule of the rule-based method (Table 8b) was derived from the common mapping *vnd* → *und*; similarly, the substitution *v* → *u* has the lowest weight associated with it for both the WLD_{uni} and WLD_{tri} algorithm. Even more, all possible *n*-grams for this word pair are represented in the top three rules of WLD_{tri}.[11] The substitution *p* → *b* is ranked high in both methods, too, while *o* → *a* additionally appears here in the form of the bigram mapping *do* → *da*. Notable differences include the mapping *z* → *s*, which is not represented in the top-ranked rules, while the frequent rule *iu* → *jü* is not reflected in the top ten rules of WLD_{tri}.

Also, it quickly becomes clear why the weighted Levenshtein method is able to handle the spelling 'cz' in Example (32) correctly: the simple deletion of the character 'c' has one of the lowest weights (0.143) for both the unigram and trigram method. Looking at the first 500 tokens of the Melk text, when the character 'c' is found in the original but not the normalization, it occurs mostly in the context of *cz* → *z* or *ch* → *k*. It is easy to see how the generalization to the deletion of 'c' can be derived from these mappings. However, in the latter case, the preferable (unigram) substitution to be learned would be *c* → *k* followed by a deletion of 'h', as this is more historically and linguistically accurate. This would require the character alignment process to opt for the alignment in Example (33a); apparently, though, it is the alignment in (33b) which is preferred by the algorithm. This can also be seen from the relatively high ranking of the substitution *h* → *k* in Table 9a.

(33) (a)
c	*h*	*i*	*n*	*d*
k		*i*	*n*	*d*

(b)
c	*h*	*i*	*n*	*d*
	k	*i*	*n*	*d*

Section 3.1.4 described the process by which alignments are selected; it is used exactly for the purpose to properly disambiguate cases like Example (33). Why, then, is the wrong alignment chosen in this case? Table 9c shows that the alignment *c* → *k* was given a weight and therefore must have been seen during training. Indeed, an unambiguous case of this alignment occurs exactly once in the training data in form of the word pair *creatur* → *kreatur* 'creature'. However, there are three occurrences of the word *parmherczichait* 'mercy/compassion', already shown in Example (25b), which—assuming standard Levenshtein alignment—only has the single optimal alignment shown in Example (34).

(34)
parmherczi	*c*	*h*	*ait*
barmher zi	*g*	*k*	*eit*

In this example, if the alignment *c* → *k* was to be used, this would require both the insertion of 'g' and the deletion of 'h', increasing the Levenshtein distance. Such an alignment would therefore not even be considered, as it is assumed to be suboptimal.

[11] The mappings *n* → *n*, *d* → *d*, and *nd* → *nd* cannot be included as they are pure identity mappings, which always have a weight of zero.

This example shows the limits of Levenshtein-based character alignment: unless the correspondence of ‘c’ and ‘k’ was somehow hardcoded into the algorithm, or unambiguously represented in the training data more often, it is hard to see how an automatic alignment method would arrive at a different solution than (34). It also shows that distance-based methods are less adequate if differences in spelling are manifold. A conceivable solution is to place less emphasis on single-character substitutions in such cases, as they might be seen as less reliable when multiple adjacent substitutions occur; this is the same philosophy that is already employed by the rule-based approach, though.

While the low weight of ‘c’ deletion is beneficial for wordforms like *chrewcz*, there is a danger of overgeneralization. Table 10 shows normalizations for three other wordforms found in Melk. The historical wordforms *chust* and *chom* are two more examples of *ch* → *k* substitution; however, none of the distance-based algorithms is able to arrive at the correct normalizations *küsste* ‘kissed’ and *kam* ‘came’ (and neither is the rule-based method or the wordlist mapper, in this case). Even though the substitution has been learned, the low weight of the ‘c’ deletion causes it to be preferred if at all possible. WLD_{uni} wrongly normalizes *chust* → *hut* ‘hat’, which is not too surprising: it has no concept of bigrams, so after the deletion of ‘c’, it will always try to leave the character ‘h’ intact if this leads to a (cheaper) solution. The trigram-based variant will try the same thing in this case, though, as the operation $c \rightarrow \varepsilon$ has a lower weight than all possible alternatives (cf. Table 9c).

ORIG:	*chust* ‘kissed’	*chom* ‘came’	*sach* ‘saw’
NORM:	**küsste**	**kam**	**sah**
WLD_{uni}:	hut	ham	**sah**
WLD_{tri}:	huste	ham	**sah**
LEVEN:	chus	chrom	bach

Table 10: Example normalizations with distance-based algorithms, taken from Melk (norm); correct normalizations shown in bold.

The same principle applies to the second example, *chom*. The wrong normalization *ham* produced by both WLD algorithms reveals yet another fundamental problem, though. At first glance, it is unclear how this wordform can be produced at all, as it is not a modern German word.[12] Remember that the distance-based approaches can only ever produce a wordform that is covered by the supplied lexicon. The lexicon used for this evaluation is a combination of wordforms from DMOR (Schiller, 1996) and the vocabulary of the Luther bible. It is the bible which supplies the basis for this incorrect normalization, and also for the equally strange-looking *chus*: Ham was a son of Noah, while Chus was a son of Ham; both are mentioned in Genesis 10,1–6. As all lexicon entries are lowercased, they cannot be identified as proper nouns anymore—however, as capitalization is an unreliable factor in historical texts, treating all words as lowercase

[12] In colloquial and/or dialectal speech, *ham* can be a contraction of *haben* ‘to have’, but such examples are not included in the lexicon.

is probably unavoidable. The actual problem lies in the fact that lexical frequency is not considered at all. In a balanced text corpus, the verb *kam* will be significantly more common than the proper noun *Ham*. Hence, these wrong normalizations suggest that (weighted) Levenshtein distance should ideally not be the only criterion for determining a normalization, but lexical frequency should be in some way factored in.

The third example in Table 10, *sach* → *sah*, shows an example of 'c' deletion applied correctly by both WLD algorithms. The suggestions of the Levenshtein algorithm, however, are nonsensical in all three cases, underlining the fact that it is not very useful on its own. For this reason, it will not be analyzed in more detail here.

A characteristic feature of distance-based normalization is that it will always produce a correct modern wordform, no matter how inappropriate it is. This is because the algorithm always returns the word in a lexicon with the lowest distance to the input string—as long as there is at least one word in the lexicon, the algorithm cannot fail to find a normalization candidate. However, this is only useful to a certain extent: if the spelling is very far from the modern word, or the correct normalization is not covered by the lexicon, even the WLD algorithm can produce very strange suggestions, as in these examples from the Berlin text:

(35) ORIG: *anshelme*
NORM: anselm
WLD: asel

(36) ORIG: *cruczegete*
NORM: kreuzigte '[he] crucified'
WLD: kruste 'crust'

Example (35) illustrates how normalization can fail if the target word is not contained within the lexicon, which can often happen with proper nouns, such as *anselm* in this case.[13] Note that while the rule-based method suffers the same handicap, as it also relies on the lexicon to verify its normalization candidates, it is less likely to produce a candidate form as different from the input form as the one in (35). Instead, it is more likely to fail and return *anshelme* unchanged. Still, a better solution might be to enrich the modern lexicon with proper nouns commonly found in the text. As all texts in the Anselm corpus refer to Saint Anselm in some way, and quite frequently so, creating a list of proper nouns and adding them to the lexicon more likely results in the correct normalization of the wordform *anshelme*, thereby noticeably improving overall accuracy. On the other hand, if a proper noun occurs frequently in a text, simple wordlist mapping is likely to be the better method to handle these cases without any modification of the lexicon.

In Example (36), the target word is covered by the lexicon, but its historical spelling is apparently too distant for WLD (both unigram and trigram variants) to normalize it correctly. The suggested candidate does not have much in common with the input word:

[13]The suggested normalization *asel* is, again, a character from the bible (1 Chronicles, 8,38).

ORIG:	*umstaende* 'circumstances'	*vsseczyk* 'leprous'	*iungern* 'disciples'	*hynmel* 'heaven'
NORM:	**umstände**	**aussätzig**	**jüngern**	**himmel**
WLD_{uni}:	umstand	fetzig	ungern	**himmel**
WLD_{tri}:	**umstände**	**aussätzig**	**jüngern**	heimel
LEVEN:	**umstände**	steck	ungern	**himmel**

Table 11: Example normalizations with distance-based algorithms, taken from JubelFeste, Berlin (norm), and Melk (norm); correct normalizations shown in bold.

it does not have the same part of speech; it is not semantically related; and it is not close to the spelling of the original wordform. Suggestions like these could be at least prevented by setting a cutoff for the WLD algorithm. A cutoff mechanism would break out of the algorithm if the calculated distance gets larger than a certain threshold, and return the original wordform instead. This way, it could "fail", similar to the mapper and the rule-based method. However, the question of how to define the threshold is not a trivial one. What can be considered a "too high" distance depends on several factors, e.g., the average weight of the learned edit operations, and the length of the input word. Especially the latter has to be taken into account in order not to preclude perfectly correct normalizations with a high number of spelling differences, such as in Example (37), which is also from Berlin (norm) and is correctly normalized by both WLD variants.

(37) *iuncvrouwen* → *jungfrauen* 'virgins'

Another feature of distance-based methods is that a historical wordform which is already identical to a modern one will never be changed. This can sometimes be inappropriate, too, as Example (38) from the LeichSermon text demonstrates: the historical form *wehrt* corresponds to the modern *wert* 'worthy' in this context, but is not modified as *wehrt* is a valid modern wordform meaning '[he] resists'. This is a general problem of context-free normalization, though, and is not necessarily restricted to distance-based algorithms—neither the rules nor the wordlist mapper changed the input word in this case.

(38) ORIG: *wir sind nicht wehrt*
NORM: wir sind nicht wert 'we are not worthy'
WLD: wir sind nicht wehrt

Finally, the accuracy scores in Table 5 (p. 26) already showed that WLD_{uni} performs worse than WLD_{tri} for all texts. Examples from the normalizations, as shown in Table 11, mainly confirm this observation. In a few cases, such as *umstaende*, standard Levenshtein

distance even returns the correct result where WLD_{uni} does not. This happens mainly for newer texts with less spelling variations (the example is from JubelFeste). Cases where WLD_{uni} produces the correct normalization and WLD_{tri} does not are exceptional, but do happen, as the right-most example (*hynmel*) shows.

3.2.3 Effect of training corpus size

Up to this point, normalizations of Anselm and Sermon texts have been evaluated after training on the first 500 tokens of the respective texts, which is a more or less arbitrarily chosen size. In the context of a research project, this data would have to be normalized manually first to provide the normalizers with training material. However, manual normalization can be a tedious and time-consuming process, and whether manual normalization of 500 tokens is a realistic goal might depend on the number of texts to be processed and the available resources. Also, when comparing the normalization algorithms, the factor of size of the training corpus has been neglected so far. Comparisons could only be made between the Luther corpus, which has a comparatively huge training part, and all other texts.

Therefore, in this section, a different evaluation method is chosen: training is performed on the gold standard normalizations of the first n tokens, with increasing values for n; afterwards, the trained methods are evaluated on the remaining part of the same text. This approach reflects a process than can conceivably be employed in an actual research project.[14] It allows an assessment of the question whether accuracy of normalization can be significantly improved by manually normalizing a sample of the text, and if so, how much training data is required. For researchers building a corpus of historical texts, this could turn out to be an attractive tradeoff between the extremes of an all-automatic and an all-manual normalization of the data.

For the Anselm texts, the normalization methods are successively trained on the first 100, 250, 500, 1,000, and 2,000 tokens of the texts. 2,000 tokens are chosen as the upper bound because this is almost half of the texts, and the benefit of a normalization method that requires more tokens for training than it actually normalizes is doubtful, at least in terms of efficiency. For the same reason, 1,000 tokens are chosen as the upper bound for the Sermon texts, which is also about half of their size.

It should be noted that as the size of the training portion increases, the size of the evaluation portion decreases. This means that the accuracies resulting from different training sizes are not based on exactly the same evaluation data. For this reason, the Levenshtein algorithm (cf. Sec. 3.1.3) is also evaluated separately for each training size, even though it does not actually learn anything. Another effect of this evaluation method is that with more training data, accuracy sometimes goes down. These fluctuations

[14] As the normalizers operate context-free, cross-validation is a possible—and probably better—alternative. Due to time contraints, cross-validation has not been carried out for all scenarios evaluated here, though Bollmann (2013) reports an evaluation of the best chain combination method (cf. Sec. 3.3) using cross-validation.

usually correspond to similar changes in the baseline score and are not statistically significant; i.e., while the performance of normalization does not improve in these cases, it does not actually get worse, either. While it makes comparison of results between different training sizes a bit more difficult, this method—again—more accurately represents real-life application scenarios, as it reports the accuracy of normalization on that part of the text which would otherwise have to be processed manually.

As the evaluation does not take the already normalized training parts into account, this means that the accuracies for the whole texts are always higher than the figures reported here. Therefore, if a normalization method reports 70% accuracy when trained on about half of a text, the accuracy for the text as a whole is actually closer to about 85%. This is, of course, relevant for the discussion of part-of-speech tagging (cf. Sec. 4), as POS tagging is always evaluated on full texts, but is not considered here as it would obscure the effects of the actual automatic normalization.

Table 12 lists the complete results for each combination of text, training size, and normalization method. The most evident result from this evaluation is that, in terms of absolute accuracy, either the rule-based method or WLD_{tri} performs best in every single case. However, in many cases other normalization methods cannot actually be shown to be significantly worse; these results are marked in italics in the table. Most often, this also applies to the rule-based method or WLD_{tri}, i.e., they sometimes both perform comparably well. Also, the wordlist mapper always stays behind the rule-based method in terms of total accuracy, but in a few cases, this difference is not statistically significant, either. For the Sermon texts, when the training portion is small, the unigram WLD model is not significantly worse than the trigram version. This is not surprising, as the baseline is relatively high for these texts, so more tokens need to be processed to reliably learn the spelling variations. Also, spelling differences are likely to be smaller and restricted to single letters more often than in older texts, reducing the advantage of learning trigrams over unigrams.

An observation that holds true in all evaluated cases is that for small training portions, the WLD_{tri} method is always best (sometimes among other, comparably good results), while for large training portions, the same usually holds true for the rule-based method. Typically, when considering the same text with increasing training corpus sizes, WLD_{tri} starts out performing best, followed by a "turning point" where both the rule-based method and WLD_{tri} perform similarly well, with the rule-based method winning out at the end with the larger training parts. This turning point can be found at only 100 tokens for training with the LeichSermon text, and at 250 tokens for JubelFeste, for example. For the Gottesdienst text, the rule-based method is never the absolute best, but not significantly worse on the largest training sample with 1,000 tokens. It is conceivable that, if the Gottesdienst text were longer and even more training data would be used, the trend would continue, with the rewrite rules producing the best normalization.

		Baseline	Mapper	Rules	Leven	WLD$_{uni}$	WLD$_{tri}$
Berlin (norm)	100	23.24%	37.09%	40.20%	37.52%	47.26%	**55.78%**
	250	23.12%	56.38%	*59.96%*	37.44%	51.37%	**61.35%**
	500	23.05%	*62.05%*	**63.17%**	37.17%	51.14%	60.71%
	1000	23.14%	*66.46%*	**67.65%**	37.19%	51.43%	63.24%
	2000	22.30%	72.41%	**75.07%**	36.74%	50.48%	65.41%
Berlin (mod)	100	21.24%	34.28%	35.61%	35.02%	42.20%	**50.70%**
	250	21.19%	51.21%	*54.47%*	35.01%	46.58%	**56.52%**
	500	21.12%	*58.05%*	**58.90%**	34.71%	47.55%	56.14%
	1000	21.35%	*62.22%*	**63.62%**	34.76%	47.65%	58.46%
	2000	20.81%	*68.52%*	**70.48%**	36.67%	47.11%	61.07%
Melk (norm)	100	39.54%	57.10%	59.78%	54.45%	64.67%	**68.21%**
	250	39.62%	61.94%	65.07%	54.44%	65.88%	**68.98%**
	500	39.32%	63.15%	64.14%	54.10%	65.40%	**69.34%**
	1000	39.11%	66.56%	*69.19%*	53.94%	65.86%	**69.70%**
	2000	39.75%	*70.84%*	**71.86%**	54.94%	67.02%	*70.72%*
Melk (mod)	100	36.05%	53.70%	54.76%	50.42%	59.90%	**62.80%**
	250	36.12%	56.79%	60.80%	50.43%	61.24%	**63.88%**
	500	35.91%	57.81%	60.41%	50.16%	60.55%	**64.56%**
	1000	35.72%	61.03%	*64.36%*	49.90%	60.83%	**64.47%**
	2000	36.09%	*67.10%*	**67.18%**	50.73%	61.87%	*65.49%*
LeichSermon	100	72.95%	73.72%	*75.84%*	71.27%	*75.79%*	**77.00%**
	250	73.13%	75.26%	**78.22%**	71.68%	*76.61%*	*77.18%*
	500	72.71%	*76.46%*	**78.67%**	70.80%	75.63%	*76.40%*
	1000	71.39%	77.08%	**82.26%**	69.86%	75.13%	76.32%
JubelFeste	100	79.82%	82.38%	81.74%	84.19%	*86.30%*	**86.99%**
	250	79.81%	83.57%	*85.00%*	84.00%	*86.01%*	**86.59%**
	500	79.47%	85.22%	**88.52%**	84.06%	86.07%	*88.15%*
	1000	78.80%	86.46%	**90.77%**	83.91%	86.28%	88.13%
Gottesdienst	100	83.81%	84.08%	83.81%	87.91%	*91.64%*	**93.20%**
	250	83.56%	84.73%	87.32%	87.90%	90.96%	**94.36%**
	500	83.41%	86.58%	90.50%	87.89%	93.05%	**95.46%**
	1000	84.05%	89.51%	*94.01%*	87.72%	93.07%	**95.80%**

Table 12: Normalization accuracy per text after training on the first *n* tokens and evaluating on the rest, ignoring punctuation and foreign words; best result for each row is highlighted in bold, differences to the best result that are not statistically significant ($p > 0.05$) are marked in italics.

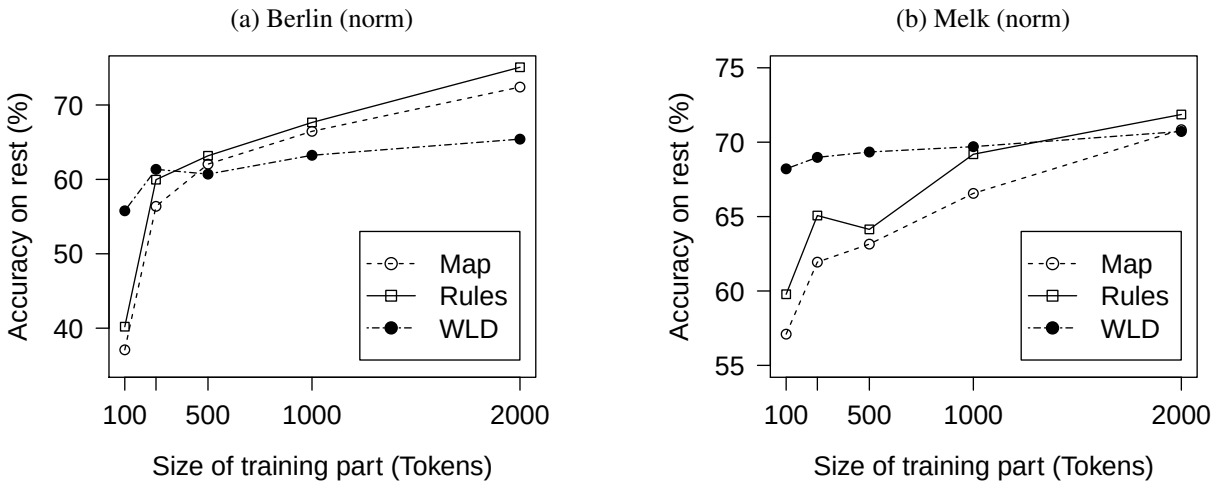

Figure 1: Normalization accuracy for different sizes of the training part

Figure 1 illustrates the progression of the accuracy scores for the Anselm texts. It shows that while the rule-based method tends to perform better with more training data, the point at which it overtakes WLD_{tri} is quite different. It only takes 250 tokens for the rule-based method to perform similarly well on the Berlin text, while for the Melk text, 1,000 tokens are needed. A possible explanation lies in the types of spelling difference found in the texts: the WLD algorithm always has a preference for leaving characters unchanged, while character rewrite rules have no such bias unless this tendency is explicitly learned. Consequently, if many characters need to be changed, the rule-based approach likely has an advantage. This could explain the difference between Berlin and Melk: a common word in both texts is modern *frau* 'woman' to refer to the Virgin Mary; in Melk, it is commonly spelled *fraw*, while Berlin has *vrouwe*. It is apparent that the former is much closer to the normalization and easier to handle with distance-based methods. The WLD approach will always have an advantage over rewrite rules if such spelling variations (e.g., $w \rightarrow u$) appear regularly and in many different contexts, as it will quickly make generalizations, while it is at a disadvantage if the word lengths are very different or the spelling variants are relatively special (deletion of 'we' occurs only with *vrouwe* and not anywhere else). Note that this factor is not directly tied to baseline accuracy—few words with lots of spelling variation will result in a higher baseline than many words with few variations. This is shown by LeichSermon (cf. Table 12): even though its baseline is much higher than that of Berlin or Melk, the rule-based method is already among the best at only 100 tokens.

Figure 1 also highlights another interesting feature of the WLD_{tri} normalizer: while it tends to be outperformed by the rule-based method at some point, it already reaches comparatively high accuracies from 100 training tokens only. This effect is not quite as

Rule	Freq.
v → u / #_n	7
ε → s / l_t	6
y → ei / d_n	4
ε → l / i_l	4
ε → d / #_u	4
a → ol / s_l	4
z → se / #_e	3
ow → au / b_m	3
y → i / m_t	2
y → i / e_n	2

(a) 100

Rule	Freq.
v → u / #_n	17
ε → s / l_t	8
v → f / #_r	7
y → ei / s_n	6
o → a / d_#	6
a → ol / s_l	6
y → i / m_t	5
y → ei / d_n	5
we → ε / u_#	5
o → a / r_u	5

(b) 250

Rule	Freq.
v → u / #_n	57
o → a / d_#	53
v → f / #_r	28
y → ie / l_b	27
y → ei / m_n	27
t → d / n_#	25
o → a / r_u	23
we → ε / u_#	22
y → ei / s_n	19
y → i / k_n	18

(c) 1,000

Table 13: Comparison of the 10 most frequent rules with differently sized training parts, learned from Berlin (norm); subcaptions indicate number of tokens for training.

pronounced with the Berlin text, where it starts at about 55% and ends at 65% accuracy (with 2,000 tokens for training). For all other texts however, the difference between 100 tokens and 1,000/2,000 tokens for training is only between 1 and 3 percentage points. With the LeichSermon text, there is no statistically significant difference at all between the different sizes of the training corpus ($p > 0.05$). These figures suggest that WLD_{tri} actually requires very little training data to successfully perform normalization, with the restriction that spelling variations should not be too irregular or distant from the target language. The rule-based method, on the other hand, is typically better with high amounts of training material.

Considering Table 12 once more, it confirms the tendency for Levenshtein distance to generally perform worse than other methods. A notable exception is JubelFeste and Gottesdienst, where it is sometimes better than the wordlist mapper and closer to, e.g., WLD_{uni} than with the other texts. Again, this is a consequence of the relatively low amount of spelling difference to be found there. If many regular spelling variations occur (*oe* → *ö* is common in JubelFeste, for example), it is easier for the Levenshtein-based measures to arrive at the correct normalization. The wordlist mapper, on the contrary, can make no generalizations at all and is therefore at a disadvantage under these circumstances.

To give an impression of how the size of the training corpus affects the rules learned, Table 13 gives an overview of the top ten rules learned from Berlin (norm) with different numbers of tokens for training (the top seven rules for a training size of 500 tokens were

	Baseline	Mapper	Rules	WLD_{uni}	WLD_{tri}
Berlin (norm)	23.05%	29.81%	32.02%	43.00%	**45.30%**
Berlin (mod)	21.12%	27.60%	29.91%	40.02%	**42.15%**
Melk (norm)	39.32%	49.39%	51.71%	57.70%	**59.35%**
Melk (mod)	35.91%	46.29%	47.68%	53.23%	**54.53%**
LeichSermon	72.71%	77.13%	77.73%	80.62%	**80.85%**
JubelFeste	79.47%	82.87%	82.55%	87.18%	**88.07%**
Gottesdienst	83.41%	83.56%	84.18%	90.42%	**90.68%**

Table 14: Normalization accuracy per text after training on Luther, ignoring punctuation and foreign words; best result for each text highlighted in bold.

already shown in Table 8a). All rules to normalize the example *vrouwe* → *frau* discussed above can be found among the top ten with only 250 tokens for training (Table 13b, third row and final two rows).

3.2.4 Using the Luther corpus for training

Evaluation showed that using more training data typically results in a higher normalization accuracy, especially for methods like the rule-based normalizer and the wordlist mapping approach. If, however, gold standard normalizations have to be created for a part of the text in order to achieve good results, there is a limit on the amount of training data that can be used. With the largest size of the training portions that was considered in the last section (2,000 tokens for Anselm and 1,000 tokens for Sermon), almost half of all texts in question would have to be normalized manually beforehand. The required effort should not be underestimated, and might not always be realizable in practice. On the other hand, this raises the question whether normalizers can be trained on other resources and successfully applied to different texts. One such resource could be the Luther corpus: the training part of Luther is roughly a hundred times larger than the largest training part considered for Anselm.

Table 14 shows evaluation results on the full Anselm and Sermon texts after training on the Luther training corpus. The figures for the Anselm texts are rather disappointing: even the method with the highest accuracy is still well below the numbers that can be achieved with as little as 100 training tokens of the same text (cf. Table 12). However, the Anselm texts have a few characteristics which make a direct transfer of the information extracted from Luther quite difficult: they are manuscripts with diplomatic transcriptions, whereas the bible is a carefully printed work, and were written considerably earlier. This is also reflected in the lower baseline score compared to Luther.

Results for the Sermon texts are a bit more promising. For LeichSermon and JubelFeste, applying the WLD_{tri} algorithm that has been trained on Luther results in a better accuracy than training with the first 250–500 tokens of the texts themselves. Still, the results do not exceed the best scores reported in Table 12 when more training data is used. Also, accuracies for Gottesdienst are worse in all cases, showing that the date of a text is not the only factor in deciding whether applying knowledge learned from the Luther corpus is successful or not. Of course, it is to be expected that the type of spelling variation also plays a role—judging from the results, LeichSermon and JubelFeste are likely to be closer to the spelling of Luther than Gottesdienst is.

An interesting fact from Table 14 is that WLD_{tri} performs best for all texts. This seems to contradict the impression that the rule-based method gets better than WLD_{tri} when more training data is available—after all, both methods were trained on about 220,000 tokens here. However, this is not necessarily the case. Rather, the rule-based method learns very specialized character replacements, relying heavily on context information. As a consequence, it does not only require a certain amount of training data to work best, but also needs to be more closely adapted to the specific spelling characteristics of a text. Therefore, it profits less from training on a different text with (even slightly) different characteristics, while WLD_{tri} is able to transfer learned substitutions more reliably to other (similar) data. When working with many different texts that have similar, yet distinct enough spellings, this may be an argument in favor of using WLD_{tri}.

3.3 Combining normalization methods

So far, all normalization methods have only been evaluated in isolation. If every normalization algorithm has particular strengths and weaknesses, it seems profitable, though, to combine them in some way. Combinations of algorithms are successfully used by Jurish (2010) and the VARD normalization tool for Early Modern English (Baron and Rayson, 2008). A first evaluation in Bollmann (2012) has also confirmed that combining techniques indeed improves the results noticeably. There, normalizers were combined in form of a "chain": the normalizer that is first in the chain is always called first; only if it is not able to successfully produce a normalization, the second one is called, and so on. Let us recapitulate what it means for a normalization algorithm to "fail": wordlist mapping can fail if the historical wordform to be normalized cannot be found in the wordlist; rule-based normalization fails if no wordform in the target lexicon can be reached using the learned transformation rules (e.g., due to a previously unseen combination of characters). The distance-based methods can never fail, as there is always a wordform in the lexicon which is closest to the input string.

In addition to these chain combinations, it is also conceivable to use a "majority vote" approach. Here, no single normalization method is favored. Instead, either all or a

	Best Single	Chain Combination			Majority Vote	
		M,R,W_3	M,W_3	R,(M,)W_3	M,W_1,W_3	All
Berlin (norm)	63.17%	**75.07%**	*74.24%*	70.14%	67.48%	46.07%
Berlin (mod)	58.90%	**69.98%**	*69.21%*	65.33%	62.71%	42.76%
Melk (norm)	69.34%	**74.49%**	*74.12%*	71.49%	*72.78%*	61.12%
Melk (mod)	64.56%	**68.30%**	*68.10%*	*67.01%*	*66.52%*	54.74%
LeichSermon	*78.67%*	**79.56%**	*79.50%*	*79.26%*	*79.02%*	73.30%
JubelFeste	*88.52%*	**91.81%**	*90.59%*	*91.45%*	*90.04%*	*88.52%*
Gottesdienst	*95.46%*	*95.73%*	*95.66%*	**95.87%**	*94.98%*	89.19%

Table 15: Normalization accuracy per text using combinations of normalization methods, training on the first 500 tokens and evaluating on the rest, ignoring punctuation and foreign words; M = Wordlist mapper, R = Rule-based method, W_1 = WLD_{uni}, W_3 = WLD_{tri}; "Best Single" refers to the previously best result (bold number from Table 5) using a single normalization method; best result for each row is highlighted in bold, differences to the best result that are not statistically significant ($p > 0.05$) are marked in italics.

selection of normalization methods is called, and if the majority of algorithms produces the same candidate wordform, this wordform is used as the final normalization. Some kind of tie resolution can still become necessary, though, if each normalizer produces a different output string. In this evaluation, ties are resolved by defining an order of preference. The order of "Mapper, WLD_{tri}, Rule-based, WLD_{uni}, Levenshtein" was empirically found to be best; e.g., if there is a tie between WLD_{tri} and the rule-based method, the normalization of WLD_{tri} is chosen as it comes first in the given ordering.

Table 15 shows the results for a selection of the possible combinations of normalizers. Chain combinations including Levenshtein normalization or the WLD_{uni} method are not shown as they always perform worse than the same combinations using the WLD_{tri} algorithm instead. Note that the WLD_{tri} method can only appear at the end of a chain as it cannot fail to produce a result; hence, subsequent normalizers in the chain would never be called. For majority vote, only two scenarios are shown: using wordlist mapping, WLD_{uni}, and WLD_{tri}, which is the best combination for majority vote for most texts; and using all five normalization methods. For reasons of clarity, only the versions that were trained on the first 500 tokens of the respective texts are shown; except for very small training sizes, the general trend of the results is the same for other sizes of the training corpora, too.

The first thing to note is that, especially for the Sermon texts, many differences in

normalization accuracy reported here cannot be shown to be statistically significant. This being said, the chain combination of using wordlist mapping first, then rule-based normalization, then WLD_{tri} (column "M,R,W_3"), gives the best accuracy in all cases but one. It is conceivable that the differences can be shown to be significant only with larger text sizes. In addition to that, chain combinations are always better than the best single normalization method alone. Therefore, while these figures do not provide conclusive evidence, they strongly suggest that this type of chain combination is the best for the given set of normalization methods.[15]

The same tendency also holds when comparing chain combinations with the majority vote approach. In all cases, the best result of a majority vote is still worse than that of the best chain. The difference is larger for texts with lower baselines (e.g., 75.07% vs. 67.48% with Berlin (norm)) and tends to become less significant with more modern texts (Sermon). Using majority vote with all five normalization methods discussed here is yet considerably worse. These results show that a bias towards specific methods is actually preferable to a majority vote where all methods are treated equally (as long as there is no tie).

Example (39) shows an excerpt from the Melk text which is normalized correctly by the best chain combination. The third line shows which normalization method is responsible for the given normalization. It also demonstrates that all algorithms actually get called during the normalization process. In the given passage, each of these normalization methods on its own would have introduced at least one error, too: WLD_{tri} normalizes *waz* 'was' to *was*, while the other methods cannot handle the phrase *symones weissagueng* 'Simon's divination'.

(39)

es	*waz*	*symones*	*weissagueng*	*dennoch*	*nicht*	*volpracht*
es	war	simons	weissagung	dennoch	nicht	vollbracht
MAP	MAP	WLD_{tri}	WLD_{tri}	RULES	RULES	RULES

'Simon's divination was still not fulfilled/accomplished'

When evaluated in isolation (cf. Table 5), the wordlist mapper usually performs worse than the rule-based method. Still, evaluation of chains shows that using the mapper before the rules is actually better than the other way around. This is not a contradiction, though, when taking into account how these algorithms work. The rule-based approach has a higher granularity than the wordlist mapper, as both can be said to learn replacement rules, but on different levels (character-based vs. word-based). Consequently, if the rule-based method fails to find a normalization, the wordlist mapper will also fail—if the wordlist included a suitable mapping, the rule-based method would have learned the applicable rules, too. This is why in the combination of the rule-based method, wordlist mapper, and WLD_{tri} (column "R,(M,)W_3" in Table 15), there is no

[15] For an evaluation of the best chain combination method ("M,R,W_3") with different training sizes and additionally using cross-validation, see Bollmann (2013). For a training size of 500 tokens, the results obtained using cross-validation are very similar to those reported here, and in case of the Berlin text even significantly better (approx. 79%).

difference in the accuracy score if the wordlist mapper is left out (hence it is given in brackets).

On the other hand, using the wordlist mapper before applying rewrite rules is still better than leaving it out completely. This also follows from the properties mentioned above. As wordlist mapping has the lowest granularity, it will fail most often, but normalize those wordforms that it has learned with a relatively high accuracy. The rule-based approach is more fine-grained as it operates on character *n*-grams, and will be able to produce a normalization more often. Finally, the WLD_{tri} method has the highest granularity by comparison, as it does not impose the strict context restrictions that are used by the character rewrite rules. Hence, what the results actually imply is that going from lower to higher granularities of normalization methods—i.e., from larger to smaller units of operation—is the most promising way of approaching the normalization problem. Further research could try to utilize this assumption by using even more gradations of input string length (e.g., considering *n*-grams with decreasing values for *n*, starting with the length of the input wordform) in the normalization process.

4 Part-of-speech tagging

This section discusses aspects and results of part-of-speech (POS) tagging on historical data. Normalization as discussed in the previous section can be useful by itself, e.g., for purposes of information retrieval: it allows the user of a corpus to perform easier and more reliable search queries on historical data. If, on the other hand, the ultimate goal of a research project is to create a morphologically or syntactically annotated resource, normalization might only be seen as an intermediate step to achieve this result. The evaluation of POS tagging performance on normalized data therefore serves to analyze how useful normalization actually is for this purpose. Additionally, even gold standard normalizations of historical data feature some characteristics which complicate the tagging process. Those will also be analyzed here in more detail.

Section 4.1 first presents the POS taggers included in the evaluation and gives more information about the evaluation procedure. Section 4.2 tests the impact of removing capitalization and punctuation—a common handicap for processing historical texts—on modern data. Section 4.3 discusses POS tagging results on the original texts and the gold standard normalizations, while Section 4.4 presents an evaluation on the automatically normalized data.

4.1 Methods and procedure

Most POS taggers are able to annotate modern German data with high accuracy; e.g., Schmid (1995) reports an accuracy of 97.5% for an optimized version of TreeTagger. However, it is not clear whether better performance on modern data equals better

performance on historical data, or data that has been automatically normalized. Hence, different POS taggers are included and compared in this evaluation. The taggers that were chosen for the evaluation are:

1. TreeTagger[16] (Schmid, 1994, 1995), a probabilistic hidden Markov model (HMM) tagger using binary decision trees to estimate transition probabilities;
2. RFTagger[17] (Schmid and Laws, 2008), a fine-grained HMM tagger using morphological features and decision trees to estimate probabilities separately for each feature value; and
3. MBT[18] (Daelemans et al., 1996), version 3.2.9, a tagger using memory-based learning techniques.

TreeTagger belongs to the family of statistical POS taggers based on Markov models. Its name derives from the fact that it uses decision trees to estimate the transition probabilities of POS tags, i.e., the probability of a given POS tag with respect to the tags immediately preceding it. By default, TreeTagger uses a trigram model for this estimation, i.e., it considers two preceding POS tags to calculate the probability for a POS tag of any given word. A fullform lexicon is stored which provides the candidate tags for a wordform. For unknown words, TreeTagger additionally uses prefix and suffix lexicons to match them to known wordforms. All lexicons are directly built from a training corpus. POS tags are treated as atomic units, i.e., the tagger does not distinguish between the base POS (e.g., NN) and its morphological attributes (e.g., Masc.Nom.Sg). The supplied parameter file for German only tags base POS without morphological attributes; therefore, the training corpus (cf. Sec. 2.2) is also stripped down to base POS tags before training with TreeTagger.

RFTagger is based on the same theoretical concepts as TreeTagger, but uses more fine-grained techniques as it takes morphological attributes into account. POS tags are split up at dot symbols to create attribute sets; the first element is treated as the base POS, while all other elements are treated as attributes dependent on that base POS tag. Similarly to TreeTagger, RFTagger makes use of decision trees for probability estimation, but builds separate trees for each possibly feature value of each attribute. In addition to that, it differs from TreeTagger with respect to the treatment of unknown words. Words are separated into disjunct word classes: by default, these are numeric expressions, capitalized words, lowercase words, and a fourth class for all other tokens. Suffix tries with a maximum suffix length of 7 are built separately for each word class, and unknown words during tagging are only matched against the suffix trie of their respective word class. RFTagger also uses a trigram model (i.e., two precedings tags as context information) by default, but Schmid and Laws (2008) report a higher accuracy

[16] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`
[17] `http://www.ims.uni-stuttgart.de/projekte/corplex/RFTagger/`
[18] `http://ilk.uvt.nl/mbt/`

when increasing the context size to 10 preceding POS tags: up to 97.97% on data from the TIGER corpus.

The Memory-Based Tagger (MBT) uses a different approach than the previously presented taggers: it uses memory-based learning, i.e., "supervised, inductive learning from examples" (Daelemans et al., 1996, p. 15). Instead of trying to derive a statistical model from a training corpus, MBT stores the training data as feature vectors in a kind of internal "memory". For the tagging process, similarity metrics are used to compare the data to already known examples in the memory; the least distant example is then used as the tagging hypothesis. Similarly to TreeTagger and RFTagger, MBT also considers token context, but is not restricted to looking only at the preceding tags: right context can also be taken into account by assigning "ambiguous tags" to these tokens, consisting of the set of possible tags they can be assigned according to the lexicon. The type and amount of context information to use can be freely configured. For unknown words, affixes can be used to relate them to known words. The default settings, which were found to be best by Daelemans et al. (1996), are to use two tags to the left and one (ambiguous) tag to the right as context for known words, and one tag to both sides plus the first and the final three letters of unknown words. MBT claims to have an especially good performance on unknowns: on a Wall Street Journal corpus, an accuracy above 90% for unknown words was achieved (Daelemans et al., 1996).

For the experiments below, all taggers are used with their default settings, with one exception: the RFTagger is additionally tried with a context size of 10 (instead of the default 2). The TIGER/Tüba corpus described in Section 2.2 is always used to train the taggers. For TreeTagger and MBT, only base POS is used, while RFTagger is trained with the full POS annotation, as the decomposition of tags and use of morphological attributes is a distinctive feature of the tagger. Only base POS is used for the evaluation, though, as neither the Anselm nor the Sermon texts are annotated with morphological information. Table 16 shows the average accuracy of tenfold cross-validation on the TIGER/Tüba corpus using the different taggers.

	Accuracy
RFT	97.04%
RFT_{10}	97.26%
TREE	96.48%
MBT	96.78%

Table 16: Tagging accuracy of different taggers on modern data (average of 10-fold CV)

In addition to the different POS taggers, different gradations of the tagset will be considered. The tagset used by all corpora in this thesis closely follows the STTS (Schiller et al., 1999), which is a relatively fine-grained tagset containing 54 tags. Tags in the STTS sometimes already encode complex information. Consider, for example, the tag VAFIN: it contains information about word class (V = verb), subtype (A = auxiliary), and grammatical category (FIN = finite). When these distinctions are already made

STTS	**STTS$_{red}$**	**Universal**
ADJA, ADJD	ADJ	ADJ
ADV	ADV	ADV
APPO, APPR, APPRART, APZR	AP	ADP
ART	ART	DET
CARD	CARD	NUM
FM	FM	X
ITJ	ITJ	X
KOKOM	KOKOM	CONJ
KON	KON	CONJ
KOUI, KOUS	KOU	CONJ
NE	NE	NOUN
NN	NN	NOUN
PDAT, PDS	PD	PRON
PIAT, PIS	PI	PRON
PPER	PPER	PRON
PPOSAT, PPOSS	PPOS	PRON
PRELAT, PRELS	PREL	PRON
PRF	PPER	PRON
PAV	PAV	PRON
PTKA	PTKA	PRT
PTKANT	PTKANT	PRT
PTKNEG	PTKNEG	PRT
PTKVZ	PTKVZ	PRT
PTKZU	PTKZU	PRT
PWAT, PWAV, PWS	PW	PRON
TRUNC	TRUNC	X
VAFIN, VAIMP, VAINF, VAPP	VA	VERB
VMFIN, VMINF, VMPP	VM	VERB
VVFIN, VVIMP, VVINF, VVIZU, VVPP	VV	VERB
XY	XY	X
$(	$(	PUNCT
$,	$,	PUNCT
$.	$.	PUNCT

Table 17: Mappings from the STTS tagset (Schiller et al., 1999) to the reduced STTS tagset (STTS$_{red}$) and Universal tagset (Petrov et al., 2011).

in the base POS tag, there is typically a difference in inflectional form or syntactic distribution which distinguishes elements of that POS tag from those of other, similar tags. Historical language, however, can follow different rules in this regard, and might not always allow the same distinctions to be made. Still, if a tagger confuses the finite and the imperative form (VAFIN vs. VAIMP), this is counted as an error in the same way as confusing VAFIN with a completely unrelated category, e.g., NN (for common nouns). As tagging on normalized historical data introduces more potential sources of errors compared to modern data (e.g., wrong normalizations), knowing the type of these tagging errors becomes much more relevant.

To account for the above-mentioned issue, the STTS tagset is mapped to two smaller tagsets which are considered in the evaluation. Table 17 gives an overview of the mappings from standard STTS to the reduced tagsets. The first tagset is a reduced version of the STTS, referred to as STTS_{red}, which takes away one level of distinction from the majority of POS tags. As an example, auxiliary verbs all receive the POS tag VA, removing the distinction of finite, imperative, or infinitive form. The second tagset takes this one step further, including only one base category VERB for all types and forms of verbs. This "universal" tagset has been proposed by Petrov et al. (2011), who also provide a mapping from STTS to their tagset (from now on referred to as "Universal"). Comparing accuracy on STTS with accuracy on the Universal tagset allows for a better evaluation of the types of errors the tagger makes, as outlined above. Both the mapping to STTS_{red} and Universal are only made for the accuracy evaluation, though; the tagging process itself is always performed using models trained on STTS. Petrov et al. (2011) found this to increase accuracy compared to directly using a model trained on the reduced tagset.

Finally, contrary to standard practice, punctuation marks are always excluded when evaluating tagging accuracy in the following sections. Historical texts do not feature consistent punctuation marks; as a result, texts with and without punctuation marks will be tagged and compared. This is only meaningful if the accuracy calculations are based on the same data, especially since punctuation is often unambiguous regarding its POS tag, causing accuracy with punctuation to always be higher than that without punctuation. Hence, punctuation marks[19] are never counted for POS tagging accuracy.

4.2 Tagging on modern data

Normalization tries to handle the problem of spelling inconsistencies found in historical language data. However, this is not the only challenge for processing the data with POS taggers: another problem lies in the fact that there is often no consistent capitalization, or at least not in the way that it is used in modern German. As a consequence, POS taggers cannot use capitalization as a reliable clue to detect nouns. Normalization

[19] More precisely: tokens that are tagged as punctuation in the gold standard POS annotation.

already ignores capitalization and only operates on lowercased data; for the reasons stated above, the same will be done for POS tagging.

Another challenge is the missing or inconsistent use of punctuation marks. In contrast to modern German, full stops are rarely used in the Anselm corpus; the Melk manuscript, for example, mostly uses virgules (visually resembling a modern slash '/') for punctuation. Virgules are often placed where modern German would use a full stop (cf. Sec. 2.3.2), however, this is far from a definite rule. Furthermore, large parts of the Anselm texts feature no punctuation marks at all.

As both capitalization and punctuation can provide clues for POS tagging, the lack of these features can potentially have a negative impact on tagging performance. To study the magnitude of this effect, tagging experiments are performed on modified versions of the modern TIGER/Tüba corpus (cf. Sec. 2.2). Three different scenarios are considered: (1) all data is lowercased; (2) all punctuation marks and sentence boundaries are removed; and (3) the combination of both.

Sentence boundaries in particular are usually taken into account by the POS tagger in some way. TreeTagger explicitly requires the identification of one POS tag as marking sentence boundaries, while RFTagger expects an empty line between sentences. MBT requires a similar, configurable marker. For the purpose of evaluating without sentence boundaries, all punctuation marks are stripped from the corpus, and all boundary markers are removed. TreeTagger is given a non-existent tag to be used as the boundary POS tag.

Evaluation is done as tenfold cross-validation for each combination of testing scenario and tagger. The tenfold partition of the TIGER/Tüba corpus was created randomly using sentences as the basis for splitting; the same partition is used for all evaluation scenarios. While no noticeable performance loss in terms of tagging speed was observed with RFTagger and TreeTagger for any scenario, MBT took considerably longer when training on data without sentence boundaries, averaging about 7 hours of computing time for processing just one fold of training and evaluation data.[20] Therefore, an exception to the cross-validation approach is made for MBT without sentence boundaries, and only the first fold is processed in these cases.

The full evaluation results for these tagging experiments are presented in Table 18, given both for all (non-punctuation) tokens and for unknowns only, i.e., wordforms that were not seen during training. The results show that lowercasing all words reduces STTS accuracy by 0.35–0.42 percentage points for the RFTagger and TreeTagger; MBT takes a slightly larger performance hit of about 0.7 percentage points. The average size of the evaluation parts is slightly below 190,000 tokens—at that size, these differences can be found to be statistically significant ($p < 0.05$).

Removing all punctuation marks and sentence boundaries is an even bigger challenge for all taggers; accuracy is significantly lower than with the lowercased version. This is

[20]The system used for the tagging experiments was the same as for the normalization: Intel i7-870 @ 2.93 GHz, 16 GB RAM, Linux 3.6.4

		All			Unknowns		
		STTS	$STTS_{red}$	Univ	STTS	$STTS_{red}$	Univ
Original	RFT	96.59%	97.36%	97.95%	89.60%	91.22%	94.96%
	RFT_{10}	**96.85%**	**97.40%**	**97.99%**	**90.22%**	91.31%	95.05%
	TREE	95.91%	96.77%	97.41%	90.01%	**91.38%**	**95.43%**
	MBT	96.26%	97.18%	97.94%	84.36%	86.67%	92.80%
Lowercased	RFT	96.24%	97.00%	97.59%	89.47%	90.93%	94.53%
	RFT_{10}	**96.50%**	**97.04%**	**97.62%**	**90.21%**	**91.18%**	**94.72%**
	TREE	95.49%	96.32%	96.97%	88.11%	89.25%	93.44%
	MBT	95.57%	96.53%	97.24%	81.36%	83.16%	88.79%
No punct.	RFT	95.90%	96.87%	97.53%	88.51%	**90.38%**	94.41%
	RFT_{10}	**96.22%**	**96.97%**	**97.63%**	**88.89%**	**90.38%**	**94.42%**
	TREE	95.04%	96.18%	96.93%	88.62%	90.22%	94.36%
	MBT†	95.20%	96.46%	97.29%	82.77%	85.16%	91.67%
Lowercased + no punct.	RFT	95.39%	96.35%	97.01%	88.01%	89.59%	93.36%
	RFT_{10}	**95.74%**	**96.48%**	**97.14%**	**88.58%**	**89.78%**	**93.52%**
	TREE	94.44%	95.55%	96.33%	86.84%	87.94%	92.28%
	MBT†	94.45%	95.76%	96.54%	80.29%	82.39%	88.06%

Table 18: Tagging accuracy of different taggers on modern data, without capitalization (= Lowercased) and punctuation marks (= No punct.); scores are the average of tenfold cross-validation, except rows marked with '†', where only one training and evaluation step was performed; accuracy is always evaluated without punctuation marks; best result for each column and tagging scenario highlighted in bold.

not too surprising, though. While capitalization in German provides an important clue for the detection of (all types of) nouns, most other languages restrict capitalization to proper nouns and sentence-initial words, yet can still be POS tagged with high accuracy. Furthermore, wordforms of nouns are often not confusable with other parts of speech even when lowercased. Removal of punctuation marks, on the other hand, is much more likely to create ambiguity and/or raise the difficulty of disambiguation. As an example, the function word *das* can be used as an article, a demonstrative pronoun, and a relative pronoun—in the latter case, it typically follows a comma, which is a strong clue for the tagger which is lost after removing punctuation. Indeed, a statistical analysis reveals that the error rate for the category PRELS (substituting relative pronouns, such as *das*) doubles when removing punctuation. This is true for all taggers, though the

exact figures vary: error rate using RFT_{10} increases from 4% (with punctuation) to 8% (without punctuation), while with MBT, it rises from 11% to about 22%. In all cases, it was most often confused with an article (ART), suggesting that the ambiguous forms *der/die/das* indeed make up the majority of these cases.

Interestingly, when comparing accuracy using the Universal tagset, there is no significant difference between the lowercase version and the "no punctuation" version, while a slight difference can still be observed with the reduced STTS. This suggests that a significant part of errors when tagging without punctuation marks is made within the same category of the Universal tagset, but less often within one category of $STTS_{red}$. Error analysis provides a clue that pronouns are largely responsible for this: with RFT_{10}, 22% of all occurrences of PWAV and 11% of PWS, PDS, PRELAT, and PIS were tagged incorrectly. If a large number of these occurrences are confused with other pronouns, this could lead to the observed results. Again, confusion between *das* as a demonstrative and a relative pronoun is one possible example for this.

A further result of Table 18 is that using RFTagger with a context size of 10 (RFT_{10}) achieves the highest score in almost every scenario. Compared to the "standard" RFTagger using a context size of 2, the increase in performance also comes with a considerable increase in tagging time. The exact difference fluctuated strongly during the tests, but time required to tag the Berlin text typically ranged from a few seconds (with context size 2) to 5–8 minutes (with context size 10). Still, the numbers suggest that RFT_{10} not only has the highest overall accuracy for modern German with the given test and training corpus, but is also best for tagging data with the handicaps of no capitalization and/or punctuation marks.

A surprising result is the comparatively bad performance of MBT on unknown tokens, which is always about 6 percentage points below that of the other taggers. This is a remarkable difference, especially considering the high performance on unknowns reported for MBT with English corpora (Daelemans et al., 1996). It is conceivable that the results can be improved by tuning the various parameters of both the tagger and the memory-based learning framework utilized by it. Experiments with parametrization were not tried for this thesis, however, a study by Ivanova and Kübler (2008) found that tagging results on German using MBT were best when setting the context information to two tokens on both sides of the input word. The default setting used here differs from this optimum by using only one token to the right, which the study found to have only a marginal effect on the result, though. Another possible factor is the prefix length taken into account for unknown words, which is one by default, but three in the study by Ivanova and Kübler (2008). The differences in accuracy reported there by changing the tagger's settings are all comparatively minor, though, letting it appear questionable whether these settings can actually close the gap of about 6 percentage points to the other taggers.

Finally, combining both handicaps and evaluating on lowercased data without punctuation gives another decrease in accuracy. This is the most relevant result, as this configuration is used the most often during evaluation of Anselm and Sermon texts. As with all other scenarios, comparing STTS accuracy, RFT_{10} performs best, followed by standard RFTagger, MBT, and TreeTagger in that order. The difference between this scenario and the unmodified corpus is about 1.1 percentage points using RFT_{10}, which means that overall tagging accuracy is still very high even with both drawbacks combined (95.74%). On the other hand, this decrease in accuracy should not be underestimated, either, as it translates to an increase of the error rate by about 35%. Thus, it is possible that the negative effect of lowercasing words and removing punctuation is magnified when POS tagging data with generally lower accuracy scores. Still, for historical data without reliable capitalization or punctuation, there is no obviously better alternative to this approach.

4.3 Tagging on historical and gold standard data

Before comparing POS tagging accuracy on automatically normalized texts, tagging is performed both on the original, unmodified data and on the gold standard normalizations. The former can serve as a reference for the improvement of tagging accuracy with normalization, while the latter is likely to be the upper bound for such an approach. For the gold standard normalization, three different variants are considered: (1) including the original punctuation marks for tagging; (2) using modern punctuation; and (3) tagging without any punctuation at all. The first two variants were tested on a tagger model that has been trained on the TIGER/Tüba corpus with modern punctuation, while the TIGER/Tüba corpus without punctuation has been used to train the tagger for the last variant. Again, tagging accuracy is only evaluated on the non-punctuation tokens in each case for better comparability of the results.

"Modern punctuation" for the Sermon texts refers to the normalization of historical punctuation marks. This also means that historical and modern punctuation marks always coincide. For the Anselm texts, modern punctuation was annotated separately from the normalization and therefore does not have this limitation. While this is a conceptual difference, the Sermon texts might already be close to modern conventions with regard to punctuation due to the time at which they were written. Hence, they do not require such an elaborate approach. Supporting this argument is the fact that in JubelFeste and Gottesdienst, there is no difference at all in punctuation between the original and the normalization layer. Section 4.3.4 will present a counterexample to this theory, though.

Table 19 shows the results of this evaluation, reporting accuracy on the STTS tagset only. For tagging on the original, historical data, accuracy scores are almost always higher than the normalization baseline, the LeichSermon text being the only exception.

		Original	GS normalization		
			OrigP	NoP	ModP
Berlin (norm)	RFT	27.95%	86.06%	*87.16%*	**87.58%**
	RFT_{10}	28.65%	85.78%	*87.07%*	*87.29%*
	TREE	26.57%	85.29%	86.14%	*86.44%*
	MBT	27.65%	84.42%	84.89%	85.89%
Berlin (mod)	RFT	27.95%	86.80%	*88.09%*	*88.30%*
	RFT_{10}	28.65%	86.73%	**88.37%**	*88.20%*
	TREE	26.57%	86.12%	*87.07%*	*87.18%*
	MBT	27.65%	85.42%	85.93%	86.88%
Melk (norm)	RFT	44.75%	86.04%	*87.47%*	**88.00%**
	RFT_{10}	44.70%	85.21%	*87.74%*	*87.76%*
	TREE	41.45%	85.76%	*86.70%*	*87.14%*
	MBT	44.29%	84.70%	86.26%	*86.84%*
Melk (mod)	RFT	44.75%	87.38%	*89.21%*	*89.32%*
	RFT_{10}	44.70%	87.01%	**89.63%**	*89.32%*
	TREE	41.45%	87.23%	88.00%	*88.33%*
	MBT	44.29%	86.02%	87.60%	88.12%
LeichSermon	RFT	67.95%	*79.95%*	**81.31%**	*80.05%*
	RFT_{10}	67.95%	*81.22%*	*81.04%*	*80.59%*
	TREE	66.55%	*80.05%*	*80.23%*	*80.32%*
	MBT	66.77%	78.74%	*78.96%*	*79.19%*
JubelFeste	RFT	81.24%	*89.56%*	*89.94%*	*89.56%*
	RFT_{10}	82.26%	**90.41%**	*90.03%*	**90.41%**
	TREE	79.74%	*89.14%*	*88.39%*	*89.14%*
	MBT	80.35%	*88.96%*	*87.41%*	*88.96%*
Gottesdienst	RFT	86.84%	*92.58%*	*91.55%*	*92.58%*
	RFT_{10}	88.07%	**93.24%**	*92.27%*	**93.24%**
	TREE	84.90%	*92.01%*	89.86%	*92.01%*
	MBT	84.23%	90.17%	88.68%	90.17%

Table 19: Tagging accuracy of different taggers on the original historical data and gold standard normalizations (OrigP = original punctuation, NoP = no punctuation, ModP = modern punctuation), using STTS tags; accuracy is always evaluated without punctuation marks; best result for each text is highlighted in bold, differences to the best result that are not statistically significant ($p > 0.05$) are marked in italics.

As an example, the normalization layer of the Berlin text has a (normalization) baseline of 23.40%, yet can achieve a tagging accuracy of 28.65%. This is especially remarkable as that number refers to the STTS tagset; with the Universal tagset, the score goes up to 38.89%. Apparently, a certain amount of tokens can be tagged correctly even though they are not identical to their normalized counterparts.

(40) *an* *sotane* *wirdekeit*
an solche würde
APPR ADJA NN
'to such dignity'

(41) *von* *den* *ewigen* *tod*
von dem ewigen tod
APPR ART ADJA NN
'from eternal death'

Example (40) is from the Berlin manuscript, showing two extinct wordforms (*sotane wirdekeit*) along with their modernizations. Even though these words do no longer exist in modern German, they are correctly tagged by the RFTagger in the historical text. This is possibly the result of suffix analysis: *–keit* is a very common derivational suffix in modern German, resulting in a noun (NN), while *–tane* can be found in some inflected adjective (ADJA) wordforms (e.g., *spontane* 'spontaneous'). An even simpler case is Example (41): here, only the inflection of the definite article changes between the original and its modernization (*den/dem*). As this does not influence the base POS tag, the whole sequence can be tagged correctly. Examples like these demonstrate that perfect normalization accuracy is not always needed for correct POS tagging, which is also highly relevant for the tagging of automatically normalized data.

Tagging on the gold standard normalization shows slightly varying results. Using modern punctuation or no punctuation at all is always best[21]; however, as the texts are comparatively short, the difference between these two versions is almost never statistically significant ($p < 0.05$; Gottesdienst with TreeTagger is the only exception). The same cannot be said for tagging with the original punctuation, though, which is almost always worse[22], and often significantly so. However, this is to be expected, as punctuation marks in historical texts often have a different distribution as the modern punctuation marks on which the tagger was trained (e.g., a virgule '/' is used for modern comma or full stop, but rarely for a modern slash). In order to achieve better results with original punctuation marks, they would probably have to be converted to the respective modern tokens (e.g., replacing virgules with full stops). However, the correlation of historical and modern punctuation analyzed for Anselm in Section 2.3.2 showed that this is not a trivial thing to do. Also, the results of tagging with correct modern punctuation

[21] As there is no difference between original and modern punctuation in JubelFeste and Gottesdienst, the original punctuation is assumed to be already "modern" in these cases.

[22] In the only counterexample where original punctuation is better than the other options, LeichSermon with RFT_{10}, the difference is again not statistically significant.

are not significantly better than those without punctuation at all. Hence, it is doubtful whether this approach would lead to improved results.

In general, the results suggest that the best course of action is to remove all punctuation marks for tagging. One reason for this is that enriching a text with modern punctuation is another annotation step which has to be performed manually. As the accuracy does not improve significantly over tagging without punctuation, though, this is an effort which does not pay off (at least if it is done exclusively for the purposes of POS tagging). On the other hand, if information about modern punctuation is available, it does not hurt the tagging process either—depending on the characteristics of the text, it might be able to produce minimally better results. Historical punctuation marks should never be included for tagging as their distribution cannot be learned from modern data, which negatively affects tagging accuracy. Of course, newer texts such as JubelFeste and Gottesdienst constitute an exception to this rule: if punctuation in the original text can be assumed to be already close to modern, it is reasonable to retain it in the tagging process.

For all texts, the accuracy scores on gold standard normalizations are comparatively low. Evaluation on lowercased, modern data without punctuation still achieved an average accuracy of over 95%, while the gold standard normalizations only give around 88% for the Anselm texts, 90–93% for JubelFeste/Gottesdienst, and an exceptionally low 81% for LeichSermon. These lower accuracies might also be the reason for the fact that using RFTagger with a context size of 10 is sometimes worse here than using it with the default size of 2 (in contrast to the results in Table 18): with a larger context size, tagging errors will negatively influence more of the following wordforms. This is consistent with the observation that standard RFT is better with LeichSermon and the normalization layer of Berlin and Melk, which always have accuracies of at most 88%, whereas RFT_{10} performs better for the other texts that show slightly higher accuracies.

The remainder of this section will try to highlight some of the reasons for the comparatively low accuracy scores on gold standard normalizations. As they represent a kind of upper bound, POS tagging on automatically normalized data can be expected to face very similar problems, mainly augmented by potential normalization mistakes. Therefore, the focus of the qualitative analysis of tagging errors in this thesis is on the gold standard data.

4.3.1 Semantic and morphologic variation

An important observation is that even perfectly normalized historical data has different characteristics than modern data. This is because normalization as it is done for the Anselm and GerManC-GS corpora only affects the spelling of the wordforms. Additionally, the transcriptions of the Anselm texts already account for differences in modern word boundaries (e.g., one historical word that would be written as two words in modern German, and vice versa), which can be used for further processing. However,

one factor that is not considered in the normalization layer of Anselm is the change of semantics or syntactic function. A prominent example is the wordform *so*, which is an adverb in modern German (and sometimes also tagged as KOUS in TIGER/Tüba in the combination *so dass* 'so that'), but is frequently used as a relative pronoun in ENHG, as in Example (42) from LeichSermon.

(42) *die faelle / so aus schwacheit geschehen*
die fälle , so aus schwachheit geschehen
ART NN $, PRELS APPR NN VVPP
'the cases which occur out of weakness'

As the wordform *so* is never annotated as PRELS in the TIGER/Tüba corpus, the POS taggers have never seen this combination in the training data and will hardly, if ever, choose the tag for this wordform. GerManC-GS annotates these occurrences with a new tag PTKREL (cf. Sec. 2.4), however, this does not solve the issue of the annotation not being found in modern training data. Extinct wordforms pose a similar problem, as they cannot be learned from the training corpus either. In Example (43) from LeichSermon, the historical *dannenhero* has been normalized as *dannenher* 'therefore'. While this normalizes the spelling, it creates a wordform which does not exist in New High German either; the modern equivalent is *daher*. Similarly, in Example (44), the normalization layer of Anselm maps the wordform *czuhant* to the artificial lemma *zehant* 'immediately', which also no longer exists in NHG.

(43) *und dannenhero das guthe [...] nicht thun*
und dannenher das gute [...] nicht tun
KON ADV ART NN PTKNEG VVFIN
'and [they] therefore do not do the good (deeds)'

(44) *czuhant chust iudas mein chint*
zehant küsst judas mein kind
ADV VVFIN NE PPOSAT NN
'Immediately, Judas kissed my child'

In these cases, a POS tagger has to resort to its algorithms for dealing with unknown words, which may or may not be able to correctly identify the word class. For *dannenher*, this might be possible with suffix analysis. The most common words in TIGER/Tüba ending in *–her* are *bisher* 'so far' and *eher* 'rather', which are both adverbs (ADV). The modern *daher* also shares this suffix, but is actually tagged as PAV (pronominal adverb) rather than ADV, which the manual annotators chose in this case. For *zehant*, suffix analysis is unlikely to be successful. In TIGER/Tüba, the most common words ending in *–nt* are nouns (e.g., *Prozent* 'percent'), while those ending in *–ant* often are verb forms (e.g., *geplant* 'planned'). In fact, RFTagger correctly identifies *dannenher* as ADV, while *zehant* is mostly tagged as a noun or a predicative adjective (ADJD), the latter probably in analogy to the modern *interessant* 'interesting'.

(45) (a) *do* *quam* *eyn* *blynt* *ritter*
da kam ein blind ritter
ADV VVFIN ART ADJA NN
'There came a blind knight'

(b) NORM: ein blind ritter
RFT & MBT: ART ADJD NN
TREE: ART NE NE

Inflectional differences pose a similar problem for tagging. Adjectives, for example, are sometimes used in ENHG without an explicit inflectional ending, such as in Example (45) from the Berlin text. As the adjective form *blind* is missing an inflectional suffix, which is mandatory for the attributive use (ADJA) in front of a noun, RFTagger and MBT do not recognize it as such and tag it as a predicative adjective (ADJD) instead. TreeTagger instead treats the whole phrase *blind ritter* 'blind knight' as a proper noun, possibly because the predicative use of the adjective is very unlikely in the given syntactic context. Examples like these are one of the reasons why the Anselm corpus distinguishes two layers; in the modernization layer, the historical wordform *blynt* is also corrected for inflection. Still, in automatically normalized data, it is likely that such cases are still included and must be dealt with during POS tagging.

Besides inflectional changes, the modernization layer of the Anselm corpus also accounts for semantic change and extinct wordforms discussed above. In Example (44), *zehant* is replaced in the modernization layer by the existing modern word *sogleich*, which has a similar meaning and the same part-of-speech tag. However, tagging results are only slightly better on the modernization than on the normalization layer. The difference between the highest accuracy scores is in fact only significant ($p < 0.05$) for the Melk text, but not for the Berlin text. A possible explanation is that Berlin contains fewer extinct wordforms (1.1%) than Melk (1.8%), while the amount of wordforms with semantic changes is about the same. In any case, even the results for the modernization layer are still considerably worse than those achieved on modern data, showing that while the mentioned problems can be an issue for tagging accuracy, they are not the most important factor.

4.3.2 Syntactic variation

Another issue with normalized data is that syntactic peculiarities of the historical text are not in any way modified. Syntactic constructions that do not occur in modern data cannot be handled by the tagger, though.

(46) *lat* *die* *pey* *mir* *sint* *gen*
lasst die [, die] bei mir sind [,] gehen
VVIMP PDS [$, PRELS] APPR PPER VAFIN [$,] VVINF
'Let those who are with me go'

In Example (46), the original Melk manuscript only has one wordform *die* where modern German typically has two: the first as a demonstrative pronoun, and the second

as the relative pronoun introducing the clause which specifies the first pronoun. Alternatively, the example could be analysed as a free relative clause, where the antecedent *die* is not needed—however, at least with the given constituent ordering, this construction would be rather unusual in modern German[23] and is unlikely to be recognized by a tagger. In any case, commas are required to separate the relative clause from the main clause.

In the automatically tagged version, *die* is annotated as an article (ART) by all four taggers (while the rest of the phrase is annotated correctly). The POS taggers have no way to detect the elision of a wordform and recognize the ambiguity between PDS and PRELS, and therefore choose the most probable tag ART in this case. Interestingly, this tag is selected even though no adjective or noun—which is typically expected after an article—is following; this is true even for MBT, which explicitly takes right context into account.

(47) (a)

daz	*her*	*cruczegete*	*den*	*vorreter*	.	*der*	*sich*	*nennet*	*gotis*	*son*
dass	er	kreuzigte	den	verräter	,	der	sich	nennt	gottes	sohn
KOUS	PPER	VVFIN	ART	NN	$,	PRELS	PRF	VVFIN	NE	NN

'that he crucify the traitor who calls himself the son of God'

(b) dass er den verräter kreuzigte, der sich gottes sohn nennt

Unusual word order can be another problem for automatic tagging, but is often handled relatively well. Consider Example (47), which shows that the normalization retains the original word order. Here, contrary to modern German, the finite verb is not placed at the end of the subordinate clauses. Example (47b) shows the normalization corrected for modern ordering of the constituents. Even though the word order in (47a) would be considered ungrammatical and is unlikely to be covered by TIGER/Tüba, RFTagger gives the correct tags for this example. This can happen due to the fact that while the given word ordering is not grammatical in relative clauses, it resembles the order found in main clauses. A statistical *n*-gram model that does not know about syntactical structure might therefore still assign high probabilities to the given POS tag sequence. Additionally, lexicon entries play an important role for tagging, and the wordforms in the example are mostly unambiguous in this regard, increasing the likelyhood of a correct result.

(48) (a)

wo	*[...]*	*meine*	*seele*	*wird*	*fahren*	*hinn*
wo	[...]	meine	seele	wird	fahren	hin
PWAV		PPOSAT	NN	VAFIN	VVINF	PTKVZ

'where my soul will be going'

(b) wo [...] meine seele hinfahren wird

When there is ambiguity in addition to unusual word order, tagging performance can get worse. In Example (48), there is an inversion of the verb cluster; Example (48b)

[23] It is less unusual with the relative clause in front position: *Die bei mir sind, lasst gehen.* Furthermore, the part-of-speech tag for *die* should be PRELS instead of PDS in Ex. (46) when using this analysis.

shows the unmarked modern word order for comparison. Verb cluster inversion is still possible in modern German, but often marked, and primarily found in constructions with modal verbs. In addition to that, the verb particle (here: *hin*) is typically not affected by the inversion, and not split off from the verb in the infinitive form (*hinfahren*). While MBT still tags Example (48) correctly, RFTagger interprets the infinitive *fahren* 'to drive' as as being the derived noun (NN; 'the driving'). Note that this tagging error does not only stem from the unusual word order, but also from the lowercasing of all words—tagging with proper capitalization would have made this type of error very unlikely, as the tag NN would not have been seen in the training data with the lowercased form of the word.

4.3.3 Limits of the training corpus

Besides the differences between normalized text and NHG text discussed above, specific properties of the TIGER/Tüba corpus can be the cause for some of the tagging mistakes. The corpus, which is used to train all POS taggers, is made up of collections of newspaper texts. One property of this text type is that direct speech occurs only infrequently when compared to certain other genres, such as letters. The Anselm texts consist of question/answer sets, with Saint Anselm and the Virgin Mary regularly addressing each other in direct speech. Similarly, the Gottesdienst text is a religious speech which addresses its audience right from the very beginning and continues to do so for large parts of the speech. Additionally, newspaper text is usually written in a rather formal style. Both factors combine in a common tagging mistake found in Example (49) from the Berlin text:

(49)

NORM:	sieh	anselm	,	da	es	da	mittag	ward	[...]
GOLD:	VVIMP	NE	$,	KOUS	PPER	ADV	NN	VAFIN	
RFT:	NE	NE	$,	KOUS	PPER	ADV	ADV	VAFIN	

'Look, Anselm, as it was noon ...'

Here, at the beginning of the sentence, the Virgin Mary addresses Anselm directly. The phrase *sieh anselm* ('Look, Anselm') is used 19 times in the Berlin manuscript; the imperative *sieh*, which is a shortened form of *siehe* common especially in colloquial and/or direct speech, is used 24 times. RFTagger wrongly identifies this phrase as a proper noun, though, reducing the overall accuracy through the same recurring error. All other taggers produce similar mistakes, too. A look at the TIGER/Tüba training data reveals the cause for this: the wordform *sieh* does not occur there at all; only the standard form *siehe* was learned 113 times. Suffix analysis is not likely to lead to the correct POS tag, either, as imperative forms in general are very uncommon in TIGER/Tüba: they only make up 397 tokens out of its 1.6 million total size (0.02%), disregarding punctuation. In comparison, the gold standard POS annotation of the Berlin text already contains 43 imperative verb forms (0.91%).

	Unknowns		Accuracy			
			RFT	RFT_{10}	TREE	MBT
TIGER/Tüba	8,880	(5.54%)	*88.01%*	**88.58%**	86.84%	80.29%
Berlin (norm)	272	(4.98%)	**54.78%**	**54.78%**	44.85%	37.50%
Berlin (mod)	222	(4.70%)	*59.91%*	**61.71%**	49.55%	41.44%
Melk (norm)	269	(5.91%)	*54.28%*	**57.62%**	*52.79%*	*51.30%*
Melk (mod)	218	(4.79%)	*65.60%*	**72.48%**	*65.14%*	61.93%
LeichSermon	195	(8.80%)	**54.36%**	**54.36%**	*50.26%*	43.08%
JubelFeste	134	(6.27%)	**82.84%**	*82.09%*	*76.87%*	*74.63%*
Gottesdienst	107	(5.48%)	*80.37%*	**85.98%**	*82.24%*	71.03%

Table 20: Tagging accuracy of different taggers on unknown tokens in the gold standard normalizations (without punctuation), using STTS tags; average of 10-fold CV with TIGER/Tüba data given for comparison; best result for each text is highlighted in bold, differences to the best result that are not statistically significant ($p > 0.05$) are marked in italics.

In the gold standard annotation, this problem could be easily rectified if all instances of the respective wordforms would be normalized as *siehe* by convention. This is not a satisfactory solution, though, if the normalization algorithms are still able to produce the shorter *sieh*. If this was disallowed (e.g., by removing the entry from the lexicon), the baseline of a historical text containing the wordform *sieh* would be artificially lowered, since the wordform would have to pass through the normalization algorithms even though it is already a valid modern form. This does not yet take into account the fact that this problem is certainly not restricted to a single wordform, but a more systematic issue resulting from the highly different text genres used for training (newspaper text) and evaluation (religious text, speeches, dialogues). Further issues raised by this genre difference can certainly be found.

Note that the second tagging error in Example (49), *mittag* 'noon' annotated by all taggers as ADV, results from missing capitalization again: *mittag* is only ambiguous between ADV and NN in the lowercased version of the TIGER/Tüba corpus.

Genre-specific differences can also manifest themselves in the vocabulary of the texts. POS taggers try to cover this problem "by design" through their integrated handling of unknown wordforms, i.e., wordforms that do not appear in the training corpus and could therefore not have been learned. For modern texts including punctuation, they typically achieve accuracies around 90% on unknown tokens (cf. Table 18). Table 20 shows that tagging unknowns is more difficult with the gold standard normalizations,

with some texts only achieving accuracies as low as 54%. These numbers, of course, do not only include changes in vocabulary, but are also affected by occurrences of extinct wordforms, which were already discussed in Section 4.3.1. Extinct wordforms also explain the large gap between the results for normalization vs. modernization of the Anselm data.

(50) *do* *wart* *dy* *prophetie* *an* *myr* *wor*
da ward die prophetie an mir wahr
ADV VAFIN ART NN APPR PPER ADJD
'Then the prophecy became true for me'

(51) (a) *der* *selb* *chnecht* *hies* *malichus*
der selbe knecht hiess malchus
ART PDAT NN VVFIN NE
'The same servant was called Malchus'

(b) NORM: der selbe knecht hiess malchus
RFT & RFT_{10}: PDS PDAT ADJD VVFIN NE
TREE & MBT: ART ADJA NN VVFIN NE

Example (50) shows the unknown word *prophetie* 'prophecy', which can be argued to be very specific to religious types of text. Still, all taggers correctly identify it as NN, likely through a combination of the preceding article and suffix analysis (e.g., from modern *Demokratie* 'democracy'). Example (51) appears to be more difficult; both *knecht* 'servant' and the name *malchus* are unknown words for the taggers. The name *malchus* is unanimously tagged as NE, which is likely triggered by the preceding verb that typically introduces a proper name (*er hieß* ... 'he was called ...'). The noun *knecht*, on the other hand, is tagged incorrectly as ADJD by the RFTagger. Note that the difficulty of tagging *der selbe* 'the same' arises from a problem of the transcription here, as this combination is written as one word (*derselbe*) in modern German, which should have been noted in the transcription. It is supposable that this factor contributes to the mis-tagging of the following noun.

(52) (a) *do* *vorbunden* *sy* *ym* *syne* *ougen* *vnd* *vorsmeten* *yn*
da verbanden sie ihm seine augen und verschmähten ihn
'Then they blindfolded and despised him'

(b) NORM: und verschmähten ihn
GOLD: KON VVFIN PPER
AUTO: KON ADJA PPER

The wordform *verschmähten* 'despised/spurned' in Example (52) is not unknown, but was learned only once from the TIGER/Tüba corpus. In this occurrence, it was used as an adjective (*dem gern verschmähten DFB* 'the gladly spurned DFB'). Consequently, all taggers choose the same tag (ADJA) for this occurrence in the Berlin text, despite it being clearly and unambiguously used as a finite verbform. While this wordform does not count as unknown, its low frequency in the training corpus is still responsible for this problem.

Finally, the unusually low accuracy of the LeichSermon text in Table 19 can be at least partly explained by the comparatively high ratio of unknowns (8.8%) and the low accuracy reported for them. In addition to that, there appear to be differences in tagging conventions: LeichSermon has an unusually high number of numerals followed by a dot (130 occurrences), which are typically used as ordinal numerals in modern German. As such, they are tagged as ADJA in the TIGER/Tüba data, as in Example (53). In LeichSermon, they mainly refer to bible verses (cf. Example (21)) and are tagged as CARD, which is shown in Example (54).

(53) am 30. Januar
APPRART ADJA NN
'on January 30'

(54) jesaja 66. 13. joel 2. 14.
NE CARD CARD NE CARD CARD
'Isaiah 66,13, Joel 2,14'

This discrepancy is responsible for a large part of tagging errors in the LeichSermon text. Possible solutions are to alter the normalization of bible verse numbers to exclude the dot, or to add a postprocessing step for POS tagging which corrects these cases. In any case, this is another example of lowered accuracy due to significant differences between the training and the test corpora, which should always be accounted for when processing any kind of non-standard data.

4.3.4 Punctuation

While the data in Table 19 shows that adding modern punctuation does not significantly increase tagging accuracy, the discussion has not yet touched upon the possible reasons for this. After all, the tagging experiments on modern data in Section 4.2 gave the opposite result: removal of punctuation led to significantly worse accuracy scores. A possible explanation is that the difference is only relatively minor, and the evaluated texts are simply too short for a significant difference to be detected. Additionally, annotation errors in the Anselm data can also be responsible for this, as the annotation of punctuation marks was carried out by a single annotator only without any further verification.

For the Sermon texts, there is a 1:1 relation between historical and modern punctuation marks, which can turn out to be problematic. Example (55) shows an excerpt from the Gottesdienst text which contains a comma at a position where none is expected in modern German. The original text and the normalization are identical for this passage.

(55) stellen wir uns die gottheit , als ein wesen vor
VVFIN PPER PRF ART NN $, KOKOM ART NN PTKVZ
'Let us imagine the deity as a being . . . '

(56) und nichts ist schwerer , als ihnen zu widerstehen
KON PIS VAFIN ADJD $, KOUS PPER PTKZU VVINF
'And nothing is harder than to resist them'

The KOKOM tag is reserved for particles used for a comparison within a clause; in the above example, 'the deity' is compared with 'a being', which is specified more closely in the clause that is following. The comparison particle *als* can also be used to introduce a subordinate clause, though, as in Example (56). As a subordinate clause is introduced with a comma in modern German, but a comparison within a clause is not, both instances of *als* are tagged as a subordinating conjunction (KOUS) by all taggers. While Gottesdienst still performs better with punctuation than without it, similar examples can be found for the LeichSermon text, too. This is a conceptual problem of the normalization layer in GerManC-GS and can only be circumvented completely by ignoring the punctuation marks for POS tagging.

4.4 Tagging on automatically normalized data

In this section, automatic normalization and POS tagging are finally evaluated in combination. For the comparison of tagging accuracies, all normalization methods described in Section 3.1 are considered. Additionally, the chain combination of wordlist mapper, rule-based method, and weighted Levenshtein distance with trigrams (WLD_{tri}) is included, as it was shown in Section 3.3 to be the best-performing combination. For the Anselm texts, only the results for the normalization layer are provided. This is because the evaluation on gold standard data (cf. Sec. 4.3) showed only a minor advantage of using modernization versus normalization, while the modernization layer is much harder to generate with automatic normalization methods (cf. Sec. 3.2.1). Regarding the issue of punctuation, all POS tagging evaluation is done on texts without any punctuation marks. Section 4.3 showed that using modern punctuation does not always improve the results over using no punctuation at all; furthermore, the concept of "modern" punctuation has been shown to be different between the Anselm corpus and the Sermon texts, making a direct comparison more difficult. Finally, using no punctuation for tagging also lends more focus to the "automatic" aspect of the process, as modern punctuation requires yet another manual annotation step.

Table 21 shows the POS tagging accuracy for all texts, using the automatical normalizations generated by using the first 500 tokens for training, and evaluating on the full STTS tagset. The normalization accuracy for each text and method is always shown for comparison. Note that these figures are higher than the accuracies reported in Tables 5 and 15 as they include the 500 training tokens that were manually normalized (and are therefore always correct), while the normalization accuracies given in Section 3 did not.

Using the chain normalization technique always results in the best POS tagging accuracy, except for JubelFeste, where it is slightly but insignificantly worse than using

		Mapper	Rules	Leven	WLD$_{uni}$	WLD$_{tri}$	Chain
Berlin (norm)	NORM	66.84%	67.47%	37.70%	56.50%	65.01%	77.77%
	RFT	68.62%	67.64%	40.07%	55.73%	63.19%	**76.27%**
	RFT$_{10}$	69.04%	67.64%	39.80%	55.73%	63.25%	*75.95%*
	TREE	65.88%	66.20%	40.43%	55.22%	63.45%	*75.50%*
	MBT	67.49%	67.47%	39.65%	55.75%	62.83%	*75.46%*
Melk (norm)	NORM	67.25%	68.11%	54.46%	69.10%	72.59%	77.14%
	RFT	69.19%	69.14%	56.29%	67.05%	69.91%	*76.24%*
	RFT$_{10}$	69.65%	69.32%	56.00%	67.80%	70.37%	*76.24%*
	TREE	67.08%	66.59%	56.99%	67.58%	70.29%	*75.87%*
	MBT	69.16%	68.88%	57.03%	67.91%	70.37%	**76.44%**
LeichSermon	NORM	81.81%	83.43%	70.34%	80.36%	80.95%	83.25%
	RFT	73.14%	*73.68%*	69.53%	*75.08%*	*75.40%*	**75.89%**
	RFT$_{10}$	*73.72%*	*74.18%*	69.84%	*75.08%*	*75.17%*	*75.85%*
	TREE	72.73%	*73.36%*	68.31%	*73.59%*	*73.59%*	*74.67%*
	MBT	71.69%	72.33%	68.44%	*74.04%*	*74.40%*	*74.45%*
JubelFeste	NORM	88.68%	91.20%	83.81%	89.33%	90.92%	93.73%
	RFT	*85.73%*	*86.38%*	80.58%	83.39%	*85.21%*	*86.57%*
	RFT$_{10}$	*85.87%*	**86.66%**	81.00%	83.72%	*85.17%*	*86.52%*
	TREE	82.50%	83.72%	79.36%	82.36%	83.67%	*84.79%*
	MBT	82.69%	84.09%	79.83%	81.66%	83.01%	84.00%
Gottesdienst	NORM	90.02%	92.93%	88.07%	94.83%	96.62%	96.83%
	RFT	89.25%	*90.68%*	85.15%	88.94%	*90.37%*	*90.68%*
	RFT$_{10}$	*90.12%*	**91.30%**	84.79%	*89.71%*	*90.99%*	**91.30%**
	TREE	86.64%	88.12%	83.67%	87.35%	88.58%	88.84%
	MBT	85.82%	86.89%	82.74%	86.79%	87.97%	88.02%

Table 21: Tagging accuracy on automatically normalized texts using the first 500 tokens as training data (cf. Table 5); normalization accuracy per text and method is given for comparison in lines "NORM"; "Chain" refers to the chain combination of Mapper → Rules → WLD$_{tri}$. Tagging was performed without punctuation marks; accuracy is evaluated on STTS tags. Best result for each text is highlighted in bold, differences to the best result that are not statistically significant ($p > 0.05$) are marked in italics.

	Original	GS	Auto	
Berlin (norm)	28.65%	87.16%	76.27%	Chain
Melk (norm)	44.75%	87.74%	76.44%	Chain
LeichSermon	67.95%	81.31%	75.89%	Chain
JubelFeste	82.26%	90.03%	86.66%	Rules
Gottesdienst	88.07%	92.87%	91.30%	Chain

Table 22: Comparison of POS tagging accuracy between the original historical data, gold standard normalizations (GS), and the best automatically normalized version using the first 500 tokens as training data (method given in last column); numbers always represent the best result from all POS taggers.

the rule-based method alone. The JubelFeste text aside, this is not a surprising result, as it shows that using the best normalization technique generally results in the best POS tagging performance. Similarly, the results confirm the tendency from the previous sections that the RFTagger usually performs best on the given data. With the Melk text, MBT sometimes achieves a slightly better accuracy, but as the difference is too small to be statistically significant, this cannot be argued to conflict with the general trend in favor of RFTagger. Finally, tagging accuracy is usually below normalization accuracy. This is to be expected, though, as even the gold standard normalizations (which can be seen as having a normalization "accuracy" of 100%) do not achieve a perfect (or near perfect) POS tagging score. This effect is more pronounced with the Sermon texts, while the Anselm texts achieve a better tagging result compared to the normalization.

Table 22 gives an overview of the best POS tagging results for the automatically normalized texts in comparison to the unmodified data and the gold standard normalization (without punctuation). The factor of the POS tagger is ignored here, and the best result from all four POS taggers is given to show the maximum accuracy that can be reached in each case. For the Anselm texts, the accuracy achieved by using automatically normalized data is significantly lower than using the gold standard, differing by more than 10 percentage points. However, the increase from the tagging baseline is still remarkable, especially considering that only 500 manually normalized tokens were used for training: from 28.65% to 76.27% for Berlin, and from 44.75% to 76.44% for the Melk text.

4.4.1 Correlation between normalization and tagging performance

In Table 21, looking at the connection between normalization and tagging accuracies for each of the evaluated normalization methods, there is a rough tendency for these two

accuracies to correlate. Standard Levenshtein distance, which always has the lowest normalization accuracy, also results in the worst tagging performance. On the other hand, the chain combination of normalizers, which produces the best normalization results, usually gives the best tagging accuracy, too. However, some exceptions to this tendency can be found; e.g., the WLD methods on their own result in slightly worse tagging accuracy than other methods when compared to their normalization score. As an example, while normalization with WLD_{uni} is slightly better than using the wordlist mapper for Melk, JubelFeste, and Gottesdienst, the texts normalized with wordlist mapping perform slightly better during POS tagging. LeichSermon suggests that this is not a definite rule, though, as it shows the opposite tendency. Still, especially for the newer texts, the rule-based method gives a similar POS tagging accuracy as the best chain method, despite being significantly worse at normalization: e.g., 92.93% for rule-based vs. 96.83% for the chain with Gottesdienst, but an equal POS tagging accuracy of 91.30%. This confirms the hypothesis that there are cases where better normalization accuracy does not equal better POS tagging, even though this is still the general tendency.

Section 4.3 already discussed many examples of what can go wrong during POS tagging even with perfect normalization. All issues mentioned there also apply to the tagging of automatically normalized text, of course, and can be responsible for the observed fluctuations in tagging accuracy. The issue of vocabulary appears to be particularly relevant, though. Example (57) shows that it is not necessarily unknown words which can cause problems, but also very infrequent ones.

(57) … *aller* *menschen-furcht* *und* *zaghafftigkeit* *vorbeugen*
… aller menschenfurcht und zaghaftigkeit vorbeugen
PIAT NN KON NN VVINF
'Prevent all of humanity's fear and timidity'

(58) (a) RULES: aller menschen-furcht und zaghaftigkeit vorbeugen
PIAT NN KON NN NN

(b) CHAIN: aller menschenwort und zaghaftigkeit verbeugen
PIAT NN KON NN VVINF

Both the rule-based method and the chain normalizer fail to normalize *menschen-furcht* (lit. 'humanity-fear') correctly; the former leaves it unchanged, while the latter produces the inappropriate *menschenwort* (lit. 'humanity-word'). As both versions are compound nouns with relatively common base nouns (*furcht* 'fear' and *wort* 'word'), though, they can be tagged correctly. The verb *vorbeugen* 'to prevent', on the other hand, is mistakenly tagged as a noun. Curiously, only the wrong normalization *verbeugen* 'to bow' is correctly tagged as VVINF. The problem here is a combination of vocabulary and training corpus size: the correct *vorbeugen* only appears once in TIGER/Tüba and was used there as a noun. Consequently, the taggers do not consider any POS tag for this word except NN. This example displays the same core problem already seen in

Example (52), and serves to illustrate that such cases are not uncommon at all in the evaluation data.

(59) *sie* *haben* *gejauchtzet*
sie haben gejauchzt
PPER VAFIN VVPP
'They rejoiced'

(60) (a) RULES: sie haben gejauchtzet
PPER VAFIN NN
(b) CHAIN: sie haben gejauchzt
PPER VAFIN ADJD

Vocabulary also plays a role in Example (59). Here, the wordform *gejauchtzet* 'rejoiced' needs to be normalized, which the rule-based method in Example (60a) fails to do. As a result, RFT_{10} incorrectly tags it as a noun (NN) instead of a verb particle (VVPP). The chain normalizer, on the other hand, normalizes it correctly, which still does not lead to the correct POS tag, though: it is wrongly tagged as an adjective form (ADJD) here (Example (60b)). From this passage alone, it is not clear where this error results from. One factor certainly is that the correct *gejauchzt* does not occur in the TIGER/Tüba corpus; however, there are many other VVPP wordforms ending in *–zt* (*verletzt, gesetzt, unterstützt*, etc.). As the RFTagger with a trigram model does not make this error, the large context size of RFT_{10} (and therefore some wordform before the passage shown in the example) is at least partially responsible for this. The same can be said for the excerpt in Example (61):

(61) *ihr* *seyd* *das* *auserwehlte* *geschlecht*
ihr seid das auserwählte geschlecht
PPER VAFIN ART ADJA NN
'You are the chosen lineage'

(62) (a) RULES: ihr seyd das auserwehlte geschlecht
PPOSAT NN ART ADJA NN
(b) CHAIN: ihr seid das auserwählte geschlecht
PPER VAFIN PDS NN NN

Just as in the previous example, the rule-based method does not normalize anything here, whereas the chain normalizer produces the perfectly correct normalization. However, the correctly normalized *auserwählte* 'chosen' is still mistakenly annotated as NN by RFT_{10}. The direct cause for this is likely the mis-tagging of the preceding article *das* as PDS, which reduces the probability of an adjective following. The cause of this mistake, in turn, can only be found outside the scope of this example, as the preceding wordforms were tagged correctly. In Example (62a), *ihr seyd* 'you are' is annotated wrongly, yet the following noun phrase is tagged correctly although it even contains an incorrect normalization. All in all, these examples show that correct normalization and correct POS tagging do not always coincide, and all combinations of wrong and correct normalizations and POS tags occur in practice.

4.4.2 Bridging the gap

Table 22 already showed that tagging accuracy on the best automatic normalization is still considerably lower than on the gold standard, which in turn stays behind the tagging results on modern data. Besides the qualitative analysis of tagging mistakes, this raises the question of what can be done to close this gap.

So far, the analysis of tagging errors has not considered the reduced versions of the STTS tagset introduced in Section 4.1. Table 23 gives the accuracies for the Berlin and the Gottesdienst text for all three tagsets when processed with RFTagger. In the Berlin text, there is a clear correlation between the different tagsets: the ranking of the different normalization methods is the same regardless which tagset is used for the comparison (with only one minor deviation for the Universal tagset on unknowns). This shows that the types of mistakes that are made during POS tagging are roughly the same for each version of normalized text; i.e., it is not the case that any particular normalization method is more likely to lead to errors within the same category of the $STTS_{red}$ or Universal tagset. Roughly the same tendency can be observed for Gottesdienst, too, with one exception: on unknown tokens, there are almost no differences between $STTS_{red}$ and Universal with some methods (Levenshtein, WLD_{tri}, the chain, and the gold standard), but differences of 4–5 percentage points for others (wordlist mapping, rule-based method, WLD_{uni}). This discrepancy mainly arises from auxiliary and modal verbs that contain umlauts, which are not normalized correctly by some methods and then tagged as full verbs (e.g., *muessen/VVFIN* vs. *müssen/VMFIN*). These verb types have different categories in $STTS_{red}$ (VV vs. VM), but not in Universal (VERB). In any case, with the low number of unknowns in Gottesdienst, the difference between these results is not significant.

In general, tagging accuracy on the Universal tagset is higher by about 5–8 percentage points compared to standard STTS. This is not a big difference compared to the relative accuracy, though, showing that the majority of tagging mistakes is made across different categories of the Universal tagset. This comes as no surprise when considering the examples of tagging errors discussed so far: Examples (42), (44), (46), (48–49), (51–54), and (57–62) all feature POS tags that also count as errors in the Universal tagset, e.g., NN vs. VVINF (in Universal: NOUN vs. VERB) or ADJD vs. VVPP (ADJ vs. VERB). In fact, only Example (45) represents a mistake made within the same category of ADJ.

On unknown tokens, the differences between tagsets are greater: up to 20 percentage points for the Berlin text. However, the overall accuracy is much lower at the same time, revealing another major problem of tagging normalized data. With Berlin, only 46.79% of unknown tokens resulting from the chain normalization can be POS tagged correctly. Compared to the accuracies of about 90% on modern data, this is a disappointing result. It also suggests that improving tagging accuracy on unknowns is an important aspect in improving the overall accuracy for these types of texts. Of course, this is at

	All			Unknowns			
	STTS	STTS$_{red}$	Univ	Count	STTS	STTS$_{red}$	Univ
Original	27.95%	29.79%	38.59%	2,915	15.09%	16.91%	29.23%
Gold	87.16%	89.40%	91.88%	272	54.78%	60.29%	69.49%
Mapper	68.62%	70.74%	75.06%	1,299	33.56%	37.18%	47.81%
Rules	67.64%	70.06%	74.17%	989	37.01%	41.35%	51.67%
Leven	40.07%	42.55%	48.19%	946	22.62%	25.90%	33.93%
WLD$_{uni}$	55.73%	58.06%	62.60%	603	33.83%	37.31%	47.60%
WLD$_{tri}$	62.19%	65.78%	70.14%	566	37.99%	42.58%	56.71%
Chain	**76.27%**	**78.64%**	**81.39%**	436	**46.79%**	**50.92%**	**60.32%**

(a) Berlin (norm)

	All			Unknowns			
	STTS	STTS$_{red}$	Univ	Count	STTS	STTS$_{red}$	Univ
Original	86.84%	90.53%	93.39%	345	60.87%	70.14%	80.58%
Gold	91.55%	94.98%	95.70%	107	80.37%	86.92%	86.92%
Mapper	89.25%	93.19%	94.57%	249	67.87%	77.51%	82.73%
Rules	**90.68%**	**94.27%**	**95.34%**	211	74.88%	**81.99%**	**85.78%**
Leven	85.15%	88.74%	90.63%	157	66.88%	74.52%	75.80%
WLD$_{uni}$	88.94%	92.47%	93.75%	132	67.42%	73.48%	78.79%
WLD$_{tri}$	90.37%	93.70%	94.62%	119	73.95%	78.99%	78.99%
Chain	**90.68%**	94.01%	94.88%	113	**76.99%**	81.42%	81.42%

(b) Gottesdienst

Table 23: Comparison of tagging accuracy using RFTagger, with different tagsets, evaluated separately for all tokens and for unknowns; "Count" gives the absolute number of unknown tokens for the respective text, "Chain" refers to the chain combination of Mapper $\rightarrow$ Rules $\rightarrow$ WLD$_{tri}$. Tagging was performed without punctuation marks; best result for each column (disregarding the gold standard normalization) is highlighted in bold.

least partly equivalent to improving the accuracy of the normalization, as many words that are unknown to the tagger result from errors made by the normalizer. Apart from unknowns, the problems of context-free normalization can also have a big effect on tagging results: e.g., in the WLD_{tri} normalization of the Melk text, 11.9% of all tagging mistakes result from the three highly ambiguous wordforms *in*, *im*, and *daz*, which can only be properly normalized using context information.

Still, there are some aspects of the POS tagging step which could also be improved in this regard. A part of the problem is that POS taggers for modern language data sometimes employ strategies which are not well-suited for processing normalized historical texts. Tagging of unknowns is a good example here. The default settings of MBT, for example, are based on the idea that "unknown words will behave similarly to infrequent known words" (Daelemans et al., 2010, p. 9). This is an assumption which does not hold for normalized historical data, though, as unknowns which stem from incorrect normalizations are just as likely to represent highly frequent words. In particular, this can also include function words, which are normally considered a closed class. For this reason, it is not a good idea to use a set of "open class tags", either, which restricts the POS tags that can be assigned to unknown words—a feature which is offered, for example, by the RFTagger.

However, previous examples—such as Examples (52) and (58)—showed that it is not only unknown words which can cause problems, but also words that are infrequent in the training corpus for the tagger. These words might have only been seen with one of several possible POS tags; however, as taggers typically use a derived lexicon to restrict the possible POS tags for known words, the other alternatives can never be generated. Consider again Example (52b), repeated here for convenience:

(52b)	NORM:	und	verschmähten	ihn
	GOLD:	KON	VVFIN	PPER
	AUTO:	KON	ADJA	PPER

The mis-tagging of *verschmähten* as ADJA is based upon a single occurrence of that wordform in the TIGER/Tüba corpus. In addition to that, the tagging sequence "KON ADJA PPER" is equally unlikely, also occuring only once in TIGER/Tüba. The combination "ADJA PPER" only occurs 27 times; in contrast, there are 9,216 instances of "VVFIN PPER", and 244 instances of the exact tagging sequence "KON VVFIN PPER". These numbers suggest that a POS tagger for historical data should ideally put less emphasis on the lexicon, but rather give more weight to the transition probabilities of POS tags. As an alternative, enriching the lexicon of the tagger with additional entries to compensate for the limits of the training corpus could also help to alleviate this problem.

Finally, normalization was shown to benefit from the combination of several normalization algorithms. It is conceivable to use a similar approach for POS tagging, too, by comparing the suggestions of different POS taggers and choosing among them. Ideally,

such an approach could also learn the individual strengths of a POS tagger by analyzing automatically tagged gold standard data, thereby trying to recognize contexts where a particular tagger is more or less likely to make mistakes. It remains to be tested whether such a combination would result in a comparable improvement of tagging accuracy.

5 Related work

There are many different approaches to normalization and annotation of historical language data. Giving a detailed comparison between some of these approaches was one of the goals of this paper, however, it is still only scratching the surface. In this section, I will try to give an overview of related work in this area.

Some research on dealing with historical spelling variation has been done in the field of information retrieval (IR). Here, the task is to find historical spellings that are related to a given modern wordform. For this purpose, Ernst-Gerlach and Fuhr (2006) derive rewrite rules from word pairs generated using a spellchecker, while Hauser and Schulz (2007) learn n-gram mapping rules from a dictionary. Although these techniques are similar to, e.g., the rule-based approach (Sec. 3.1.2), the nature of the task is quite different. For information retrieval, the focus is on recall, as there are typically many different spelling variants for a given modern wordform. For automatic annotation, precision is most important, as only one normalization candidate can ultimately be chosen for a given historical word. Therefore, mistakes are much more critical for the annotation scenario than for IR.

For automatic spelling normalization, some variant of weighted Levenshtein distance is often used. Strunk (2003) uses manually defined weights to generate wordform variants for Low Saxon (again in an IR context). Adesam et al. (2012) derive weighted substitution rules for Old Swedish; their approach is most similar to the WLD_{tri} method described here. However, they only report a pilot experiment on 249 tokens. For a comparison of several distance measures, see Kempken (2005), who also describes FlexMetric, a flexible distance measure similar to WLD_{uni}. A slightly different approach is used by Porta et al. (2013) for Old Spanish, who use edit transducers to model context-aware rules in a similar way to the rule-based approach of Sec. 3.1.2.

VARD 2 (Baron and Rayson, 2008) is a software tool developed for Early Modern English which combines different normalization techniques: wordlist mapping, character replacement rules, phonetic similarity, and standard Levenshtein distance. It is notable for providing a graphical user interface which allows for a semi-automatic normalization approach: the software suggests several normalization candidates which the user can accept—either for all identical tokens or for a specific instance only—or override with another wordform. However, while the wordlist and character replacement rules can easily be customized, the phonetic matching algorithm is specific to English. While

Hendrickx and Marquilhas (2011) successfully adapted VARD 2 to Portuguese, a first experiment in Bollmann (2012) showed that it performed worse for German than the chain-based approach described in Sec. 3.3.

A more recent approach is the application of character-based statistical machine translation to historical data. For Slovene texts from the late 18th century, Scherrer and Erjavec (2013) report an accuracy of 72.4% (from a baseline of 15.4%) using supervised character-based SMT, and 48.9% using an unsupervised variant. Pettersson et al. (2013) perform similar experiments for Icelandic and Swedish, achieving accuracies of up to 92%. They also show that their technique can achieve notable results with sparse training data, e.g., 76.5% accuracy on Icelandic texts after using 1,000 tokens for training. Sánchez-Martínez et al. (2013) successfully use character-based SMT for historical Spanish, and also show that the character error rate decreases by half compared to the "naive" wordlist mapping approach (0.21% vs. 0.50%; the baseline error rate was 5.75%).

Almost all normalization methods discussed so far only operate on types, i.e., they do not take context information into account at all. This effectively puts an upper limit on the maximum accuracy they can achieve (see, e.g., Table 4). Jurish (2010) is a notable exception; he uses hidden Markov models to choose between different normalization candidates depending on token context. In principle, this approach can be used with any of the aforementioned normalization techniques, as it does not rely on any particular method to generate the candidate wordforms. However, to what extent other techniques could benefit from this (or a similar) model remains open to future research.

Fewer studies have examined the effect of POS tagging on normalized data. Dipper (2010) reports a tagging accuracy above 91% for data from Middle High German, but normalization was done manually. POS tagging on parts of the GerManC-GS corpus has been tried before with an average accuracy of 79.7% (Scheible et al., 2011b), but again only manual normalization was considered. For Early Modern English, Rayson et al. (2007) report an accuracy of 89–91% on gold standard normalizations and 85–89% on texts automatically normalized using the VARD tool. A similar experiment for Portuguese achieved 86.6% and 83.4% on gold standard and automatic normalizations, respectively (Hendrickx and Marquilhas, 2011). Pettersson et al. (2013) also perform tagging on their normalized data, achieving 56.6% for Icelandic and an F-score of 88.7% for verb identification in Swedish.

There are some approaches for tagging historical texts that do not use spelling-normalized data as their input. For 14th century Dutch, van Halteren and Rem (2013) choose to expand the lexicon of the POS tagger instead. Using a form of weighted Levenshtein distance, they automatically generate hypothetical spelling variants for all lexicon entries, expanding the lexicon from 50,000 entries to about 1.25 million. They achieve an accuracy of around 95% for both tagging and lemmatization. Sánchez-Marco

et al. (2010) use an open source library for Spanish and adapt tagger internals, e.g., an affixation module, to historical data.

Finally, there is also active research on spelling variation for other types of non-standard data, most notably internet and social media texts, e.g., from Twitter, Facebook, or chat room conversations (van Halteren and Oostdijk, 2012; Derczynski et al., 2013; Neunerdt et al., 2013, are only a few recent examples). Although some domain-specific issues exist, there is a considerable overlap with problems found in historical data, e.g., inconsistent orthography, differences in tokenization, or syntactic variation. Zhang et al. (2013) present an adaptive normalization approach which they evaluate on data from Twitter, SMS, and call-center logs. However, while their method could conceivably be applied to historical data as well, such experiments have not yet been performed. In general, research on spelling variation in historical vs. social media data is still largely disjoined.

6 Conclusion

This thesis presented several methods for spelling normalization of historical language data and evaluated them on different texts in Early New High German (Anselm texts) and Early Modern German (Sermon texts), originating from the 15th to the 18th century. Additionally, it discussed several aspects of part-of-speech tagging applied to normalizations of historical data, and compared the performance of different POS taggers in various scenarios.

For normalization, four different methods were presented: a simple wordlist mapping approach, which learns word-based substitutions; the rule-based approach by Bollmann et al. (2011) operating on a character level; a normalization algorithm based on Levenshtein distance; and the same algorithm using a weighted variant of Levenshtein distance (WLD). Evaluation of these methods on the Anselm and Sermon texts showed that WLD with unigram, bigram, and trigram weights (WLD_{tri}) performed best when only a small amount of training data was available; e.g., using 100 tokens for training resulted in an accuracy of 68.21% for one of the Anselm texts compared to a baseline of 39.54%. When more training data was available, the rule-based method generally performed better, though the exact threshold varied between the texts. A combination of normalization methods in the form of a chain which starts with one algorithm, but calls the next one in the chain if the previous algorithm failed to find a result, was shown to improve results considerably. For the text which contained the most spelling variations in this evaluation, it was able to improve the baseline of 23.05% to 75.07% after training on only 500 manually normalized tokens from the same text.

The qualitative evaluation of normalization performance revealed several possible ways to improve upon these results. First of all, most of the normalization methods

considered here require a lexicon of modern wordforms to work. Accuracy of normalization naturally decreases if the correct normalization is not found in the modern lexicon, which commonly happens with all types of proper nouns. It is therefore helpful to compile a list of proper nouns referenced within a text or otherwise mark them in the transcription, so that they can either be added to the modern lexicon or be excluded from the normalization process. On the other hand, lexicons containing too many irrelevant entries can also be detrimental to the normalization accuracy, as they lead to more "false positives", i.e., wordforms that are generated due to a high similarity to the historical input wordform without being a correct normalization. This often concerns obscure proper nouns and abbreviations, the latter because shorter words are especially susceptible to this effect. To remedy this problem, lexical frequency could be factored into the probability score of normalization candidates, reducing the likelihood of generating very rare wordforms.

With regard to the combinations of normalization methods, the best results were achieved by using the wordlist mapper first, followed by the rule-based method and WLD_{tri}. This way, the method operating on the largest units of input (i.e., whole wordforms) is used first, followed by further methods of decreasing granularity: the rule-based method always considers at least two characters at a time (and at least three for non-insertion rules), while WLD_{tri} can also modify single characters in isolation. This property of this chain combination could be explored further, e.g., by trying an even longer chain of normalizers operating on decreasing amounts of characters. Similarly, an *n*-gram based normalization technique (e.g., based on the WLD algorithm) could be tried that explicitly prefers larger *n*-gram substitutions over shorter ones. While it is not guaranteed to produce better results, the evaluation at least suggests that such an approach could be promising.

Finally, to overcome the maximum accuracy limit inherent to all of the evaluated methods, integration of token context into the normalization process is ultimately required. A method to achieve this has been suggested by Jurish (2010), which can theoretically be extended to use the normalization algorithms presented here.

Part-of-speech tagging of historical German texts was shown to be more challenging than POS tagging of modern data, even when using gold standard normalizations. Reasons include the lack of consistent capitalization and punctuation, as well as semantic and syntactic differences to New High German. Ignoring all punctuation marks for POS tagging was shown to be better than using the original punctuation, and not significantly worse than using manually annotated modern punctuation marks. For this evaluation, POS taggers were retrained on a corpus of modern German in which all punctuation marks and sentence boundary markings had been removed. Cross-validation on the modern corpus revealed that removing punctuation and capitalization (by lowercasing all words) reduces the average tagging accuracy from 96.85% to 95.74%.

Tagging accuracy on the historical text with the lowest baseline increased from 28.65% (using the original data) to 87.16% using the gold standard normalization; with an automatical normalization using the first 500 tokens as training data, a tagging accuracy of 76.27% could be achieved. RFTagger proved to be better in most evaluation scenarios than TreeTagger or MBT, although no experiments with different settings of the taggers have been performed.

Semantic and syntactic peculiarities of Early New High German (ENHG) remain difficult to handle, though. They cannot be learned from modern training corpora, and spelling normalization does not cover these issues, either. For a corpus of Old Spanish, this led Sánchez-Marco et al. (2010) to abandon the normalization approach and use a customized POS tagger instead. Indeed, directly manipulating the suffix handling, lexicon lookup, or syntactic coverage of a tagger could prove to be rewarding for historical data, as some assumptions made by POS taggers that hold true for modern data do not necessarily apply to automatically normalized historical texts. Handling of unknown or infrequent wordforms was shown to be especially problematic in this regard, often leading to incorrect POS tags even for correct normalizations. An ideal solution would probably combine a normalization step to cover frequent spelling variations with a modified POS tagger that is more sensitive to the problem of imperfect data, and/or has been customized to cover specific syntactic or morphological features of ENHG. Additionally, a combination of several POS taggers—similar to the combination of normalization algorithms that was successfully employed—is also conceivable.

In conclusion, using automatic normalization for part-of-speech tagging of historical data was shown to be a viable approach. If a human annotator manually normalizes 500 tokens of a text containing 2,000–5,000 tokens in total, automatic normalization followed by POS tagging achieves a tagging accuracy between 75% and 91% depending on the type and amount of spelling variation found in the text. While this is still considerably worse than tagging on modern data, it can certainly be useful to aid the annotation process of a historical corpus. Furthermore, several improvements both to the normalization and the tagging process are conceivable, making further increases to the accuracy scores appear realistic.

References

Yvonne Adesam, Malin Ahlberg, and Gerlof Bouma. *bokstaffua, bokstaffwa, bokstafwa, bokstaua, bokstawa...* Towards lexical link-up for a corpus of Old Swedish. In *Proceedings of the 11th Conference on Natural Language Processing (KONVENS 2012), LThist 2012 workshop*, pages 365–369, Vienna, Austria, 2012.

Alistair Baron and Paul Rayson. VARD 2: A tool for dealing with spelling variation in historical corpora. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*, 2008.

Alistair Baron, Paul Rayson, and Dawn Archer. Automatic standardization of spelling for historical text mining. In *Proceedings of Digital Humanities 2009*, Maryland, USA, 2009.

Werner Besch. Die Rolle Luthers für die deutsche Sprachgeschichte. In Werner Besch, Anne Betten, and Oskar Reichmann, editors, *Sprachgeschichte. Ein Handbuch zur Geschichte der deutschen Sprache und ihrer Erforschung*, pages 1713–1745. de Gruyter, Berlin, New York, 2nd edition, 2000.

Marcel 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)*, pages 3–12, Lisbon, Portugal, 2012.

Marcel Bollmann. POS tagging for historical texts with sparse training data. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability in Discourse*, pages 11–18, Sofia, Bulgaria, 2013.

Marcel Bollmann, Florian Petran, and Stefanie Dipper. Rule-Based Normalization of Historical Texts. In *Proceedings of the International Workshop on Language Technologies for Digital Humanities and Cultural Heritage*, pages 34–42, Hissar, Bulgaria, 2011.

Marcel Bollmann, Stefanie Dipper, Julia Krasselt, and Florian Petran. Manual and semi-automatic normalization of historical spelling – Case studies from Early New High German. In *Proceedings of the 11th Conference on Natural Language Processing (KONVENS 2012), LThist 2012 workshop*, pages 342–350, Vienna, Austria, 2012.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER treebank. In Erhard Hinrichs and Kiril Simov, editors, *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT 2002)*, Sozopol, Bulgaria, 2002.

Thorsten Brants. TnT — a statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP 2000)*, pages 224–231, Seattle, USA, 2000.

Kenneth W. Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

Walter Daelemans, Jakub Zavrel, Peter Berck, and Steven Gillis. MBT: A memory-based part of speech tagger-generator. In *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 14–27, Copenhagen, Denmark, 1996.

Walter Daelemans, Jakub Zavrel, Antal van den Bosch, and Ko van der Sloot. MBT: Memory-based tagger, version 3.2, reference guide, 2010. URL `http://ilk.uvt.nl/downloads/pub/papers/ilk.1004.pdf`.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the ACL Workshop on Recent Advances in Natural Language Processing (RANLP)*, pages 198–206, Hissar, Bulgaria, 2013.

Stefanie Dipper. POS-tagging of historical language data: First experiments. In *Proceedings of the 10th Conference on Natural Language Processing (KONVENS 2010)*, pages 117–121, Saarbrücken, Germany, 2010.

Stefanie Dipper and Bettina Schrader. Computing distance and relatedness for medieval text variants from German. In Angelika Storrer, Alexander Geyken, Alexander Siebert, and Kay-Michael Würzner, editors, *Text Resources and Lexical Knowledge. Selected Papers from the 9th Conference on Natural Language Processing (KONVENS-08)*, pages 39–51. Mouton de Gruyter, Berlin, 2008.

Tomaž Erjavec. The goo300k corpus of historical Slovene. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2257–2260, Istanbul,

Turkey, 2012.

Andrea Ernst-Gerlach and Norbert Fuhr. Generating search term variants for text collections with historic spellings. In *Proceedings of the 28th European Conference on Information Retrieval Research (ECIR 2006)*. Springer, 2006.

Andreas W. Hauser and Klaus U. Schulz. Unsupervised learning of edit distance weights for retrieving historical spelling variations. In *Proceedings of the First Workshop on Finite-State Techniques and Approximate Search (FSTAS 2007)*, pages 1–6, Borovets, Bulgaria, 2007.

Iris Hendrickx and Rita Marquilhas. From old texts to modern spellings: an experiment in automatic normalisation. *Journal for Language Technology and Computational Linguistics (JLCL)*, 26(2): 65–76, 2011.

Steliana Ivanova and Sandra Kübler. POS tagging for German: How important is the right context? In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 994–997, Marrakech, Morocco, 2008.

Bryan Jurish. More than words: using token context to improve canonicalization of historical German. *Journal for Language Technology and Computational Linguistics*, 25(1):23–39, 2010.

Sebastian Kempken. *Bewertung historischer und regionaler Schreibvarianten mit Hilfe von Abstandsmaßen*. Diploma thesis, Universität Duisburg-Essen, 2005.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Petar Mitankin, Stoyan Mihov, and Klaus U. Schulz. Deciding word neighborhood with universal neighborhood automata. *Theoretical Computer Science*, 412(22):2340–2355, 2011.

Mehryar Mohri and Richard Sproat. An efficient compiler for weighted rewrite rules. In *Proceedings of the 34th annual meeting of the Association for Computational Linguistics (ACL '96)*, pages 231–238, Santa Cruz, California, USA, 1996.

Mehryar Mohri, Fernando Pereira, and Michael Riley. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32, 2000.

Melanie Neunerdt, Bianka Trevisan, Michael Reyer, and Rudolf Mathar. Part-of-speech tagging for social media texts. In Iryna Gurevych, Chris Biemann, and Torsten Zesch, editors, *Language Processing and Knowledge in the Web*, Lecture Notes in Artificial Intelligence (LNAI), pages 139–150, Heidelberg, 2013. Springer.

Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. *ArXiv:1104.2086*, 2011.

Saša Petrović, Miles Osborne, and Victor Lavrenko. The Edinburgh Twitter Corpus. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pages 25–26, Los Angeles, California, USA, 2010.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. An SMT approach to automatic annotation of historical text. In *Proceedings of the NODALIDA Workshop on Computational Historical Linguistics*, Oslo, Norway, 2013.

Jordi Porta, José-Luis Sancho, and Javier Gómez. Edit transducers for spelling variation in Old Spanish. In *Proceedings of the NODALIDA Workshop on Computational Historical Linguistics*, Oslo, Norway, 2013.

Paul Rayson, Dawn Archer, Alistair Baron, Jonathan Culpeper, and Nicholas Smith. Tagging the bard: Evaluating the accuracy of a modern POS tagger on Early Modern English corpora. In *Proceedings of Corpus Linguistics 2007*, University of Birmingham, UK, 2007.

Martin Reynaert, Iris Hendrickx, and Rita Marquilhas. Historical spelling normalization. A comparison of two statistical methods: TICCL and VARD2. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)*, pages 87–98, Lisbon, Portugal, 2012.

Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern*

Recognition and Machine Intelligence, 20(5):522–532, 1998.

Cristina Sánchez-Marco, Gemma Boleda, Josep Maria Fontana, and Judith Domingo. Annotation and representation of a diachronic corpus of Spanish. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation*, pages 2713–2718, 2010.

Silke Scheible, Richard J. Whitt, Martin Durrell, and Paul Bennett. A gold standard corpus of Early Modern German. In *Proceedings of the ACL-HLT 2011 Linguistic Annotation Workshop (LAW V)*, pages 124–128, Portland, Oregon, USA, 2011a.

Silke Scheible, Richard J. Whitt, Martin Durrell, and Paul Bennett. Evaluating an 'off-the-shelf' POS-tagger on Early Modern German text. In *Proceedings of the ACL-HLT 2011 Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH 2011)*, pages 19–23, Portland, Oregon, USA, 2011b.

Yves Scherrer and Tomaž Erjavec. Modernizing historical Slovene words with character-based SMT. In *Proceedings of the 4th Biennial Workshop on Balto-Slavic Natural Language Processing*, Sofia, Bulgaria, 2013.

Anne Schiller. Deutsche Flexions- und Kompositionsmorphologie mit PC-KIMMO. In Roland Hausser, editor, *Proceedings of the First Morpholympics*, Tübingen, 1996. Niemeyer.

Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. Guidelines für das Tagging deutscher Textcorpora mit STTS, 1999.

Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, Great Britain, 1994.

Helmut Schmid. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*, Dublin, Ireland, 1995.

Helmut Schmid and Florian Laws. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING '08)*, Manchester, Great Britain, 2008.

Simone Schultz-Balluff and Stefanie Dipper. St. Anselmi Fragen an Maria – Schritte zu einer (digitalen) Erschließung, Auswertung und Edition der gesamten deutschsprachigen Überlieferung (14.–16. Jh.). In Anne Bohnenkamp-Renken, editor, *Medienwandel/Medienwechsel in der Editionswissenschaft*, number 35 in Beihefte zu editio, Berlin/Boston, 2013. de Gruyter.

Jan Strunk. Information retrieval for languages that lack a fixed orthography. Seminar Paper, Stanford University, 2003. `http://www.linguistics.ruhr-uni-bochum.de/~strunk/LSreport.pdf`.

Felipe Sánchez-Martínez, Isabel Martínez-Sempere, Xavier Ivars-Ribes, and Rafael C. Carrasco. An open diachronic corpus of historical Spanish: annotation criteria and automatic modernisation of spelling. *arXiv:1306.3692v1*, 2013. URL `http://arxiv.org/abs/1306.3692v1`.

Heike Telljohann, Erhard Hinrichs, and Sandra Kübler. The Tüba-D/Z Treebank: Annotating German with a Context-Free Backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2229–2235, Lisbon, Portugal, 2004.

Hans van Halteren and Nelleke Oostdijk. Towards identifying normal forms for various word form spellings on Twitter. *Computational Linguistics in the Netherlands*, 2:2–22, 2012.

Hans van Halteren and Margit Rem. Dealing with orthographic variation in a tagger-lemmatizer for fourteenth century dutch charters. *Language Resources and Evaluation*, 2013. doi: 10.1007/s10579-013-9236-1. URL `http://link.springer.com/article/10.1007/s10579-013-9236-1`.

Justyna Walkowska. Gathering and analysis of a corpus of Polish SMS dialogues. *Challenging Problems of Science. Computer Science. Recent Advanced in Intelligent Information Systems*, pages 145–157, 2009.

Martijn Wieling, Jelena Prokić, and John Nerbonne. Evaluating the pairwise string alignment of pronunciations. In *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education (LaTeCH – SHELT&R 2009)*,

pages 26–34, Athens, Greece, 2009.

Yorick Wilks and Mark Stevenson. Word sense disambiguation using optimised combinations of knowledge sources. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'98)*, volume 2, pages 1398–1402, Montreal, Canada, 1998.

Congle Zhang, Tyler Baldwin, Howard Ho, Benny Kimelfeld, and Yunyao Li. Adaptive parser-centric text normalization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1159–1168, Sofia, Bulgaria, 2013.