Rule-Based Normalization of Historical Texts

Marcel Bollmann **Florian Petran** **Stefanie Dipper**
Ruhr-University Bochum
bollmann,petran,dipper@linguistics.rub.de

Abstract

This paper deals with normalization of language data from Early New High German. We describe an unsupervised, rule-based approach which maps historical wordforms to modern wordforms. Rules are specified in the form of context-aware rewrite rules that apply to sequences of characters. They are derived from two aligned versions of the Luther bible and weighted according to their frequency. The evaluation shows that our approach (83%–91% exact matches) clearly outperforms the baseline (65%).

1 Introduction[1]

Historical language data differs from modern data in that there are no agreed-upon, standardized writing conventions. Instead, characters and symbols used by the writer of some manuscript in parts reflect impacts as different as spatial constraints or dialect influences. This often leads to inconsistent spellings, even within one text written up by one writer.

The goal of our research is an automatic mapping from wordforms from Early New High German (ENHG, 14th–16th centuries) to the corresponding modern wordforms from New High German (NHG). Enriching the data with modern wordform annotations facilitates further processing of the data, e.g. by POS taggers.

In this paper, we present a rule-based approach. Given an input wordform, (sequences of) characters are replaced by other characters according to rules that have been derived from two word-aligned corpora. The results show that our approach clearly outperforms the baseline. However, there is still room for some improvement.

The paper is organized as follows. Sec. 2 discusses related work; in Sec. 3, we introduce our data. Sec. 4 addresses the way we derive rewrite rules from the data, while Sec. 5 deals with the application of the rules to generate modern wordforms. Sec. 6 presents the evaluation, Sec. 7 the conclusion.

2 Related Work

Baron et al. (2009) present two tools for normalization of historical wordforms. VARD consists of a lexicon and user-defined replacement rules, and offers an interface to edit and correct automatically normalized wordforms. DICER[2] is a tool that derives weighted context-aware character edit rules from normalized texts. The algorithm that creates these rules is not described in their paper, though.

Jurish (2010) compares different methods to normalize German wordforms from 1780–1880. The methods include mappings based on phonetic representations and manually created rewrite rules. The highest F-score (99.4%) is achieved by an HMM (Hidden Markov Model) that selects one of the candidates proposed by the other methods.

Research on normalizing historical language data has also been done in the field of Information Retrieval, applied to historical documents. They address the task reverse to ours: mapping modern wordforms to historical (or dialectal) wordforms.

Ernst-Gerlach and Fuhr (2006) run a spellchecker on their data (19th century German) to detect wordforms that differ from NHG wordforms, and to generate normalized wordform candidates. Context-aware rules that rewrite character sequences are derived from the pairs. They achieve an F-score of 60%.

[1]We would like to thank the anonymous reviewers for helpful comments. The research reported here was financed by Deutsche Forschungsgemeinschaft (DFG), Grant DI 1558/4-1.

[2]http://corpora.lancs.ac.uk/dicer/

ENHG	AM	anfang	schuff	Gott	Himel	vnd	Erden	[. . .]	VND	Gott	schuff	den	Menschen	jm	zum	Bilde /
NHG	Am	Anfang	schuf	Gott	Himmel	und	Erde	[. . .]	Und	Gott	schuf	den	Menschen	ihm	zum	Bilde ,
EN	In_the	beginning	created	God	heavens	and	earth		and	God	created	the	man	him	in_the	image

ENHG	zum	Bilde	Gottes	schuff	er	jn	/	Vnd	schuff	sie	ein	Menlin	vnd	Frewlin .
NHG	zum	Bilde	Gottes	schuf	er	ihn	;	und	schuf	sie	ein	Männlein	und	Fräulein .
EN	in_the	image	of_God	created	he	him		and	created	them	a	male	and	female

Table 1: The original ENHG and the modernized NHG version of Genesis 1:1 and 1:27, along with an English interlinear translation.

Hauser and Schulz (2007) use a corpus from ENHG and a dictionary from NHG to derive replacement rules of (sequences of) characters. To this end, they first assign ENHG wordforms to NHG dictionary entries, using Levenshtein distance to pick suitable candidate entries; wordforms with ambiguous assignments are excluded. The rule frequencies are used as weights for the rule applications. For generating ENHG lemmas from NHG lemmas, they achieve F-Scores between 56.9% (without weights), 66.2% (with weights derived from lexicon mappings), and 71.2% (with perfect training pairs).

Strunk (2003) uses weighted Levenshtein distance to generate dialectal wordform variants for IR of Low Saxon texts. Weights are manually defined and encode phonetic similarity.

Our approach is similar to Ernst-Gerlach and Fuhr (2006) and Hauser and Schulz (2007) in that we derive replacement rules of character sequences from aligned pairs. The algorithms to learn rules differ, though. Ernst-Gerlach and Fuhr (2006) specify recursive definitions that take into account rewrite sequences and contexts of varying sizes; rules can refer to characters or to the underspecified classes 'vowel' and 'consonant'. Hauser and Schulz (2007) extract n-gram sequences of varying size from the aligned wordforms, and learn n-gram mapping rules. Furthermore, our approach differs from theirs in that our training pairs stem from aligned corpora.

The evaluations cannot be easily compared because it is not clear to what extent the language data base is comparable with regard to its variation. The data from Jurish (2010) contains 59.2% identical word pairs (types), the data from Ernst-Gerlach and Fuhr (2006) 94%. In our data, only 51% of the pair types align identical words.

Furthermore, in our task, a historical form is most often mapped to one modern equivalent form; in the reverse task, a modern form is mapped to multiple historical variants.

3 The Corpus

In our approach, replacement rules are derived from word-aligned parallel corpora. A source that provides a parallel corpus in many languages, including historical ones, is the bible.

The collected works of Martin Luther are freely available from the web.[3] They include several versions of his bible translation, modernized to varying degrees. We chose the original ENHG version of the 1545 bible (which has been enriched with modern punctuation) as well as a revised NHG version of it, which uses modern spelling and replaces extinct words by modern ones.[4]

Table 1 shows text fragments in both versions. Differences concern capitalization (*AM – Am, anfang – Anfang*), character reduplication (*schuff – schuf, Himel – Himmel*), deletion (*Erden – Erde*), insertion, or replacement (*Frewlin – Fräulein*).

Compared to other texts from that time, the language of Luther's 1545 bible is rather close to NHG, since the evolution of NHG was heavily influenced by Luther's bible translation (Besch, 2000). Furthermore, printed texts in general show more consistent spelling than manuscripts, and use abbreviations to a lesser extent than manuscripts (Wolf, 2000); no abbreviations occur in Luther's 1545 bible. Hence, we expect that our approach can be transferred and applied to other printed texts more easily than to manuscripts.

Alignment The files contain one bible verse per line. A verse usually corresponds to one sentence; some verses, however, consist of more than one sentence. Sometimes the assignment of sentences or phrases to verses had been changed from the original to the modernized version to an assignment that is presumably closer to the original

[3]For instance: `http://www.sermon-online.de`.

[4]The original 1545 version is incorrectly called "Altdeutsch" ('Old German') in the archive. Our NHG text (which is possibly the 1892 revision) is a rather conservative version, while the 1912 and 1984 revised versions also contain corrections of mistranslations by Luther and, hence, deviate from the original ENHG text to a greater extent.

Greek version. The verses were rectified manually so that each version had the same number of verses. A few OCR mistakes were also manually corrected (e.g. *3ott* was replaced by *Gott* 'god'). The entire corpus was then tokenized using the tokenizer script supplied with the Europarl corpus (Koehn, 2005), and afterwards downcased.

Since there were still asymmetries in the assignment of phrases and sentences to the verses between the versions, the resulting corpus was not yet properly aligned. We applied a sentence aligner to our data, the Gargantua toolkit (Braune and Fraser, 2010). Next, the words of each aligned verse pair were aligned using the GIZA++ toolkit (Och and Ney, 2003).

The modernization often involves transforming words into phrases and vice versa, such as *soltu – sollst du* 'should you', or *on gefehr – ohngefähr* 'approximately' (literally: 'without danger (of saying so)'). Hence, it is crucial that the word aligner can handle 1:n/n:1 alignments. Upon manual inspection, we found a lot of misalignments with rarer tokens, many of which involve numbers, such as *zweiundzwanzig* 'twenty-two', which would actually correspond to the multi-word token *zwey vnd zwenzig* in the original 1545 version. These were probably misaligned because their frequency in the corpus is not high enough to train a translation model.

To minimize noise in our system's input, alignment pairs with a length difference of more than five characters were excluded from further processing. Since the two texts are highly similar—around 65% of the pairs align identical wordforms—a length difference of that magnitude rarely leads to meaningful alignments.

Some corpus statistics To assess the quality of the resulting word pairs, we had a small sample of 1,000 pairs of aligned non-identical wordforms from the evaluation corpus manually inspected by a student assistant. We identified six types of alignments, listed in Table 2.[5] Instances that were difficult to classify are assigned to a special class.

For deriving replacement rules, type 1 alignments are the perfect input. Pairs of type 2 and 3 are still useful, to a certain extent: correct rules can be derived from the word roots; mapping of differing inflection and affixes, however, should probably not be learned. Type 4 and 5 pairs (which occur very rarely) could be used to derive mappings of entire words rather than character sequences. Type 6 alignments clearly constitute noise.

Type	**Example**	**Freq**	**Eval**
1. Unproblematic	*vnd – und* 'and'	609	77%
2. Differing inflection	*truncken – trunkenen* 'drunken'	261	18%
3. Differing affixes	*oben – obenan* 'on top'	40	5%
4. Closer modern form exists	*noch – weder* 'neither/nor'	0	–
5. Extinct form	*stündlin - an dem Tage* 'on that day'	1	–
6. Incorrect	*zwey* 'two' – *für* 'for'	25	0%
7. Unclear cases	*fur* 'for' (?) – *für* 'for'	64	19%

Table 2: Alignment types: types 1–5 are correct to various degrees, type 6 is incorrect. For each type, its frequency in the sample and evaluation results for the more frequent types are given (see Sec. 6).

We further computed the number of target types a source word maps to. The pie chart in Fig. 1 shows that the vast majority of source types map to only one target type, with a significant minority mapping to two forms. The proportion of source types mapping to 4–8 target types was so negligible that they were merged in the pie chart. The quantity of source *tokens* that map to more than one target type on the other hand is pretty large. Even though the majority still has 1–2 mappings, about 20–30% of source tokens map to more than 5 target types, as the central bar plot of Fig. 1 shows. However, if we exclude targets that occur only five times or less, the graph shows a more balanced picture (right bar plot). Since the application of rules is based on their frequencies, it is highly likely that the impact of the infrequent rules is balanced out by the dominant ones.

Procedure The resulting corpus consists of 550,000 aligned pairs of words or phrases. We randomly picked 20% of the alignment pairs for a development corpus, which was used for the development of the rule extraction and application processes described below. Another 40% were afterwards used to extract the replacement rules for the following experiments (= training corpus), and a final 20% were picked for an evaluation corpus.

4 Normalization Rules

We used a modified algorithm for Levenshtein distance which not only calculates the numerical edit

[5]Throughout the paper, the examples are taken from the development corpus while the figures are calculated based on the evaluation corpus.

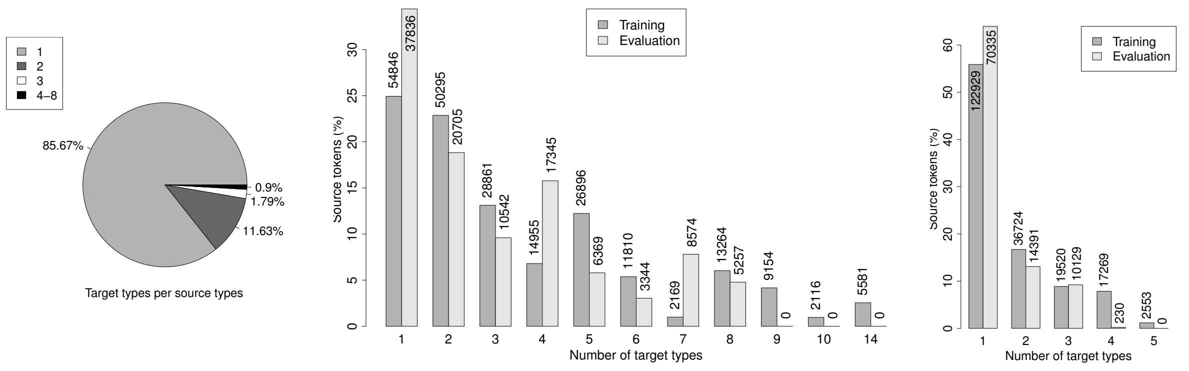

Figure 1: Target types per source type/token. The pie chart shows the result for source types (evaluation corpus), the central bar plot shows the results for all source tokens, the right bar plot only considers tokens with a frequency > 5 (training and evaluation corpus). The columns are labeled by absolute frequencies.

distance, but also outputs the respective edit operations (substitution, insertion, and deletion of single characters) that map the strings to each other. The record of edit operations was enriched with information about the immediate context of the edited character. Ex. (1) shows two sample edit operations, using the notation of phonological rewrite rules.

(1) a. $\varepsilon \rightarrow$ h / j _ r
(‘h’ is inserted between ‘j’ and ‘r’)

b. v $\rightarrow$ u / # _ n
(‘v’ is replaced by ‘u’ between the left word boundary (‘#’) and ‘n’)

Determining the context for these edit operations is not straightforward, because applying one rule can change the context for other rules. Since the Levenshtein implementation applies the rules from left to right, we decided to use the (new) target word for the left context and the (unaltered) source word for the right context (see also Fig. 2).

Identity rules In addition to canonical replacement rules, our rule induction algorithm also produces identity rules, i.e. rewrite rules that map a character to itself. Identity rules reflect the fact that the majority of words remain unaltered when mapped to their modernized forms, and many words are modified by few characters only. Identity rules and actual rewrite rules are intended to compete with each other during the process of rewriting.

Multiple optimal paths Since the dynamic programming algorithm works by determining a least-cost path through a matrix, we are faced with the problem that there may be multiple optimal paths. In fact, this situation arises quite often. Ex. (2) shows two optimal paths/alignments for the pair *jrem* – *ihrem* ‘their’.[6]

(2) a.

Input	*j*		*r*	*e*	*m*
Operations	s	+	=	=	=
Output	*i*	*h*	*r*	*e*	*m*

b.

Input		*j*	*r*	*e*	*m*
Operations	+	s	=	=	=
Output	*i*	*h*	*r*	*e*	*m*

The ambiguity is obviously of no consequence for the numerical distance, which is two for both cases, but it makes a big difference for the rules. In particular, it is usually very clear that one of the alignments is the “correct” one (reflecting facts about language change, here: Ex. (2a)), while the other one is an implementation artifact (Ex. (2b)).

Rule sets and sequence-based rules To solve the ambiguity problem, we first pick a random path by preferring substitution over deletion over addition on the cell level in the matrix. The rules derived this way are combined to rule *sets* afterwards. This is done by inspecting the positions in the source and target word where the edits are made. Whenever a series of edits occurs at the same target or source position, we assume that this is actually an insertion or deletion of a *sequence* of characters, such as an affix. Whenever edits occur at adjacent positions, we assume that it is a substitution of a character sequence by another. By merging substitutions with adjacent deletions/additions, we account for character sequences of variable length on each side of the rule.

[6]Operations: ‘+’ means insertion, ‘–’ deletion, ‘s’ substitution, ‘=’ identity.

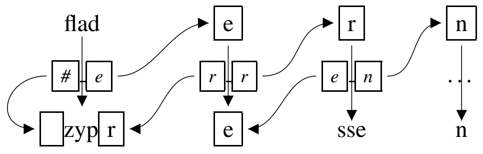

Figure 2: The graph illustrates the actual mappings derived from *fladernholtz – zypressenholz* ‘cypress wood’. The central row shows the rule contexts and how they are determined.

In the example *jrem – ihrem* above (Ex. (2)), the first two rules would be merged since they are derived from adjacent non-identity edit operations. This leads to the correct solution to the multi-path problem, shown in Ex. (3).

(3)

Input	*j*	*r*	*e*	*m*
Operations	s	=	=	=
Output	*ih*	*r*	*e*	*m*

Note however that the rule merging does not always come up with entirely correct alignments. For the word pair *fladernholtz – zypressenholz* ‘cypress wood’, optimal alignment would map `flader→zypresse/#_n` whereas the actual algorithm outputs smaller mappings, see Fig. 2. Indiscriminate merging of identity rules would result in highly specific mappings of entire words rather than character sequences—hence, identity rules are never merged.

Epsilon identity rules In the system developed so far, insertion rules are rather difficult to handle. In generating modern wordforms by means of the rewrite rules (see Sec. 5), they tend to apply in an uncontrollable way, garbling the words in the process. This is due to the fact that the conditions for the application of insertion rules are less specific: while substitutions and deletions require the left hand side (LHS) *and* the context to match in the source word, insertions are constrained by their two context characters only, since the LHS of the rule is empty. At word boundaries, the problem gets even worse, since only one context side is specified here. Furthermore, substitution and deletion rules compete against the identity rules for their LHS, which reflects the fact that the majority of characters are not changed in a mapping to the modernized form—but no similar competitor exists so far to curb the application of insertion rules.

We therefore introduced epsilon identity rules: after all replacement rules for an alignment pair have been generated, an epsilon identity rule is inserted between each pair of non-insertion rules, i.e. at each position where no insertion has taken place. The epsilon identity rules are taken to mean that no insertion should be performed in the respective context, thereby restricting the actual insertion of characters.

	Rank	Frequency	Rule
=	1	24,867	$\varepsilon \rightarrow \varepsilon$ / n _ #
=	2	18,213	$\varepsilon \rightarrow \varepsilon$ / e _ r
=	3	18,200	$\varepsilon \rightarrow \varepsilon$ / e _ n
=	4	17,772	$\varepsilon \rightarrow \varepsilon$ / # _ d
=	5	14,871	$\varepsilon \rightarrow \varepsilon$ / r _ #
=	6	14,853	n $\rightarrow$ n / e _ #
s	20	8,448	v $\rightarrow$ u / # _ n
-	176	1,288	f $\rightarrow \varepsilon$ / u _ f
+	239	932	$\varepsilon \rightarrow$ l / o _ l
	156	1,443	j $\rightarrow$ ih / # _ r
	272	796	j $\rightarrow$ ih / # _ n
	329	601	j $\rightarrow$ ih / # _ m
	605	263	$\varepsilon \rightarrow$ ␣d / t _ u
	879	142	ss $\rightarrow$ ß / o _ e

Table 3: Sample rankings and rules

Ranking of the rules Applying the rule induction algorithm to the development corpus yielded about 1.1 million rule instances (training corpus: 2.2 million) of about 6,500 different types (training: 7,902). These were sorted and ranked according to their frequency. Table 3 lists sample instances of rules. Not surprisingly, none of the top-ranked rules modifies the input word. Rank 6 is taken by the first rule that maps some real character rather than ε to itself (identity rules, ‘=’). Rank 20 features the first substitution rule (‘s’), etc. The bottom part of the table lists frequent sequence-based rules. The rule ranked 605th maps the empty string to a whitespace followed by ‘d’. This rule applies in mappings such as *soltu – sollst du* ‘should you’.

5 Generating Normalized Wordforms

Normalizing ENHG texts is done on a word-by-word basis, i.e. the input of the normalizing process is always a single wordform. Words are processed from left to right; for each position within a word, applicable edit rules are determined. As with the rule extraction process, the left context is taken from the output already generated up to that point, while the right context is always taken from the input word. If a rule is applied, its right-hand side is appended to the output string, and the next character from the input word is processed. The process continues until the end of the word has been reached.

Rules with sequences of characters on the left-hand side (LHS) are applicable at the position of their first LHS character. In that case, if the rule is applied, processing continues with the next character that is not part of the LHS.

Epsilon rules Epsilon rules, and epsilon identities in particular, require special consideration to work within this system. If an epsilon identity is applied, no other epsilon rules should be allowed for the same context, otherwise insertions would not be blocked. To achieve this, epsilon is treated like an ordinary letter with regard to the LHS, and words are preprocessed so that exactly one epsilon is placed between each character and at word boundaries. For example, if the input word is *jrem* 'theirs', it is converted internally to the following form:

(4) # ε j ε r ε e ε m ε #

Now, if an epsilon rule is applied, the read/write head moves to the next character, thereby ensuring that no other epsilon rule can be applied at the same time in the same position. This also prevents application of multiple insertion rules but at the same time still permits insertion of multiple characters, since those are merged into sets during rule extraction. Of course, epsilon characters have to be ignored for all purposes except for the LHS of replacement rules; in particular, they do never influence rule contexts or prevent the recognition of character sequences on the LHS.

Ranking methods For each character and its context within a word, there will usually be a number of applicable rules to choose from. As our aim is to generate exactly one (modernized) form for each input word, a decision has to be made about which rule to apply. Two approaches were tested: (i) selecting the rule which had the highest frequency during rule acquisition ('best-rule'); and (ii) selecting the word with the highest probability score ('best-probability'), which is calculated based on rule frequencies.

(i) 'Best-rule': In the first approach, if more than one rule application is possible at a given position, we simply select the rule with the highest frequency. The evaluation shows that this method already works quite well, however, it also has a disadvantage: as the left context for a rule depends on the output of the previous one, applying one rule can result in different rules being applicable at later positions in the word. If we always apply the most frequent rule, this can create situations in which only very low-frequent rules are applicable later, indicating that the resulting wordform is unlikely to be correct. Also, when applying a rule with a sequence of characters on the LHS, replacement rules that modify any but the first character of that sequence are never even considered.

For this reason, we came up with another method that takes into account the frequencies of all applied rules across the whole word.

(ii) 'Best-probability': In the second approach, each generated variant is assigned a probability score. We define the probability of a replacement rule as its frequency divided by the sum of all rule frequencies. The word probability is calculated from the probabilities of the rules that were used to generate it; for this, we used the weighted harmonic mean, with the length of the LHS as weights. If the LHS contains a sequence, length is counted including additional epsilons between each character. This way, all variants generated from the same input word have the same total weight, regardless of whether sequence-based rules were used or not.

Table 4 lists sample mappings as they result from both methods.

	Original bible words		**Generated words**	
	Old	Modernized	Best-rule	Best-prob.
1.	vnd	und	✓	✓
	jrem	ihrem	✓	✓
	vmbher	umher	✓	✓
2.	wetter	wetter	✓	*dieuetter
	krefftige	kräftige	✓	*krefftige
	vrteil	urteil	✓	*urteill
3.	fewr	feuer	*feur	✓
	soltu	sollst du	*soltu	✓
4.	ermanen	ermahnen	*ermanen	*ermannen
	zween	zwei	*zween	*zwen

Table 4: Example mappings (prior to the dictionary lookup) from the development corpus. '✓' means that the word is generated correctly by the respective method, '*' marks incorrectly generated wordforms. Words listed under 1. are generated correctly by both methods, words under 4. by neither of them, the rest by one of the methodes.

Dictionary lookup Quite often, the highest-ranked rules generate non-existing words. This

		Total	**Identical Tokens**		**NLD**
		Count	Count	Ratio	Mean ± SD
Old bible text (Baseline)	All	109,972	71,163	64.71%	0.1019 ± 0.1630
	Identical	71,163	71,163	**100.00%**	**0.0000** ± 0.0000
	Non-identical	38,809	0	0.00%	0.2889 ± 0.1460
	Unknowns	2,911	1,190	40.88%	0.1215 ± 0.1359
Best-rule method	All	109,972	91,620	83.31%	0.0408 ± 0.1039
	Identical	71,163	70,467	99.02%	0.0018 ± 0.0198
	Non-identical	38,809	21,153	54.51%	0.1122 ± 0.1483
	Unknowns	2,911	1,390	***47.75%***	***0.1081*** ± 0.1357
Best-probability method	All	109,972	92,172	***83.81%***	***0.0396*** ± 0.1041
	Identical	71,163	69,358	97.46%	0.0042 ± 0.0279
	Non-identical	38,809	22,814	***58.79%***	***0.1046*** ± 0.1509
	Unknowns	2,911	1,255	43.11%	0.1201 ± 0.1407
Best-probability + dictionary method	All	109,972	100,074	**91.00%**	**0.0251** ± 0.0913
	Identical	71,163	70,854	***99.57%***	***0.0008*** ± 0.0126
	Non-identical	38,809	29,220	**75.29%**	**0.0697** ± 0.1423
	Unknowns	2,911	2,238	**76.88%**	**0.0646** ± 0.1428

Table 5: Evaluation of exact matches and normalized Levenshtein distance (NLD) compared to the modernized bible text; separately for all tokens (All), tokens that are or are not identical in both old and modernized version (Identical/Non-identical), and tokens that were not seen in the training corpus (Unknowns). The best result for each class is indicated in bold, the second-best in bold italics.

can be avoided by combining the methods described above with a dictionary lookup. For this, all variants are generated and then matched against a dictionary. From all variants that are covered by the dictionary, the one with the highest score (best-rule or best-probability) is selected as the output form. If no variant can be generated in this way, the input word is left unchanged. The dictionary used here consists of all wordforms from the modernized NHG Luther bible.[7]

6 Evaluation

For evaluation, we generated normalized forms of all words in the evaluation corpus and compared them to their aligned forms in the modernized bible text. Two methods of comparison were applied: (i) counting the number of identical wordforms; and (ii) calculating the average normalized Levenshtein distance (NLD). Full evaluation results are shown in Table 5. Results for the dictionary method are only reported in combination with the best-probability method, which clearly outperform the combination with the best-rule method.

Exact matches As our aim is to generate modernized wordforms from historical ones, the logical first step of an evaluation is to check how many words from the ENHG text from 1545, when processed with our algorithms, exactly match their modernized NHG counterparts. Before normalization, the ratio of identical tokens in the historical and the modernized text is about 65%, i.e., only a third of all wordforms even differ at all. This is the baseline for our algorithm; any normalizing process that results in less than 65% exact matches has likely done more harm than good, and it would be better to leave all words unchanged. Table 5 shows that both ranking methods we employed, best-rule and best-probability, achieve match ratios above 83% (lines 'All', column 'Ratio'), which is a significant increase from the baseline; combining the best-probability method with the dictionary lookup even yields 91% exact matches. Our normalization approach is not only successful in changing historical forms to modern ones, but also in correctly leaving most of the wordforms unchanged that do not need to be changed (97.46–99.57%; lines 'Identical', column 'Ratio').

Results from evaluating the dictionary method on the annotated sample set of non-identical word pairs are shown in Table 2. Although the sample size is very small, the numbers suggest that our approach is mostly suitable for pairs of type 1 (besides identical word pairs); in particular, it can only 'repair' some inflection or affixes (types 2–3). Incorrect mappings (type 6) are not produced.

[7]Using a dictionary with wordforms from current newspaper texts turned out problematic, since modern abbreviations, typos, etc., result in too many false positives with the dictionary lookup. Moreover, the vocabulary of newspaper texts differs considerably from religious texts.

	Total	Identical Tokens		NLD
	Count	Count	Ratio	Mean ± SD
Best-rule	18,352	696	3.79%	0.2483 ± 0.1368
	18,352	0	0.00%	**0.2443** ± 0.1226
Best-prob.	17,800	1,805	10.14%	**0.2372** ± 0.1495
	17,800	0	0.00%	0.2447 ± 0.1297
Dict.	9,898	309	3.12%	0.2876 ± 0.1528
	9,898	0	0.00%	0.2789 ± 0.1479

Table 6: NLD evaluation of word pairs that do not match their aligned word after normalization. The first line of each method shows the NLD before rule application, the second line afterwards.

Normalized Levenshtein distance However, simply counting correct guesses does not take into account near-misses, where the algorithm edited only a part of the word correctly, and is also not a very fine-grained way of evaluation. Therefore, we chose to use normalized Levenshtein distance (Beijering et al., 2008) to assess whether our normalized variants are 'closer' to the correct modern version than the (non-normalized) source words. The NLD of a word pair is defined as the Levenshtein distance divided by the length of the longest possible alignment of the two words.[8] To evaluate a set of word pairs, we calculate the average NLD of all word pairs in that set. This measure is not ideal for our task, since short words are unduly penalized for being wrong, but it has the advantage of being relatively intuitive; e.g., a NLD of 0.5 indicates that roughly every second character in a word was altered.

Comparing the old and the modernized bible text yields an average NLD of 0.1019; as two thirds of all words are already identical, this number is quite low. The three normalization methods reduce that number to 0.0408–0.0251; this reduction stems, in parts, from the higher match ratio. To evaluate whether the normalization improved the wordforms even if they do not exactly match the aligned form, we re-evaluated average NLD on the sets of words that did not result in an exact match. The results, given in Table 6,[9] show only a slight improvement for the best-rule method, while the best-probability method has even increased the average NLD. With the latter method, a high percentage of mismatches (10.14%) results from source words that did not need to be changed. Combined with the dictionary lookup, the ratio of such cases is considerably reduced (to 3.12%). Even the dictionary method is not able to completely avoid superfluous modifications. The reason is that there are ambiguous words such as *waren*, which can either be mapped to *wahren* 'true' or left unchanged: *waren* 'were'. This means that at the type level it cannot be decided which of the two mappings is correct. Instead, we would need context information at the token level, which is beyond our word-based approach.[10]

Method comparison Even though the results for both ranking methods are quite close when evaluated on all word pairs (83.31% versus 83.81%), the difference is statistically significant within a confidence level of 99.9%, i.e., the best-probability method results in better normalizations on average. This is especially reflected in the numbers for the non-identical word pairs, where the difference between the two methods is even greater. On the other hand, the best-rule method performs better on 'unknowns', i.e. words which were not already part of the training set (see Table 5). This seems to indicate that a combination of both methods could be favorable.

The overlap of the word lists generated by the two methods is 93.13%, showing that there is a noticeable percentage of words (around 3%) which is modernized correctly with one method but not the other. One crucial difference is the normalization of second person verb forms ending in *–tu*, e.g. *soltu* 'should you', which should be modernized to *sollst du*. These forms show a contraction of verb and pronoun and are quite common in the original ENHG bible text. With the best-rule method, they do not get changed at all, as the epsilon identity rules are ranked higher than the ones that would perform the necessary insertions. The best-probability method, on the other hand, outputs the correctly modernized form; rules that process the letter *u* appearing word-final after *t* have a very low probability, thus decreasing the total probability of the unmodified wordform.

Finally, combining the methods with a dictionary lookup results in remarkable improvements.

[8]As a side effect of our rules induction process, we can easily calculate the number of alignment slots, since it equals the total number of edit and (non-epsilon) identity rules.

[9]Words considered here include (i) words from identical word pairs that unnecessarily have been modified (this, e.g., concerns 696 words with the best-rule method) and (ii) words from non-identical word pairs that have not been normalized successfully (best-rule: 18,352–696 = 17,656 words).

[10]Jurish (2010) takes context information into account.

The number of exact matches increases from 83.31% to 88.66% (best-rule), and from 83.81% to 91.00% (best-probability). As can be seen from Table 5, the dictionary lookup on the one hand helps to avoid superfluous normalization (with identical word pairs). On the other hand, it also improves on rule application, by filtering out rules (or combinations of rules) that do not result in sensible words. In contrast to rules, which operate locally, the dictionary lookup has access to the entire wordform and thus serves as a complementary "guide" for suitable rule selection.

Since we use the complete modernized Luther bible as the source of our dictionary, all correctly normalized wordforms are guaranteed to be listed in the dictionary. In this sense, the results represent an upper bound of this method. However, the larger a dictionary is, the higher will be the chance of "false friends", i.e. wordforms that accidentally match a generated wordform.

7 Conclusion

We showed that using only unsupervised learning, a minimum of knowledge engineering, and freely available resources, it is possible to map historical wordforms to their modern counterparts with a high success rate. Even the simplest implementation of the process performs far better than the baseline, a success that we were able to further improve upon.

Open issues include the multi-path problem, which is still not entirely solved, despite the introduction of sequence-based rules (see Section 4 and especially Fig. 2). There are a number of potential solutions that we could pursue. The rule extraction process could be modified to output all possible paths from the source to the target word. This would require some means to decide on the most plausible path, such as the total number of rules after the single-character rules have been merged to sets. Phonetic or even graphemic similarity (such as the common substitutions of *u* and *v*) could also be taken into account.

Another issue would be to replace our simple heuristic of sequence determination by the use of an association measure, such as the log-likelihood ratio or the Fisher-Yates test, to determine which rules are merged to sequences. This could include identity rules in the sequences, which might solve problems such as the one presented in Fig. 2.

Association measures could be further used to determine the significance of the association between a rule and its context, and to potentially abstract the rules from their specific contexts.

Another open question is the handling of multi-tokens as source words. Since currently the rule application operates on single words, multiple source tokens can never be merged to a single target token, even though that happens quite frequently in our texts.

References

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

Karin Beijering, Charlotte Gooskens, and Wilbert Heeringa. 2008. Predicting intelligibility and perceived linguistic distances by means of the Levenshtein algorithm. *Linguistics in the Netherlands*, pages 13–24.

Werner Besch. 2000. Die Rolle Luthers für die deutsche Sprachgeschichte. In Werner Besch et al., editor, *Sprachgeschichte. Ein Handbuch zur Geschichte der deutschen Sprache und ihrer Erforschung*, pages 1713–1745. de Gruyter, 2nd edition.

Fabienne Braune and Alexander Fraser. 2010. Improved unsupervised sentence alignment for symmetrical and asymmetrical parallel corpora. In *Proceedings of COLING, Poster Volume*, pages 81–89.

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

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

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

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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

Norbert Richard Wolf. 2000. Handschrift und Druck. In Werner Besch et al., editor, *Sprachgeschichte. Ein Handbuch zur Geschichte der deutschen Sprache und ihrer Erforschung*, pages 1705–1713. de Gruyter, 2nd edition.