Normalisation of Historical Text Using Context-Sensitive Weighted Levenshtein Distance and Compound Splitting

Eva Pettersson[1,2]*, Beáta Megyesi*[1] *and Joakim Nivre*[1]

(1) Department of Linguistics and Philology, Uppsala University
(2) Swedish National Graduate School of Language Technology

firstname.lastname@lingfil.uu.se

ABSTRACT

Natural language processing for historical text imposes a variety of challenges, such as to deal with a high degree of spelling variation. Furthermore, there is often not enough linguistically annotated data available for training part-of-speech taggers and other tools aimed at handling this specific kind of text. In this paper we present a Levenshtein-based approach to normalisation of historical text to a modern spelling. This enables us to apply standard NLP tools trained on contemporary corpora on the normalised version of the historical input text. In its basic version, no annotated historical data is needed, since the only data used for the Levenshtein comparisons are a contemporary dictionary or corpus. In addition, a (small) corpus of manually normalised historical text can optionally be included to learn normalisation for frequent words and weights for edit operations in a supervised fashion, which improves precision. We show that this method is successful both in terms of normalisation accuracy, and by the performance of a standard modern tagger applied to the historical text. We also compare our method to a previously implemented approach using a set of hand-written normalisation rules, and we see that the Levenshtein-based approach clearly outperforms the hand-crafted rules. Furthermore, the experiments were carried out on Swedish data with promising results and we believe that our method could be successfully applicable to analyse historical text for other languages, including those with less resources.

KEYWORDS: Digital Humanities, Natural Language Processing, Historical Text, Normalisation, Levenshtein Edit Distance, Compound Splitting, Part-of-Speech Tagging, Underresourced Languages, Less-Resource Languages.

1 Introduction

When working with natural language processing (NLP) for historical text, one problem is that there are often not large enough amounts of annotated corpus data available for training NLP tools specifically aimed at handling historical text. Nevertheless, using existing NLP tools as they are is rarely an option, since these generally rely on dictionaries and/or statistics based on certain word forms observed in the training data. Spelling in historical texts however differs from contemporary spelling, and due to the lack of spelling conventions, spelling may also vary between different authors, genres and time periods, and even within the same text written by the same author.

Even though there are differences between old and modern text, there are also similarities, and a native speaker of the language is often able to understand all or part of a historical text in spite of the odd spelling. Bearing this in mind, one way to get around the lack of NLP tools for historical text is to first normalise the input text to a more modern spelling, before applying existing NLP tools trained on contemporary corpora. In this context, the normalisation process may be viewed as a kind of spelling correction, where the historical word form is treated as a misspelling, that should be corrected to the most probable modern spelling.

In this paper we describe an approach to normalisation of historical text based on Levenshtein edit distance, where certain edits can be weighted to reflect common (and thus more likely) spelling variation in texts from this time period. Apart from handling single edit operations, we also include operations involving multiple characters. Another innovative feature of our approach is the use of a compound splitter for normalising parts of a compound word individually. Finally, we explore the effect of using manually annotated historical data to learn the normalisation of frequent words in a supervised fashion. We evaluate our results on historical Swedish data, both regarding normalisation accuracy, and by running a modern tagger on the historical text, before and after normalisation.

The outline of this paper is as follows. Section 2 describes related work on normalisation of historical text. In Section 3, we present the data used in our experimental setup, whereas our approach is described in detail in Section 4. In Section 5, we present the results. Finally, conclusions and future work are discussed in Section 6.

2 Related Work

The use of NLP tools for analysing historical text is still largely unexplored, even though there is a rapidly growing interest in this field. The idea of implementing spelling correction techniques for normalising the input text to a more modern spelling before applying existing NLP tools is a popular approach, and may be performed using rule-based or data-driven methods, or a combination of both.

Rayson et al. (2005) tried a rule-based approach based on dictionary lookup. A large mapping scheme from historical to modern spelling for 16th to 19th century English texts was manually created. The resulting VARD tool (VARiant Detector) comprises 45,805 entries, and also includes fuzzy matching techniques and context rules for normalisation of ambiguous words. The performance of the normalisation tool was evaluated on a set of 17th century texts, and compared to the performance of modern spell checkers (MS-Word and Aspell) on the same text. The results showed that between a third and a half of all tokens (depending on which test text was used) were correctly normalised by both VARD and MS Word, whereas approximately one third of the tokens were correctly normalised only when using VARD. The comparison

between VARD and Aspell showed similar results. VARD was later further developed into VARD2, combining the original word list with data-driven techniques in the form of phonetic matching against a modern dictionary, and letter replacement rules based on common spelling variation patterns (Baron and Rayson, 2008).

Another rule-based method, based on pattern matching, was developed by Pettersson et al. (2012). In this experiment, a relatively small set of 29 hand-crafted normalisation rules for handling spelling variation in Early Modern Swedish texts (1550–1800) was produced, based on a subset of a 17th century court records text. The resulting rule set was applied to a gold standard corpus of 600 sentences (33,544 tokens) extracted from 15 documents within two separate genres (court records and church documents), varying in time from 1527 to 1812. Even though the rule set had been developed on a single 17th century court records text, normalisation had a positive effect on texts from all centuries and for both genres within the scope of the study. In fact, the largest error reduction was achieved for a 16th century church document. On average, an error reduction of 22% was achieved, meaning that approximately 73% of the tokens were correctly normalised.

A third rule-based technique, referred to as conflation by phonetic form, was explored by Jurish (2008) for normalisation of historical German text. In this approach, the similarity between phonetic forms is computed, rather than comparing orthographic forms. The general assumption is that since there were no orthographic conventions for writers of historical text, spelling generally reflects the phonetic form of the word. Furthermore, it is assumed that phonetic properties are less resistant to diachronic change than orthography. For the grapheme-to-phoneme transition, they used the conversion module of the IMS German Festival text-to-speech system (Black and Taylor, 1997), with a rule-set adapted to historical word forms. The method was evaluated on a corpus of historical German verse quotations extracted from *Deutsches Wörterbuch*, containing 5,491,982 tokens (318,383 types). Without normalisation, approximately 84% of the tokens were known to their morphological analyser. After conflation by phonetic form, coverage was extended to 92% of the tokens. When adding lemma-based heuristics to this method, coverage increased further to 94% of the tokens.

As an extension to the work on conflation by phonetic form, Jurish (2010) further explored the impact of taking the sentential context into consideration in the normalisation process. For this purpose, a Hidden Markov Model (HMM) was developed for contextual disambiguation of a set of normalisation candidates that were originally extracted on a token level using four different normalisation techniques: string identity, transliteration, phonetization and rewrite transduction. The resulting HMM produced precision scores of 99.7% and recall scores of 99.1% on a test corpus of 152,776 tokens from the time period 1780–1880, extracted from the *Deutsches Textarchiv*.

A data-driven approach based on Levenshtein similarity was presented by Bollmann et al. (2011) for normalisation of Early New High German (14th to 16th century). Normalisation rules were automatically derived by means of the Levenshtein edit distance, based on a word-aligned parallel corpus consisting of the Martin Luther bible in its 1545 edition and its 1892 version, respectively. Since bible text is rather conservative in spelling and terminology, approximately 65% of the words in the old bible version already had an identical spelling to the one occurring in the more modern version. For non-identical word forms, only word forms that could be found in the modern version of the bible were accepted. Other word forms produced by the Levenshtein-based rules were discarded, leaving the old spelling preserved. Using this method,

the proportion of words with an identical spelling in the normalised text as compared to the "modern" text increased from 65% to 91%. Bollmann (2012) also tried a combination of dictionary lookup and different modifications to the original Levenshtein distance measure, improving normalisation accuracy further to 92.6% for the same training and test corpora.

3 Data

In this study, we make use of a 787,122 token corpus consisting of 15 Swedish texts, from the genres of court records and church documents, ranging from 1527 to 1812. The texts have been automatically digitized, and only parts of the corpus have been manually reviewed and corrected with regard to OCR errors. For evaluation we use the same subset of this corpus as was used for evaluation by Pettersson et al. (2012), i.e. 600 sentences extracted by randomly selecting 40 sentences from each text in the entire data set. For training, i.e. for estimating different parameters of our model, we extracted another 600 sentences out of the remaining sentences in the corpus (40 randomly selected sentences from each text). The resulting token distribution for training and evaluation is given in Table 1.

Court Records				
Name	**Year**	**Total**	**Training**	**Evaluation**
Östra Härad	1602–1605	38,477	1,980	2,069
Vendel	1615–1645	64,977	1,583	2,509
Per Larsson	1638	12,864	2,848	2,987
Hammerdal	1649–1686	75,143	1,508	1,859
Revsund	1649–1689	113,395	2,275	2,328
Stora Malm	1728–1741	458,548	1,627	1,895
Vendel	1736–1737	61,664	3,032	3,450
Stora Malm	1742–1760	74,487	2,034	2,336
Stora Malm	1761–1783	66,236	1,905	1,825
Stora Malm	1784–1795	58,738	2,036	1,378
Stora Malm	1796–1812	47,671	2,345	1,683
Church Documents				
Name	**Year**	**Total**	**Training**	**Evaluation**
Västerås	1527	14,149	2,831	3,709
Kyrkoordning	1571	60,354	2,093	2,246
Uppsala Möte	1593	34,877	1,070	1,184
Kyrkolag	1686	35,201	1,660	2,086
Total	**1527–1812**	**787,122**	**30,827**	**33,544**

Table 1: Corpus distribution, given in number of tokens in the documents. Total = Number of tokens in the whole corpus. Training = Number of tokens in the training sample of the corpus. Evaluation = Number of tokens in the evaluation sample of the corpus.

4 Method

This paper explores an approach to normalisation of historical text using the Levenshtein edit distance measure for comparing the original word form to word forms listed in an electronically available dictionary or corpus of contemporary language. Our approach is similar to the one presented by Bollmann et al. (2011) in that Levenshtein similarity is used in the normalisation

process. In the Bollmann approach, however, the Levenshtein weights are calculated on the basis of aligned parallel data, consisting of the same text in an old spelling version and a modern spelling version, respectively. Our approach does not presuppose access to such a corpus, since the Levenshtein distance is solely computed by comparing the historical word forms to tokens present in a contemporary dictionary or corpus.

Two steps are included in the normalisation process: *generation of normalisation candidates* and *candidate selection*. As stated above, the generation of normalisation candidates is performed by comparing the historical word form to the tokens present in a contemporary language resource. In our experiments we use the SALDO dictionary (version 2.0) for this purpose. SALDO is an electronically available lexical resource developed for present-day written Swedish, comprising approximately 100,000 lexical entries (Borin et al., 2008). The dictionary entry/entries with the smallest edit distance as compared to the original word form are stored as possible normalisation candidates, given that certain requirements are met concerning string length and edit distance, as further discussed in Section 4.1.

The generation of candidates may optionally be refined, to improve precision further. For example, if there is a set of manually normalised historical tokens available, this corpus could be included as a *validated cache*, mapping historical word forms to a manually validated modern spelling. The validated cache is consulted before possible normalisation candidates are generated based on Levenshtein distance. If the token is present in the validated cache, the normalised form found in the cache is chosen, and no Levenshtein-based candidates are generated.[1] Such a corpus could also be used for optimisation of the Levenshtein calculations, as described further in Section 4.1. Another optional refinement that we have experimented with is the addition of a *compound splitter* (Stymne, 2008). In this case, the token is first matched against the contemporary language resource as usual. If no normalisation candidates are found by the Levenshtein comparisons, the word is split into its compound parts (provided that the splitter regards the token as a compound) and each compound part is matched separately against the contemporary language resource by means of Levenshtein calculations. Finally, the resulting normalised parts are merged into a compound normalisation candidate.

When possible normalisation candidates have been extracted, one candidate is to be selected as the final choice. In the default setting, a final candidate is randomly chosen from the list of highest-ranked candidates. The candidate selection step can however be further refined, if there is a corpus of modern language available. In this case, the candidate with the highest frequency in the corpus is chosen. Only if none of the highest-ranked normalisation candidates are present in the corpus, or if there are several candidates with the same frequency distribution, a final candidate is randomly chosen. In our experiments, we use the Stockholm-Umeå corpus (SUC) (Ejerhed and Källgren, 1997) for frequency calculations. SUC is a balanced corpus consisting of approximately one million tokens extracted from a number of different text types representative of the Swedish language in the 1990s.

The whole normalisation process, as described above, is illustrated in Figure 1, where optional resources are marked by dotted lines and arrows.

[1] We also include *unvalidated* items in the cache, solely for reasons of efficiency. This means that normalisations previously performed by the algorithm, as well as tokens present in the modern dictionary, are stored in the cache and consulted as a first step in the normalisation process.

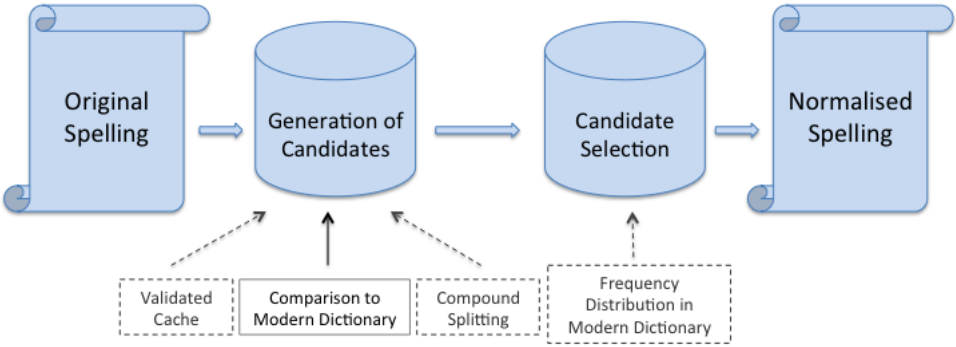

Figure 1: Overview of the normalisation workflow. Optional resources are dotted.

4.1 Parameter Optimisation

The Levenshtein distance gives an indication of the similarity between two strings (Levenshtein, 1966). Still, in the context of normalisation, it is not a trivial decision what string similarity to regard as close enough for considering a normalisation candidate as valid. In a traditional spelling correction context, it has been shown that for the majority of the human-generated spelling errors, the misspelled word form is within one letter in length of the intended word form, and that most errors constitute single instances of insertion, deletion, substitution or transposition (Kukich, 1992). If no empirical data on the correlation between historical and modern spelling is available, one could thus assume that a valid normalisation candidate should be maximally one character shorter or longer than the original word form, with a maximum Levenshtein distance of 1. In the following we explore the characteristics of our manually normalised training data as regards edit distances and differences in string length between the original word forms and their normalised counterparts. The aim is to find out whether the traditional spelling correction assumptions hold also in the context of normalisation, or (if not) what the optimal configuration would be. We also explore the inclusion of context-sensitive, weighted edit operations, compound handling and supervised learning by means of manually validated normalisations.

4.1.1 Observed String Length Differences

To decide what string length differences to consider in the normalisation process, we investigated the differences in string length between the original word forms and their normalised counterparts in the training corpus. As previously stated, the training corpus comprises in total 30,827 tokens (see further Section 3). Approximately 35% of these tokens (10,735) differ from the manually normalised spelling. As shown in Table 2, the assumption that most errors do not influence string length by more than one character may still be regarded as appropriate even in the context of normalisation. However, the proportion of tokens meeting this requirement is limited to approximately 86.9%. Instead, if we consider all cases where the normalised word form is at most one character longer down to four characters shorter than the original word form, approximately 99.5% of the tokens observed in the training corpus will be covered. We will therefore focus on varying the valid difference in string length within this interval.

Length Difference	Frequency	Example	Translation
−7	1	*besluthninger/beslut*	'decisions'
−6	3	*fadherbrodher/farbror*	'uncle'
−5	7	*noghsambligha/nogsamma*	'careful'
−4	127	*närvarellse/närvaro*	'presence'
−3	308	*slechttenn/släkten*	'the relatives'
−2	918	*kyrckian/kyrkan*	'the church'
−1	4804	*aff/av*	'of/off'
0	3326	*wara/vara*	'be'
+1	1201	*til/till*	'to'
+2	35	*sokn/socken*	'parish'
+3	2	*tilbragte/tillbringade*	'spent'
+5	3	*församhs/församlingens*	'the congregation's'

Table 2: Observed differences in string length between the historical word form and the manually normalised version in the training corpus. Grey rows illustrate string length differences with a special focus in our experimental setup.

4.1.2 Observed Edit Distances

As illustrated in Table 3, we see that if we consider only normalisation candidates with a maximum Levenshtein distance of 1 as compared to the original word form, we merely cover approximately 55.8% of the tokens in the training corpus. Furthermore, we see that all edit distances up to and including 4 have a high frequency in the training corpus. If we consider an edit distance span where 4 is the maximum value, we cover 98.8% of the observed entities in the training corpus. Hence, to broaden normalisation coverage, we will focus our experiments on considering edit distances up to and including 4.

Edit Distance	Frequency	Example	Translation
1	5986	*sigh/sig*	'oneself/himself/herself/itself'
2	3008	*dher/där*	'there'
3	1161	*blefwe/blev*	'became'
4	455	*afwachta/avvakta*	'await'
5	86	*öfwertalter/övertalad*	'persuaded'
6	28	*söllfuermynthe/silvermynt*	'silver coin'
7	10	*sielffuer/själv*	'himself/herself/itself'
8	1	*öffuergiffua/överge*	'abandon'

Table 3: Observed edit distance between the historical word form and the manually normalised version in the training corpus. Grey rows illustrate edit distances with a special focus in our experimental setup.

4.1.3 Weighted Edit Operations

When computing a traditional Levenshtein distance, all edit operations have a cost of one. However, in the context of normalisation of historical text, some edits seem more likely than others. For example, as historical texts to some degree are written in a spoken-language fashion, substituting the letter *c* for a *k* is more likely than substituting for example a *c* for an *r*. For this

reason, we have experimented on assigning weights lower than 1 for frequently occurring edits in the training corpus.

Out of the 8,881 edits observed in the training corpus, we have assigned special weights to edits occurring at least 50 times. The weights have been calculated by comparing the frequency of each edit occurring in the training corpus to the frequency with which the specific source characters are left unchanged, in accordance with the formula below:

$$1 - \frac{\text{FrequencyOfEdit}}{\text{FrequencyOfEdit+FrequencyOfSourceCharacterLeftUnchanged}}$$

which could be further simplified into:

$$\frac{\text{FrequencyOfSourceCharacterLeftUnchanged}}{\text{FrequencyOfEdit+FrequencyOfSourceCharacterLeftUnchanged}}$$

To illustrate, the deletion of the letter *h* happens in 2,382 cases in the training corpus, when comparing the historical spelling to the modernised spelling. In the rest of the 3,912 cases where the letter *h* occurs, it is preserved in the modernised spelling. Thus, the weight for the deletion of the letter *h* is calculated as: $\frac{3912}{2382+3912} \approx 0.6215$.

In some cases, the observed edits involve more than one character, for example when comparing the historical spelling *oförsichtigt* to the modernised version *oförsiktigt* ("carelessly"). This would intuitively be regarded as a substitution of *ch* for *k*, rather than first substituting the *c* for a *k*, and then deleting the *h*. To deal with cases like this, we therefore included the following context-sensitive edit operations, in addition to the standard single-character operations:

- double deletion
  example: -fv, *gifvvett* → *givet*
- double insertion
  example: +es, *orklöse* → *orkeslöse*
- single-to-double substitution
  example: l/ll, *tilbytt* → *tillbytt*
- double-to-single substitution
  example: ss/s, *frälsse* → *frälse*

For all strings in the training corpus that are not identical in the historical and the modern spelling, all possible single-character edits as well as multi-character edits are counted. Hence, the token pair *oförsichtigt-oförsiktigt* will yield weights for (i) single deletion of *c*, single substitution of *h*→*k*; (ii) single deletion of *h*, single substitution of *c*→*k*; and (iii) double-to-single substitution of *ch*→*k*.

4.1.4 Compound Handling

The Levenshtein method described in this paper is based on comparison of a historical word form to similar word forms found in a modern dictionary. Since the Swedish language has a high degree of compounds, some of the intended words will inevitably not be found in the dictionary, even if the word could perfectly well be used in contemporary Swedish. Stymne and Holmqvist (2008) showed that in their Swedish evaluation corpus, approximately 37% of all the words with a length of 12 characters or longer were compounds (and approximately 5% of the shorter words). This was calculated on modern Swedish European Parliament text, but might still be indicative of the frequencies in historical Swedish texts as well. To deal with the

compounding issue, we included a compound splitter developed by Stymne (2008). When the compound splitter is included, any word form for which no appropriate normalisation candidate is found by the ordinary Levenshtein calculations, is run through the compound splitter. If the splitter is able to split the word into compound parts, each part is processed separately by the normalisation program, and the resulting normalisation candidates are merged into a final normalisation candidate. For example a word like *krigztienst* will be split into *krigz* (normalised as *krigs* "war") and *tienst* (normalised as *tjänst* "duty"). The two normalised versions *krigs* and *tjänst* will then be merged into the final normalisation candidate *krigstjänst*.

4.1.5 Manually Validated Normalisations

Even though historical texts present a high degree of spelling variation, some word forms are frequently occurring in many of these texts. Incorrect normalisation of a few frequently occurring word forms might result in poor NLP performance, even when a high proportion of the remaining word forms has been normalised correctly. Therefore we will explore the effect of adding a set of manually validated normalisation candidates for the most frequently occurring word forms in the training corpus. When such a word form is encountered in a text, only the manually validated set of normalisation candidates will be considered, and no Levenshtein distance is computed for finding alternative candidates.

Since each token in the training corpus has been manually assigned its normalised counterpart, we use this annotation for providing a cache with information on what manual normalisation(s) that have been assigned for the word form in question. In order to avoid incorrect attempts at normalisation, the caching procedure also involves word forms where the original spelling has been preserved in the manual validation, as illustrated by the first entry in Table 4, where the historical word form *och* is mapped to its identical modern spelling *och* ("and"). Furthermore, if one word form has been manually normalised in several ways in the training corpus, all candidates are stored in the cache, as illustrated by the second entry, where the historical word form *i* is mapped both to the modern spelling *i* ("in") and to *ni* ("you"). A sample from the manually validated normalisations is given in Table 4. In order to explore the impact of caching manually validated normalisations, we will consider different frequency thresholds in our experimental setup.

Frequency	Original form	Normalised Form(s)	Translation(s)
1,144	*och*	*och*	'and'
634	*i*	*i* or *ni*	'in' or 'you'
242	*at*	*att*	'to/that'
229	*på*	*på*	'on'
219	*af*	*av*	'of'
121	*effter*	*efter*	'after'
109	*thet*	*det*	'it'
92	*ther*	*där*	'there'
45	*j*	*i*	'in'
35	*meth*	*med*	'with'

Table 4: A sample from the manually validated normalisations observed in the training corpus.

5 Results

Evaluation of our proposed Levenshtein-based normalisation algorithm has been performed on the basis of the evaluation corpus (see further Section 3). The results are presented below, both in terms of actual normalisation accuracy and by measuring the performance of a modern tagger on the historical text, before and after normalisation.

5.1 Normalisation

Normalisation results are evaluated using two different measures: *normalisation accuracy* and *error reduction*. By normalisation accuracy we mean the percentage of tokens in the normalised version of the text that are identical to the manually reviewed tokens in the gold standard, whereas error reduction refers to the percentage of correctly normalised tokens that were not originally identical to the gold standard spelling. Error reduction has been calculated by the following formula:

$$\frac{\text{CorrectAfterNormalisation} - \text{CorrectBeforeNormalisation}}{\text{IncorrectBeforeNormalisation}}$$

where *CorrectAfterNormalisation* is the percentage of tokens with an identical spelling to the gold standard version *after* normalisation, *CorrectBeforeNormalisation* is the percentage of tokens with an identical spelling to the gold standard version *before* normalisation, and *IncorrectBeforeNormalisation* is the percentage of tokens differing in spelling from the gold standard version before normalisation.

Furthermore, we have experimented on different restrictions for the algorithm, regarding 1) valid string length difference between the original word form and its normalisation candidate (as described in Section 4.1.1), 2) valid edit distance between the original word form and its normalisation candidate (as described in Section 4.1.2), 3) special weights for frequently occurring edits (as described in Section 4.1.3), compound handling (see further Section 4.1.4, and 4) a frequency threshold for inclusion of manually validated normalisation candidates in a cache (as described in Section 4.1.5).

It could also be noted that in the following, normalisation accuracy and error reduction are calculated on the evaluation corpus as a whole, without taking differences between genres and age of the subtexts into account. For an investigation of the impact of age and genre on normalisation performance, see further Pettersson et al. (2012).

5.1.1 String Length Restrictions

Table 5 shows the normalisation results when varying the threshold for valid difference in string length between the original word form and its normalisation candidate. As shown in Section 4.1.1, 99.5% of the manually normalised word forms in the training corpus are within the range of one character longer to four characters shorter compared to the original historical word form. We experimented with different settings within this range. No manually validated cache entries nor compound handling or weighted edits are included at this stage, and the accepted edit distance for a normalisation candidate to be regarded as valid is set to 4, i.e. the value needed to capture the majority of the instances in the training corpus.[2] The results are also compared to the baseline, i.e. the unnormalised version of the evaluation corpus, and to

[2]It would of course have been preferable to have no limit on the accepted edit distance in these experiments. However, due to computational complexity, a threshold had to be included.

the approach of Pettersson et al. (2012) using hand-crafted rules. We see that approximately 64.6% of the original tokens already have a spelling identical to the manually normalised gold standard spelling. The hand-written rules have a positive impact, improving accuracy to 72.7%. This is however outperformed by the Levenshtein approach, for which accuracy is higher in all settings. Furthermore, we see that the general spelling correction assumption that the original string should not differ in length with more than one character from the intended word form does not hold in this context, since we get better results if we increase the valid difference in string length so that the normalised word form may be two or three characters shorter than the original form. Increasing this range to four characters however does not have a noticeable effect on the results. In the following we will therefore regard the $+1$ to -3 setting as the setting to be used as a basis for further experiments.

Approach	Accuracy	Error Reduction
Baseline	64.6%	n/a
Hand-written	72.7%	22.9%
Levenshtein stringdiff $+1$ to -1	76.2%	32.7%
Levenshtein stringdiff $+1$ to -2	76.9%	34.9%
Levenshtein stringdiff $+1$ to -3	77.0%	35.0%
Levenshtein stringdiff $+1$ to -4	77.0%	35.0%

Table 5: Normalisation accuracy and error reduction for different normalisation settings. Baseline = Unnormalised version of the evaluation corpus. Hand-written = Normalisation using a set of hand-crafted rules, as presented by Pettersson et al. (2012). Levenshtein = The normalisation approach described in this paper. Stringdiff = Valid difference in string length between the original word form and its normalisation candidate(s).

5.1.2 Edit Distance Restrictions

As presented in the previous section, normalisation results are influenced by what string length differences we accept for a normalisation candidate to be considered valid. Another important factor to take into consideration is the maximum edit distance allowed. We have experimented on varying the accepted edit distance between 1 and 4. However, if we statically accept an edit distance of for example 3, a three-letter word that is to be normalised may be transformed into a completely different word, where each character in the original word has been substituted by another character. To avoid such cases, we also experimented on a stringlength-based measure for deciding the maximum edit distance allowed for a certain source word. First, we calculated the average number of edits per character for all historical words in the training corpus that were not identical to the modern spelling. This showed an average edit distance of 0.27/character. With a standard deviation of approximately 0.18, the valid edit distance is then calculated as follows (where 1.96 times the standard deviation is added to cover 95% of the cases):

$$\text{distance} \leq 1 \text{ OR distance} \leq ((0.27 * \text{wordlength}) + (1.96 * 0.18))$$

The first condition is included to assure that one edit is always allowed, also in short words consisting of only one or two characters. The above formula could be further simplified into:

$$\text{distance} \leq \max(1, ((0.27 * \text{wordlength}) + (1.96 * 0.18)))$$

Table 6 presents our findings on varying the accepted edit distance between 1 and 4, as well as including the stringlength-based method. For all settings in this experiment, the string length restrictions are set to the best-performing settings as presented in the previous section, i.e. letting the normalisation candidates differ in string length from the original word form by at most +1 to −3 characters. Again, the results show that we may not rely on findings from general spelling correction observations, stating that edit distance should not exceed a value of 1. In our data, we get better results when increasing this threshold to 2, 3 or 4. We can also see that the stringlength-based method is slightly less successful than the approach where we always allow for an edit distance of 3 or 4. In the following we will therefore regard a Levenshtein distance threshold of 4 as the setting to be used for further experiments.

A closer look at the results of the stringlength-based method as compared to the static Levenshtein method, shows that the stringlength-based method has a higher precision, but a lower recall. The cases where a word form has been normalised in an incorrect way is substantially lower in the stringlength-based method, whereas the proportion of word forms that have incorrectly been left unnormalised is higher. This is particularly striking for words consisting of 4–9 characters. Both shorter and longer words are more or less equally well handled by both methods. Thus, in cases where precision is considerably more important than recall, the stringlength-based method could be preferred.

Approach	Accuracy	Error Reduction
Baseline	64.6%	n/a
Hand-written	72.7%	22.9%
Levenshtein max distance 1	74.7%	28.5%
Levenshtein max distance 2	76.7%	34.1%
Levenshtein max distance 3	77.0%	34.9%
Levenshtein max distance 4	77.0%	35.0%
Levenshtein stringlength-based	76.2%	32.9%

Table 6: Normalisation accuracy and error reduction for different normalisation settings. Baseline = Unnormalised version of the evaluation corpus. Hand-written = Normalisation using a set of hand-crafted rules, as presented by Pettersson et al. (2012). Levenshtein = The normalisation approach described in this paper. Max distance = Maximum valid edit distance between the original word form and its normalisation candidate(s).

5.1.3 Introducing Lower Weights for Frequently Occurring Edits

As stated in Section 4.1.3, some edits are more likely to occur than others. Table 7 presents the results of adding weights lower than 1 to frequently occurring differences when comparing the spelling of the original historical word form to the spelling of the corresponding manually normalised form in the training corpus. For this experiment, the string length and edit distance restrictions are set to the best-performing settings as presented in the previous sections, i.e. letting the normalisation candidates differ in string length from the original word form by at most +1 to −3 characters, with an allowed edit distance of maximally 4.

As seen from the results, the inclusion of context-free weights lower than 1 means that normalisation accuracy increases from 77.0% to 78.7% as compared to normalisation without special weights. Including context-sensitive weights as well, the normalisation accuracy increases further to 79.1%. The training corpus used in this experiment is however rather small (10,735

tokens with a spelling that is different from the manually normalised spelling). It is likely that the inclusion of weights would have a larger impact on normalisation accuracy if more training data was available.

Approach	Accuracy	Error Reduction
Baseline	64.6%	n/a
Hand-written	72.7%	22.9%
Levenshtein no weights	77.0%	35.0%
Levenshtein context-free weights	78.7%	39.9%
Levenshtein context-sensitive weights	79.1%	40.9%

Table 7: Normalisation accuracy and error reduction for different normalisation settings. Baseline = Unnormalised version of the evaluation corpus. Hand-written = Normalisation using a set of hand-crafted rules, as presented by Pettersson et al. (2012). Levenshtein no weights = The normalisation approach described in this paper, with no special weights included. Levenshtein weights = The normalisation approach described in this paper, with weights lower than 1 for frequently occurring edits.

5.1.4 Compound Handling

For compound handling, we experimented on adding a compound splitter to the normalisation process. In this setting, words that are not found in the modern dictionary by the ordinary normalisation approach, are split into their compound parts, and Levenshtein distance is calculated for each compound part separately. This feature has a small but positive effect on the normalisation process. The normalisation accuracy is still 79.1%, as for the best Levenshtein setting presented in the previous section. At a token level however, 26,997 tokens in the normalised text are identical to the manually modernised spelling when no compound handling is performed, as compared to 27,013 identically spelled tokens in the compound setting.

5.1.5 Varying the Threshold for Manually Validated Normalisations

As argued in Section 4.1.5, a small number of frequently occurring incorrectly normalised word forms may have a large negative impact on the result of contemporary NLP tools when applied to the normalised text. To avoid this, we created a cache function for manually validated normalisations observed in the training corpus. As could be expected, the manually validated cache boosts normalisation performance further. Table 8 summarises the results of varying the frequency threshold for what words to include in the cache. In this table, cache100 means that only word forms that occur 100 times or more in the training corpus are included in the cache. Based on our training corpus, this means that the cache of manually validated word forms holds 28 entries. In this setting, accuracy improves from 79.1% to 80.9%, increasing error reduction from 40.9% to 46.0%. If the word forms in the training corpus are sorted by frequency before manual normalisation is performed, this improvement is achieved by a rather modest manual annotation effort, since only 28 word forms need to be normalised. Including less frequent words in the cache improves accuracy as well as error reduction in all cases. When including all tokens in the validated corpus, regardless of frequency, an overall accuracy of 86.9% is achieved, corresponding to an error reduction of 63.0%.

Approach	Cache Entries	Accuracy	Error Reduction
Baseline	n/a	64.6%	n/a
Hand-written	n/a	72.7%	22.9%
Levenshtein, no cache	n/a	79.1%	40.9%
Levenshtein, cache100	28	80.9%	46.0%
Levenshtein, cache50	51	81.5%	47.7%
Levenshtein, cache40	71	81.8%	48.6%
Levenshtein, cache30	107	82.1%	49.4%
Levenshtein, cache20	156	82.5%	50.6%
Levenshtein, cache10	319	83.5%	53.4%
Levenshtein, cache1	8272	86.9%	63.0%

Table 8: Normalisation accuracy and error reduction for different normalisation settings. Baseline = Unnormalised version of the evaluation corpus. Hand-written = Normalisation using a set of hand-crafted rules, as presented by Pettersson et al. (2012). Levenshtein = The normalisation approach described in this paper. Cache Entries = Number of entries in the manually validated cache.

5.2 Tagging

The ultimate goal of the normalisation process is to improve the performance of contemporary NLP tools when applied to historical texts. In the scope of this study, we do not have access to a fully linguistically annotated data set for assessing for example tagger performance before and after normalisation. However, the corpus used in our experiments has been created within the Gender and Work project (Ågren et al., 2011), where there is a specific interest in the verbs included in the text. Therefore, all verbs in the evaluation corpus have been manually annotated as verbs, and we may indirectly evaluate the performance of the tagger based on precision and recall measures regarding verb identification. In these experiments, part-of-speech tagging is performed using HunPOS (Halácsy et al., 2007), a free and open source reimplementation of the HMM-based TnT-tagger by Brants (2000). The tagger is used with a pre-trained language model based on the Stockholm-Umeå Corpus (SUC).

Table 9 presents precision and recall measures for verb extraction, based on the best settings for our Levenshtein approach as regards normalisation accuracy, i.e. 1) allowing for the normalisation candidates to differ in string length from the original word form by at most +1 to −3 characters, 2) with a valid edit distance of maximally 4, 3) with weights lower than 1 for frequently occurring edits, 4) with compound handling, and 5) with a threshold of 1 for manually validated normalisations in the cache. The results are also compared to verb extraction from unnormalised text, as well as verb extraction based on normalisation using manually written rules. The hand-crafted normalisation rules have a positive impact on the results, increasing recall from 64.2% for unnormalised text to 78.0%. With the Levenshtein approach this figure is increased further to 86.2%. At the same time, precision increases from 77.9% to 80.0% with the rule-based approach, and increases further to 83.3% using the Levenshtein method. To get an idea of the optimal performance on historical text, we also tried verb extraction based on the manually normalised gold standard version of the test text. This resulted in a precision of 90.1% and a recall of 92.3%. This could be compared to the evaluation scores achieved for verb extraction for contemporary text, in this case the SUC corpus, with a 99.1% precision and recall. This indicates that even with perfect spelling normalisation, some differences in language such

as vocabulary, morphology and syntax, still need to be handled to achieve the same results as for modern language.

Approach	Precision	Recall	F-score
Baseline	77.9%	64.2%	70.4%
Hand-written	80.0%	78.0%	79.0%
Levenshtein	83.3%	86.2%	84.7%
Gold Standard Normalisation	90.1%	92.3%	91.2%
Contemporary text (SUC)	99.1%	99.1%	99.1%

Table 9: Precision and recall measures for verb identification based on the approach of hand-written rules (Pettersson et al., 2012) versus Levenshtein distance calculations. Baseline = Verb identification results when no normalisation is performed.

6 Conclusion

In this paper, we have proposed an approach to normalisation of historical text based on string similarity, using context-sensitive, weighted Levenshtein edit distance combined with compound splitting. We have evaluated our method both with respect to normalisation accuracy and regarding verb identification performance using a contemporary part-of-speech tagger on the normalised version of the input text.

With the presented approach, the proportion of tokens with an identical spelling to the gold standard spelling increased from a baseline of 64.6% (for unnormalised text) to 77.0%, using no manually annotated data for training. When including Levenshtein weights and a validated cache created on the basis of a small, manually normalised historical training corpus, normalisation accuracy increased further to 86.9% in the best setting. Furthermore, it is encouraging that both precision and recall increase for verb identification based on tagging, since the ultimate goal of the normalisation procedure is to enable the use of contemporary NLP tools for analysing historical data. The results are indeed promising, since our approach requires little or no manually annotated historical data, and is generalisable to similar (or somewhat distant) normalisation tasks for historical or other low density languages.

Future work includes refinement of the Levenshtein approach to normalisation, as well as exploring alternative techniques. One refinement could be to make the choice of normalisation candidates context-sensitive. Currently, the final normalisation candidate is solely chosen on a word-by-word basis, meaning that the same candidate will be chosen regardless of context. As for alternative techniques, it would be interesting to experiment with a phonetically based similarity measure for generating normalisation candidates. This seems reasonable since the lack of spelling conventions probably made the texts more alike spoken language than today.

Finally, it would also be interesting to try our normalisation approach in a different setting, such as other time periods and languages. This is enabled by the modularity and generalisability of the appoach, only requiring access to an electronically available dictionary of contemporary language. Modules such as the cache of manually validated normalisations could then be discarded in case of limited resources.

References

Ågren, M., Fiebranz, R., Lindberg, E., and Lindström, J. (2011). Making verbs count. The research project 'Gender and Work' and its methodology. *Scandinavian Economic History Review*, 59(3):271–291. Forthcoming.

Baron, A. and Rayson, P. (2008). Vard2: A tool for dealing with spelling variation in historical corpora. In *Postgraduate Conference in Corpus Linguistics*, Aston University, Birmingham.

Black, A. W. and Taylor, P. (1997). Festival speech synthesis system: system documentation. Technical report, University of Edinburgh, Centre for Speech Technology Research.

Bollmann, M. (2012). (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 Humanitites (ACRH-2)*.

Bollmann, M., Petran, F., and Dipper, S. (2011). Rule-based normalization of historical texts. In *Proceedings of the Workshop on Language Technologies for Digital Humanities and Cultural Heritage*, pages 34–42, Hissar, Bulgaria.

Borin, L., Forsberg, M., and Lönngren, L. (2008). Saldo 1.0 (svenskt associationslexikon version 2). Språkbanken, University of Gothenburg.

Brants, T. (2000). TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP)*, Seattle, Washington, USA.

Ejerhed, E. and Källgren, G. (1997). Stockholm Umeå Corpus. Version 1.0. Produced by Department of Linguistics, Umeå University and Department of Linguistics, Stockholm University. ISBN 91-7191-348-3.

Halácsy, P., Kornai, A., and Oravecz, C. (2007). HunPos - an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 209–212, Prague, Czech Republic.

Jurish, B. (2008). Finding canonical forms for historical German text. In Storrer, A., Geyken, A., Siebert, A., and Würzner, K.-M., editors, *Text Resources and Lexical Knowledge: Selected Papers from the 9th Conference on Natural Language Processing* (KONVENS 2008), pages 27–37. Mouton de Gruyter, Berlin.

Jurish, B. (2010). More Than Words: Using Token Context to Improve Canonicalization of Historical German. *Journal for Language Technology and Computational Linguistics*, 25(1):23–39.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10(8):707–710.

Pettersson, E., Megyesi, B., and Nivre, J. (2012). Rule-based normalisation of historical text - a diachronic study. In *Proceedings of the First International Workshop on Language Technology for Historical Text(s)*, Vienna, Austria.

Rayson, P., Archer, D., and Nicholas, S. (2005). VARD versus Word – A comparison of the UCREL variant detector and modern spell checkers on English historical corpora. In *Proceedings from the Corpus Linguistics Conference Series on-line e-journal*, volume 1, Birmingham, UK.

Stymne, S. (2008). German compounds in factored statistical machine translation. In Ranta, A. and Nordström, B., editors, *Proceedings of GoTAL, 6th International Conference on Natural Language Processing*, volume 5221, pages 464–475, Gothenburg, Sweden. Springer LNCS/LNAI.

Stymne, S. and Holmqvist, M. (2008). Processing of Swedish Compounds for Phrase-Based Statistical Machine Translation. In *Proceedings of the 12th EAMT conference*, Hamburg, Germany.