Unsupervised Normalisation of Historical Spelling

A Multilingual Evaluation

Nicklas Bergman
Abstract

Historical texts are an important resource for researchers in the humanities. However, standard NLP tools typically perform poorly on them, mainly due to the spelling variations present in such texts. One possible solution is to normalise the spelling variations to equivalent contemporary word forms before using standard tools.

Weighted edit distance has previously been used for such normalisation, improving over the results of algorithms based on standard edit distance. Aligned training data is needed to extract weights, but there is a lack of such data. An unsupervised method for extracting edit distance weights is therefore desirable.

This thesis presents a multilingual evaluation of an unsupervised method for extracting edit distance weights for normalisation of historical spelling variations. The model is evaluated for English, German, Hungarian, Icelandic and Swedish.

The results are mixed and show a high variance depending on the different data sets. The method generally performs better than normalisation based on standard edit distance but as expected does not quite reach up to the results of a model trained on aligned data. The results show an increase in normalisation accuracy compared to standard edit distance normalisation for all languages except German, which shows a slightly reduced accuracy, and Swedish, which shows similar results to the standard edit distance normalisation.
Acknowledgements

I’d like to thank my supervisor Eva Pettersson for her continuing support, advice and encouragement.
# Contents

Acknowledgements 3

1 Introduction 5
   1.1 Purpose and Research Questions 6
   1.2 Thesis Outline 6

2 Background 8
   2.1 Characteristics of Historical Text 8
   2.2 Spelling Normalisation 9
   2.3 Edit Distance 10
   2.4 Weighted Edit Distance 11
   2.5 Edit Distance for Spelling Normalisation 12
   2.6 Unsupervised Weight Extraction 14

3 Data and Method 16
   3.1 Data 16
      3.1.1 English 17
      3.1.2 German 17
      3.1.3 Hungarian 17
      3.1.4 Icelandic 17
      3.1.5 Swedish 18
   3.2 Experimental Method 19
      3.2.1 Additional Experiments 21

4 Results and discussion 22
   4.1 Units of Measure 22
   4.2 Edit Distance Threshold 22
   4.3 Results and discussion 22
      4.3.1 Results Compared to Normalisation without Weights 23
      4.3.2 Results Compared to Supervised Algorithm 23
      4.3.3 Performance on German and Swedish 24
      4.3.4 Types of Errors 25

5 Conclusion and Future Work 27
   5.1 Conclusion 27
   5.2 Future Work 28

Bibliography 29
1 Introduction

Large collections of historical documents are held in museums, libraries, and archives across the world. Many of these documents are still only available in their original - handwritten or perhaps printed - form, which makes searching for information in them and using them for research a laborious process. Therefore, there are in many countries ongoing efforts to digitise the documents, both in order to make the information in them more easily accessible but also to preserve them since the physical media will not last forever (Piotrowski 2012).

Converting the information within the documents to digital text requires either manual transcription or the use of language technology in the form of OCR\(^1\). Language technology, or natural language processing (NLP) is also used for tasks such as searching in the digitised documents or performing linguistic analyses on them (Jurafsky and Martin 2014).

Unfortunately, the tools which are used for natural language processing of contemporary text often perform badly on their historical equivalents, because historical text is generally different in many respects (Piotrowski 2012). As described in Piotrowski (2012) and Pettersson (2016), the lack of standardised writing conventions often result in historical text having features such as a freer word order and an inconsistent usage of punctuation. However, perhaps the largest obstacle to NLP of historical texts is orthography, or spelling variations. It is also the feature which affects NLP results the most.

As a solution to these problems you could consider adapting modern NLP tools to work on historical text, but there are a number of difficulties involved in this. Most modern NLP tools depend on supervised statistical methods, meaning they need annotated data to train on (Jurafsky and Martin 2014). The lack of consistency displayed in historical texts means there are fewer examples of each data point to train for instance a tagger\(^2\) or a parser\(^3\), meaning there is a problem of data sparsity. In addition to the data sparsity caused by spelling variations, there generally aren’t as many resources available in the first place for historical text as there are for modern text (Piotrowski 2012).

All these obstacles might lead us to consider adapting the text to the NLP tools instead of the opposite. As mentioned above, spelling is a prominent difference between historical and modern text and since it has a high impact on NLP performance (Pettersson 2016), it seems prudent to focus on it.

---

\(^1\) Optical character recognition is the automatic conversion of images of machine- or handwritten characters into digital text (Jurafsky and Martin 2014).

\(^2\) A tagger, or part-of-speech tagger, is a program for automatically labelling words with their parts-of-speech (Jurafsky and Martin 2014).

\(^3\) A parser is a program which takes text as an input and produces as an output a linguistic structure of some sort. A parser could for instance be used to analyse a sentence in terms of grammatical constituents or dependencies (Jurafsky and Martin 2014).
Pettersson (2016) described several methods of normalising historical spellings in her doctoral thesis. Among these methods is one based on comparing the historical word forms to the word forms in a contemporary dictionary in terms of edit distance, a concept which is explained further in chapter 2. In order to improve the accuracy of the normalisation, Pettersson (2016) used training data consisting of historical word forms mapped to their modern spelling to learn weights. These weights discounted certain edits which were used often in the training data when transforming a historical word form to its modern equivalent. As an example, *th* is often substituted for just *t* when normalising both historical German and historical Swedish. This edit is then made cheaper than others and therefore more likely to be selected. Again, the concepts are described in more detail in chapter 2.

In order to extract the weights, a parallel corpus of historical word forms mapped to their historical equivalents is needed. This can be problematic due to the lack of such material and the laborious process of producing it. Hauser and Schulz (2007) describe an unsupervised method for extracting edit distance weights which they evaluate on Early New High German. In this thesis a multilingual evaluation of a similar method for unsupervised weight extraction is presented.

### 1.1 Purpose and Research Questions

This thesis addresses problems of modernising historical texts in order to be able to apply standard NLP software to them. The purpose of this thesis is to perform a multilingual evaluation of an automated and unsupervised method for generating edit distance weights for the purpose of historical spelling normalisation, without the need for parallel corpora. The resources used are instead separate historical and modern corpora along with modern lexica.

The model will be tested and evaluated for the languages of English, German, Hungarian, Icelandic and English. The resources for those languages were readily available which is why they were chosen. The results will be compared to those gained from using both regular edit distance for normalisation as well as weighted edit distance with weights extracted from manually normalised training data.

The thesis aims to answer the following questions:

- Which level of quality is achievable when it comes to unsupervised historical spelling normalisation using weighted edit distance as opposed to normalisation using standard edit distance without weights?

- Which level of quality is achievable when it comes to unsupervised historical spelling normalisation using weighted edit distance as opposed to normalisation using manually annotated data?

- Are the results generalisable across different languages?

### 1.2 Thesis Outline

The outline of the thesis is as follows. Chapter 2 contains background information for the thesis as well as a description of related work. In chapter 3 the data sets
and the experimental method used in the thesis are described. Chapter 4 contains the results of the normalisation experiments, along with a discussion. Finally, in chapter 5 the thesis comes to an end with conclusions and suggestions for future work.
2 Background

This chapter describes background information relevant to the thesis. The chapter starts with a description of what characterises historical text and moves on to describe what spelling normalisation entails. The following section defines what edit distance is, what weighted edit distance entails, and how they can be used for spelling normalisation. Finally, the chapter ends with a description of a method for unsupervised spelling normalisation with edit distance weights.

2.1 Characteristics of Historical Text

Historical text displays several different characteristics that distinguish it from modern text and are relevant for NLP. As described in Pettersson (2016), the concept of historical language is wide and encompasses texts from many different time periods, genres and languages. The features of one historical text are therefore not necessarily present in another. However, the following features are regularly encountered when dealing with historical text. Historical text typically differs from contemporary text when it comes to syntax, morphology, vocabulary, semantics and orthography. In general, the older a text is, the more it will differ from contemporary text. However, different languages have changed at different rates. Icelandic has remained relatively static for a thousand years, while English has changed dramatically during the same time period, thanks to the influences of the viking and Norman invasions (Pettersson 2016).

In terms of syntax, it is common for the lack of standardised writing conventions for historical language to lead to a freer word order than is often present in contemporary languages. When it comes to morphology, historical languages often display a higher complexity, with more types of morphological forms and irregular inflections. The syntactic and morphological differences make parsing the historical text with NLP tools trained on modern corpora a difficult prospect (Campbell 2013; Pettersson 2016).

Another characteristic of historical text is inconsistent punctuation. This leads to ambiguous sentence boundaries and makes automatic sentence segmentation, an important preprocessing step in NLP, more difficult (Pettersson 2016).

The vocabularies of historical languages often differ from their modern equivalents, as languages both gain and lose words over time. New words are often loan words borrowed from other languages. Campbell (2013) cites tomato and chocolate as examples, which both trace their origins to the Aztec language Nahuatl. As examples of languages losing words over time, van Gelderen (2006) lists Early Modern English words which no longer remain in use, including illecebrous ("alluring") and ingent (immense). Such words are obviously difficult for modern NLP tools to deal with since they would not have seen them before. Normalising them to a contemporary equivalent is also a
difficult prospect since they aren’t simply spelling variations with contemporary equivalents.

When it comes to semantics, it is common for the meanings of words to have shifted over time, which can cause confusion for a modern reader of historical text. Campbell (2013) mentions nice as an example, which originally meant “foolish, stupid, senseless” as well as silly, which originally had the meaning of “blessed, blissful”.

Another common phenomenon in historical text is code switching, which is the seamless mixing of two languages in the same text. Code switching can cause trouble for NLP tools since they are usually not made to process two different languages at once (Campbell 2013; Pettersson 2016).

Perhaps most significantly, historical language differs from contemporary language in terms of orthography, which as mentioned in chapter 1 has the largest effect on NLP results. Historical text presents two kinds of spellings variations. The first is synchronic spelling variations, meaning orthography varies within the same time period due to a lack of writing conventions. The second is diachronic spelling variations, meaning the spelling changes over time. The synchronous spelling variations are so prevalent that the same word is often spelt in different ways in the same text by the same author. In addition, the spelling can also vary due to errors in transcription or in the OCR process. NLP tools trained on contemporary language will likely show poor performance on historical text because they do not recognize the historical word forms (Piotrowski 2012).

2.2 Spelling Normalisation

As mentioned in chapter 1, one solution to the poor performance of modern NLP tools on historical text is to map the historical spelling variations to their modern counterparts. As an example, the historical word forms wych and wich would both be replaced by the modern spelling of which. This is referred to as spelling normalisation. Spelling normalisation allows tools trained on modern training data, which is more widely available, to also be used on historical text (Piotrowski 2012).

Several different approaches to spelling normalisation have been tried. Rayson et al. (2005) use a dictionary-based approach in their VARiant Detector (VARD), which was developed for use on Early Modern English. With the VARD tool, historical spelling variations are mapped to modern canonical spellings through the use of a dictionary. When the tool is used all words are looked up in the dictionary and those with matches are replaced with their modern spellings. The dictionary was compiled manually from Early Modern English text sources. According to Piotrowski (2012), the dictionary-based normalisation method performs surprisingly well, despite its simplicity. However, a drawback of the method is the difficulty of including all possible spelling variations in a pre-defined list.

Another approach to spelling normalisation involves transformation rules for mapping a historical word form to its contemporary equivalent. Pettersson (2016) uses a manually compiled set of rewrite rules to normalise Early Modern Swedish. The rules are based both on established historical linguistic knowledge of how
the spelling has changed over time and empirical findings from a historical text. Example rules include the deletion of double vowels as in *een* (one) becoming *en* and the transformation of *dt* to *t* as in *varidt* (been) becoming *varit*. Bollmann et al. (2011) also use a rule-based method for spelling normalisation, but they create the rules automatically from a word-aligned parallel corpus by recording the operations needed to go from a historical word form to a modern word form. One advantage of rule-based methods is that they do not require each word to be normalised to be defined beforehand. A single rule can be used to normalise many words. However, manually defining rules can be laborious and require knowledge of historical linguistics. Bollmann et al. (2011) mention a disadvantage in their automatic method for generating rules in that their approach cannot handle spelling variations that were not encountered in the training data.

Spelling normalisation is usually treated as an error correction problem, similar to spelling correction. However, one can also see it as a translation problem. Scherrer and Erjavec (2013) use character-based statistical machine translation (SMT) to normalise historical Slovene. Pettersson (2016) similarly uses character-based SMT in a multilingual normalisation experiment. In statistical machine translation the goal is to find the most probable string in a target language given a string in a source language. SMT requires training data in the form of aligned parallel corpora. Normally, statistical machine translation works at the level of phrases and words, but in this case the units are instead characters and words. Spelling normalisation with character-based SMT performs well even with relatively small amounts of training data (Pettersson 2016).

Techniques similar to those used in spelling correction are often also used for spelling normalisation. One of those techniques is *edit distance*, which is described in further detail in the sections below.

### 2.3 Edit Distance

Edit distance is a method for quantifying the difference between two word forms. It is often used in spell checkers (Jurafsky and Martin 2014). Edit distance works by defining *edit operations* and then counting how many of those operations are necessary to transform a source string into a target string. The most common form of edit distance is known as Levenshtein distance (Levenshtein 1966), which defines the operations of insertion, deletion and substitution of a character.

In spell checkers, or when normalising historical spelling variations, the candidates for correcting or normalising the historical word form can be found by selecting words from a contemporary dictionary with a low edit distance to the original word form.

The distance between identical word forms is defined as 0. For every insertion, deletion or substitution necessary to transform a source word into a target word, the distance increases by 1.

The Levenshtein distance can be calculated with a recursive algorithm (Jurafsky and Martin 2014). Let \( d(i,j) \) denote the distance between the first \( i \) characters of string \( a \) and the first \( j \) characters of string \( b \). The following formula can then be
used to calculate the edit distance:

\[
d(o, o) = 0 \\
d(i, o) = i \\
d(o, j) = j \\
d(i, j) = \min \begin{cases} 
  d(i - 1, j) + 1 \\
  d(i, j - 1) + 1 \\
  d(i - 1, j - 1) + \begin{cases} 
    0 & \text{if } a_i = b_j \\
    1 & \text{if } a_i \neq b_j 
  \end{cases}
\end{cases}
\]

The first case in the minimum corresponds to the deletion of a character, the second to insertion, and the third to either equality or substitution.

As an example, the Levenshtein distance between golden and holding is 3. The minimum edit distance operations needed to transform the former into the latter are:

1. golden → holden (substitution of -g for -h)
2. holden → holdin (substitution of -e for -i)
3. holdin → holding (insertion of -g at the end)

2.4 Weighted Edit Distance

Certain spelling errors are more common than others. In terms of typographical mistakes, characters which are next to each other on the computer keyboard are often accidentally substituted for each other. In terms of normalising historical spelling, certain patterns reoccur when comparing historical and modern word forms for each language. Historical texts are written to some degree in a spoken-language fashion. Therefore, characters that are phonologically similar are more likely to be substituted for each other (Pettersson 2016). As an example, y is often substituted for i when normalising Early Modern English, such as in kyng and hym becoming king and him, respectively.

Standard edit distance does not account for this. As described in Jurafsky and Martin (2014), when using edit distance for spelling correction or normalisation you look in a dictionary for target word candidates within a certain edit distance of the original word form, most commonly 1.

In the example above and with everything else being equal, hym is as likely to be normalised to ham as to him, since both words have an edit distance of 1 to the original word form. If you want to take advantage of the fact that you know one of the substitutions is more likely, you can use weighted edit distance, as described in Pettersson (2016) and Piotrowski (2012).

For standard Levenshtein distance the edit distance between two word forms is equal to the sum of the minimum amount of edit operations needed to go from one form to the other. In other words, each operation has a cost of 1. However, that does not have to be the case. Changing the costs, or adding weights, can change the output of the edit distance algorithm (Piotrowski 2012).
In the example above, setting the weight of the substitution \( y \rightarrow i \) to a value lower than 1 would, everything else being equal, allow the correct normalisation candidate to be selected over others.

As described in Pettersson (2016), for weighted Levenshtein distance the edit distance is calculated by summing the weights of all edits needed to transform the source word form into the target word form.

Costs could be set to vary depending on the operation itself and/or depending on which characters are involved. For instance, if it was known that words in modern text were shorter in general than those in historical text, making the deletion operation cheaper than the others might improve the results of the normalisation.

### 2.5 Edit Distance for Spelling Normalisation

Edit distance has been used by several different researchers to normalise historical spelling variations. Kempken et al. (2006) compared different kinds of edit distance measures in terms of their effectiveness in an information retrieval task. Bollmann et al. (2011) used a modified Levenshtein distance algorithm to extract rules for a rule-based normalisation of historical spelling variations.

This thesis is based on the work of Pettersson (2016), who described several different methods of normalising historical spellings in her doctoral thesis and tested their effectiveness in a multilingual evaluation. One of the methods the author implemented and evaluated was weighted Levenshtein distance.

In addition to the historical text to be normalised, the Levenshtein-based approach also requires a contemporary dictionary and, optionally, a modern corpus. Each word in the historical text is compared to the words in the dictionary in terms of edit distance. The candidate with the lowest edit distance is chosen, given that the distance is below a given threshold. If several candidates are found within the same edit distance, the one which occurs most frequently in the modern corpus is chosen, if one is available. Otherwise, the normalised form is picked at random from the candidates.

In order to improve the normalisation accuracy Pettersson (2016) also added weights to the edit operations, discounting edits which were frequently observed in a training corpus consisting of historical word forms manually mapped to their modern spellings.

Pettersson (2016) created edit distance weights from the manually aligned training corpora by means of a modified edit distance algorithm which keeps track of the performed edits as it goes. Each historical word form in each training corpus was compared to its manually normalised equivalent and the operations needed to go from one form to the other were recorded. This is illustrated in figure 2.1. The first column in the figure displays the historical word form, the second column displays the normalised word form, and the third column is composed of the recorded edit operations needed to transform the historical spelling into the modern one. The operations are separated by vertical bars and show the characters involved as well as the operations performed on them. An alphabetical character without any additional symbols indicates that the character involved was identical in both the historical word form and the modern word form, meaning no operation...
Figure 2.1: Example output for Swedish of the algorithm which records edit operations. The columns are composed of the original word form, the normalised word form, and the edit operations performed, in order.

After the operations were recorded, the edits were then weighted based on their relative frequency in the training material. More specifically, the weights were set to the result of dividing the frequency of a character being left unchanged with the overall frequency of the character as seen in the formula below:

\[
\text{Edit weight} = \frac{\text{Frequency of character left unchanged}}{\text{Frequency of character}}
\]

To reduce noise, only edits which occurred 50 times or more were added as weights.

Some edits may involve more than one character. As an example, Pettersson (2016) compares the Early Modern English spelling of *personnes* to its modernised version *persons*. The author notes that the change would intuitively be regarded as the deletion of *-ne* rather than the separate deletions of *-n* and *-e*. Therefore, in addition to the single-character weights described in chapter 2, the following multi-character edit operations were also included (Examples by Pettersson (2016):

- Double deletion (*personnes* → *persons*)
- Double insertion (*strait* → *straight*)
- Single-to-double substitution (*juge* → *judge*)
- Double-to-single substitution (*moost* → *most*)

As described by Pettersson (2016), all possible single-character and multi-character weights are taken into account when the weights are extracted. For the example of *personnes* above, the edits counted include the double deletion *ne*, but also the deletions of *n* and *e*, as well as the double-to-single substitutions of *nn* → *n* and *ne* → *n*.

An example of how the weights can look can be seen in figure 2.2. The two different columns denote the edit operation with the characters involved as well
as the discounted cost, respectively. The minus sign indicates the deletion of the character or characters after the sign, while the plus sign signifies insertion. The forward slash between characters indicates that the first part is to be substituted for the second. Finally, the last column is the weight associated with the edit in question.

As previously mentioned, the process of extracting edit distance weights as described in Pettersson (2016) requires training data in the form of manually compiled mappings from historical spelling to modern spelling. These manually normalised corpora are not always available, and are time-consuming to produce, which is why an unsupervised method for weight generation is desirable.

### 2.6 Unsupervised Weight Extraction

Hauser and Schulz (2007) describe an unsupervised method for extracting edit distance weights. They use weighted edit distance to normalise Early New High German and evaluate the results in terms of an information retrieval task. Their technique is similar to the one used in this thesis to extract weights without aligned training data.

Hauser and Schulz (2007) use a noisy channel model described in Brill and Moore (2000), adapted from the domain of spelling correction to be used instead for historical spelling normalisation. In the noisy channel model, which is often used in spelling correction, the goal is to find an intended word based on a word where the letters are assumed to have been scrambled by "noise" such as typos (Brill and Moore 2000).

The model used in Brill and Moore (2000) and Hauser and Schulz (2007) uses generic string to string edits instead of just weights for single character operations. Instead of the different character operations of insertion, deletion and substitution, longer substring to substring are used. This increases the complexity of the algorithm but allows it to take more of the surrounding context into account when performing the normalisation.

Hauser and Schulz (2007) use a supervised algorithm which is also described in Brill and Moore (2000), with training pairs of words with historical and modern spelling to learn the weights for normalisation. However, they also present an
unsupervised method for generating weights, using a modern lexicon and a historical corpus but avoiding the need for training data. It consists of going through a normalisation process similar to the one outlined above as used by Pettersson (2016), comparing each historical word form to those in a contemporary dictionary, but with tighter restrictions in order to maximise the chance that a correct candidate is chosen. The edit distance operations used to normalise those words are recorded and used to create weights in order to improve the result of further normalisations.

When training their model, Hauser and Schulz (2007) set the maximum threshold for Levenshtein distance to one, meaning only modern word forms that are one edit from the historical form are chosen as candidates for normalisation. They also skip normalisation if more than one such candidate is found. The intuition behind both choices is to minimise the risk of errors by only picking normalisation candidates as close to the original word form as possible and only picking those where there is one possible option.

As can be expected, the supervised learning algorithm shows the best results in Hauser and Schulz (2007). However, the unsupervised weighted edit distance normalisation does perform better than normalisation based on regular Levenshtein distance, without weights.

As described earlier the normalisation algorithms in Hauser and Schulz (2007) were evaluated in terms of an information retrieval task while the algorithms in Pettersson (2016) and this thesis were evaluated directly in terms of normalisation accuracy, character error rate, etc. Additionally, in this thesis each historical word form is mapped to one equivalent modern word form. However, for the information retrieval task in Hauser and Schulz (2007), one modern word form is instead mapped to multiple historical word forms. The results are therefore not directly comparable, but can still give an indication of the performance of the different techniques.
3 Data and Method

This chapter first presents the language resources used in the thesis and then goes on to describe the experimental method.

3.1 Data

The language resources used in this thesis are the same as those in Pettersson (2016). There were two reasons for this choice. The first was convenience, as the data sets were readily available due to the author’s generosity, and already preprocessed and split into training and evaluation sets. Secondly, part of the aim of this thesis is to compare the results of the unsupervised weight extraction algorithm described in this thesis with those of the supervised method presented in Pettersson (2016). Using the same data sets is necessary for such comparisons to be fruitful. The method will be evaluated for the languages of English, German, Hungarian, Icelandic and Swedish.

As can be seen in table 3.1, the data sets used vary in terms of size, genre and time periods. Because of these variations it will be difficult to compare the results of the normalisation between the different languages. Instead, the aim is to test the generalisability of the method as well as how it performs compared to the supervised method used in Pettersson (2016) and to standard edit distance without weights.

The following resources are needed for each language in the spelling normalisation experiment:

- A training corpus of historical data with the original spellings intact. No modern equivalents are needed for this part since the method is unsupervised.
- An evaluation corpus of historical data with the original spellings as well as manually normalised modern equivalents to use as a gold standard.
- A contemporary dictionary to use for edit distance comparisons. If no dictionary is available a corpus of contemporary text can be used instead.
- A contemporary corpus to be used for frequency data regarding modern word forms.

The data set for each language in Pettersson (2016) was split into three parts; training, tuning and evaluation. The tuning parts were used to calculate maximum edit distance thresholds in Pettersson (2016), but since there is no unsupervised method for the calculations the tuning parts were not used in this thesis.

The resources used for each language are described below. A table with further details including token and type counts is presented further on in the chapter.
3.1.1 English

The historical corpus used for English is the *Innsbruck Corpus of English Letters*, a part of the *Innsbruck Computer Archive of Machine-Readable English Texts* or ICAMET (Markus 1999). This corpus is composed of a manually normalised collection of letters. The time period of the letters stretches from 1386 to 1698. The training part of the corpus is composed of 148,852 tokens, while the evaluation part is composed of 17,791 tokens.

A subset of the British National Corpus (BNC) is used both as a modern dictionary and as a modern corpus.

3.1.2 German

The historical corpus used for German is a subset of the *GerManC* corpus (Scheible et al. 2011). As described in Pettersson (2016), the subset used is manually normalised and is composed of 22 texts from the time period of 1659 to 1780. The genres of the texts are varied and include newspaper text, letters, sermons and narrative prose. The training part of this corpus is composed of 39,887 tokens, while the evaluation part was composed of 5,005 tokens.

The modern language resource used for German is the *Parole* corpus (Teubert 2003).

3.1.3 Hungarian

The historical resource used for Hungarian is a collection of codices from the *Hungarian Generative Diachronic Syntax* project, or HGDS (Ezter 2014). In total there are 11 codices covering the time period 1440 to 1541. The training part of the corpus contains 137,669 tokens while the evaluation part is composed of 17,214 tokens.

The modern resource used for Hungarian is the *Szeged Treebank* (Csendes et al. 2005).

3.1.4 Icelandic

The historical corpus used for Icelandic is a manually normalised subset of the *Icelandic Parsed Historical Corpus*, or IcePaHC (Rögnvaldsson et al. 2012). IcePaHC is diachronic corpus that covers the time period of 1150-2008. The subset contains four texts from the 15th century, including three sagas and one religious text (Pettersson 2016). The training part of the data set is composed of 52,440 tokens, while the evaluation part contains 6,384 tokens.

The modern language dictionary used for Icelandic is a combination of two different resources, namely *Beygingarlýsing Íslensks Nútímasl*, or BÍN, which is a collection of modern Icelandic inflectional forms (Bjarnadóttir 2012) and the *Tagged Icelandic Corpus of Contemporary Icelandic Texts*, or MÍM (Helgadóttir et al. 2012), from which all tokens which occurred 100 times or more were included. As described in Pettersson (2016), the reason for this threshold was the noisy nature of the corpus.
<table>
<thead>
<tr>
<th>Language</th>
<th>Part</th>
<th>Tokens</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>Training</td>
<td>148,852</td>
<td>18,267</td>
</tr>
<tr>
<td>English</td>
<td>Evaluation</td>
<td>17,791</td>
<td>4,573</td>
</tr>
<tr>
<td>English</td>
<td>Dictionary</td>
<td>2,088,680</td>
<td>69,153</td>
</tr>
<tr>
<td>English</td>
<td>Corpus</td>
<td>2,088,680</td>
<td>69,153</td>
</tr>
<tr>
<td>German</td>
<td>Training</td>
<td>39,887</td>
<td>9,055</td>
</tr>
<tr>
<td>German</td>
<td>Evaluation</td>
<td>5,005</td>
<td>1,966</td>
</tr>
<tr>
<td>German</td>
<td>Dictionary</td>
<td>18,662,434</td>
<td>662,510</td>
</tr>
<tr>
<td>German</td>
<td>Corpus</td>
<td>18,662,434</td>
<td>662,510</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Training</td>
<td>137,669</td>
<td>45,529</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Evaluation</td>
<td>17,214</td>
<td>8,798</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Dictionary</td>
<td>1,257,089</td>
<td>144,248</td>
</tr>
<tr>
<td>Hungarian</td>
<td>Dictionary</td>
<td>1,257,089</td>
<td>144,248</td>
</tr>
<tr>
<td>Icelandic</td>
<td>Training</td>
<td>52,440</td>
<td>9,748</td>
</tr>
<tr>
<td>Icelandic</td>
<td>Evaluation</td>
<td>6,384</td>
<td>2,244</td>
</tr>
<tr>
<td>Icelandic</td>
<td>Dictionary</td>
<td>27,224,798</td>
<td>2,820,623</td>
</tr>
<tr>
<td>Icelandic</td>
<td>Corpus</td>
<td>21,339,384</td>
<td>9,461</td>
</tr>
<tr>
<td>Swedish</td>
<td>Training</td>
<td>28,237</td>
<td>7,925</td>
</tr>
<tr>
<td>Swedish</td>
<td>Evaluation</td>
<td>33,544</td>
<td>8,859</td>
</tr>
<tr>
<td>Swedish</td>
<td>Dictionary</td>
<td>1,110,731</td>
<td>723,138</td>
</tr>
<tr>
<td>Swedish</td>
<td>Corpus</td>
<td>1,166,593</td>
<td>97,670</td>
</tr>
</tbody>
</table>

Table 3.1: Statistics for the language resources used in this thesis.

3.1.5 Swedish

The historical corpus used for Swedish is the Gender and Work corpus (Fiebranz et al. 2011). As described in Pettersson (2016), a subset of the complete Gender and Work material is used, which is comprised of 11 court records and 4 church documents. The time period of these documents stretches from 1527 to 1812 and together they comprise a total of 787,122 tokens.

Pettersson (2016) extracted 600 sentences from these documents to a training set, and another 600 to an evaluation set. The sentences were selected at random, with 40 being selected from each of the 15 texts. The training and evaluation parts of the corpus were then manually normalised by Pettersson (2016). The training part of the corpus ended up being composed of 30,827 tokens, while the evaluation part comprises 33,544 tokens.

The contemporary dictionary used for Swedish is SALDO, an electronic lexicon of modern Swedish containing approximately 1.1 million word forms (Borin et al. 2008).

The modern corpus used for Swedish is the Stockholms Umeå Corpus, which contains approximately 1 million words and is composed of Swedish representative of the 1990s (Ejerhed and Källgren 1997).
3.2 Experimental Method

The unsupervised method for weight extraction is very much built on the work of Pettersson (2016) and the steps involved are similar to those of her supervised algorithm, as described in section 2.5.

As illustrated in figure 3.1, the unsupervised method for weight extraction consists of performing a Levenshtein-based spelling normalisation without weights and with a low edit distance threshold, recording the edit operations performed in the normalisation, and then calculating edit distance weights based on those recorded edits. In essence, the results of an unweighted normalisation are used similarly to the aligned training data in Pettersson’s supervised algorithm. As described in Hauser and Schulz (2007), this could be described as the algorithm trying to find training pairs with an approximate search in a lexicon rather than using prepared mappings of historical word form to modern word form.

As can also be seen in figure 3.1, a modern dictionary and a modern corpus are required, in addition to the historical corpus. The modern dictionary is used to find the modern normalisation candidates, while the modern corpus is used to resolve ties by selecting the most frequent candidate. The historical corpus and the modern corpus are not required to be aligned or mapped to each other in any way, however.

One of the most important parameters one can set when running different experiments with edit distance is the maximum edit distance allowed between the historical word form and modern word forms when considering candidates for normalisation. This is also referred to as the edit distance threshold.

The intuition behind the unsupervised weight extraction algorithm in this thesis is that running the normalisation process with a low edit distance threshold is likely to result in a higher precision, since only the normalisation candidates very close to the original word form are selected.

A higher threshold for edit distance would instead result in a lower overall precision, but a higher recall since more words are being normalised instead of being skipped over due to a lack of appropriate normalisation candidates. This can be seen to hold true in table 4.1 and table 4.2.

For the experiments in this thesis a value of 1 has been chosen as the edit distance threshold in the weight generation step. When the results are evaluated on a test corpus, varying values of 1 to 3 are used for the maximum edit distance.

During the normalisation process, each word in the historical corpus is looked up in the dictionary. If it is present in the modern dictionary the word is skipped. Otherwise, the word form is compared to every word in the dictionary in terms of edit distance. If no candidates are found within the maximum edit distance the word is again skipped. Otherwise, if a single candidate is found, it is used for the normalisation. If multiple candidates are found the one with the lowest edit distance is selected. If multiple candidates exist within the same edit distance, the one with the highest frequency in the modern corpus is selected. Finally, if none of the candidate words can be found in the modern corpus, one of the candidate word forms is selected at random.

When performing Levenshtein-based spelling normalisation, if the algorithm cannot find a normalisation candidate within the given edit distance threshold,
the original word is kept. This is sometimes the right choice, since the dictionaries used for look-up do not always contain the correct words forms, especially in the case of proper names. If the maximum edit distance is too small, the correct candidates can be missed if they are too far away from the original word in terms of edit distance. As an example, during the normalisation experiments the historical Swedish word form *sielfwe* (-selves, as in themselves) was not able to be normalised into its modern equivalent *själva*, because the word forms were too far away from each other in terms of edit distance. On the other hand, if the threshold is set too high, the original word forms are occasionally wrongly changed when they should be kept. As an example, the English word form *seneschal* (steward) was wrongly normalised into *sensual* during the normalisation experiment, when it should have been kept unchanged.

When the first step of normalisation without weights is done, parts of the training corpus will have been normalised. However, since the edit distance threshold is low, many historical word forms will likely have been left as is, since no normalisation candidates could be found within the maximum edit distance.

From the parts which were normalised, the pairs of historical word forms and their normalised equivalents are extracted. These are then effectively treated as the aligned training data is in section 2.5. By comparing each historical word form and modern word form in the training pairs using another instance of the edit distance algorithm, the operations needed to transform the former into the latter are recorded. As an example, one common edit when normalising historical Swedish is the deletion of *h*. This occurs for instance when normalising the historical word form *ahnklagadhe* (accused) into its modern equivalent *anklagade*.

As in the supervised algorithm, only those edits which occur 50 times or more are used, in order to reduce noise. Different values for this parameter were tried, including 10 and 25, with the hope that more useful weights could be extracted with a lower threshold value. However, the original parameter of 50 produced the best results and was the one used in the final experiments.

After the edit operations are recorded, the weights are calculated. The weight of each edit is set to the result of dividing the frequency of a character being

---

1Recording the edits could theoretically have been done in the normalisation step. However, this way was chosen due to practical reasons, since it meant Pettersson’s scripts could be used.
left unchanged with the overall frequency of the character, as in section 2.5. If, for instance, the deletion of h had occurred 750 times while the character had been left unchanged 500 times, the weight for that edit would have been set to $500 / (500 + 750) = 0.4$. The edit distance in the example of \textit{ahnklagadhe}→\textit{anklagade} would then be $0.4 \times 2 = 0.8$ instead of the normal cost of $1 \times 2 = 2$, making the correct normalisation cheaper and therefore more likely to be selected.

Finally, when the weights have been calculated, the normalisation process is run again with the generated weights, but this time on the test corpus. Ideally, the normalisations performed by the first run of the algorithm are correct and useful edit weights can be captured from them to improve the normalisation accuracy, as in the example above.

### 3.2.1 Additional Experiments

In order to decrease the chance that incorrect normalisations are performed, Hauser and Schulz (2007) only select normalisation candidates that match a single word in the lexicon. This rule was also tried for the experiments in this thesis. However, this resulted in too few training pairs to create weights from and reduced normalisation accuracy. The rule was therefore removed for the final experiments and not present for the results presented in chapter 4.

The safeguard against using unrelated pairs of words as training data is instead – as described in chapter 2 – to pick the candidate with the highest frequency in a modern corpus when several candidates are found within the same edit distance from the historical word.

In addition, an additional component of adding the start and end tags, \texttt{<s>} and \texttt{</s>} to all words was tried, with the aim to increase the contextual information contained in the edit operations and thus the normalisation accuracy. This did not, however, result in any increased accuracy, but instead decreased it slightly. This component was therefore removed in the final tests.
4 Results and discussion

This chapter begins with a description of the units of measure used in the thesis. This is followed by an account of the main varying parameter in the experiments, the edit distance threshold. Following that, the results are presented and discussed.

4.1 Units of Measure

The results of the experiments are measured in two ways. Normalisation accuracy measures the percentage of tokens which are identical to their manually normalised equivalents. The character error rate (CER) is a more granular measurement which measures the differences between the normalised text and the gold standard at a character level. It is defined as the quotient between the number of character-level errors and the total number of characters:

\[
CER = \frac{\text{Character error count}}{\text{Character count}}
\]

4.2 Edit Distance Threshold

The edit distance threshold, or maximum edit distance allowed, has a large effect on the results. In Pettersson (2016), the author used a tuning corpus in order to set the maximum edit distance allowed between the original word forms and the normalisation candidates. This was achieved via the following formula to give a 95% confidence interval:

\[
\text{Threshold} = \text{Average edit distance} + (1.96 \times \text{Standard deviation})
\]

Average edit distance refers in this case to the average of the edit distance used to go from historical word form to the manually corrected modern word form in the training data. This calculation was specific for each language. Since this thesis is about exploring the possibility of unsupervised and language independent generation of edit distance weights, no manually normalised training or tuning data are used and neither are the precalculated threshold values of Pettersson (2016). The normalisation experiments are instead run with values from 1 to 3 for the edit distance threshold, and the results for different values of the parameter are presented.

4.3 Results and discussion

The highest results in the tables are marked with shaded cells. The baseline normalisation accuracy, meaning the percentage of tokens which are already identical to the gold standard, can be found in table 4.1.
As can be seen in table 4.1 and table 4.2, the normalisation accuracy increases with automatically extracted weights for English, Hungarian, and Icelandic. For Swedish the best normalisation accuracy, which is found when the maximum edit distance is set to 2, remains the same at 75.1 both with and without weights. The CER does increase slightly with the addition of unsupervised weights, from 0.27 to 0.28. When it comes to German both normalisation accuracy and CER show worse results with the addition of weights no matter the maximum edit distance set.  

4.3.1 Results Compared to Normalisation without Weights

4.3.2 Results Compared to Supervised Algorithm

Oddly enough, the unsupervised algorithm seems to perform slightly better than the supervised one for English when the edit distance thresholds are set to 2 and 3. Similarly, the highest normalisation accuracy for Hungarian is shown at an edit distance threshold of 3 for the unsupervised algorithm. These results are

---

1 Although the CER appears to be the same when the maximum edit distance is set to 2, that is due to rounding and the weighted algorithm does perform slightly worse in that case as well.
counter-intuitive since the lack of real training data should cause the unsupervised algorithm to perform worse. Perhaps the precalculated edit distance thresholds for those languages, used in the supervised algorithm, were not optimal for maximising normalisation accuracy. Another reason could be the weights themselves. It could be that not all the edit weights extracted from the aligned data were useful. If the test data is sufficiently different from the training data the weights could hurt more than they help.

There is also an element of chance in the normalisation which has an influence in the final normalisation accuracy. Candidates with the same edit distance from the original word form are selected at random if it cannot be determined which one occurs the most in a modern corpus.

It is important to note that the differences in normalisation accuracy between the algorithms are rather small when it comes to the actual number of correct normalisations being made. When it comes to German the margin is in the realm of 10s rather than 100s or 1000s, since the test corpus is relatively small. One should therefore be a bit cautious when drawing conclusion from the relative differences in percentage points.

### 4.3.3 Performance on German and Swedish

The normalisation with unsupervised weights shows the worst performance on German, where both the normalisation accuracy and CER worsen slightly as the weights are added. The performance on Swedish is also relatively poor.

Why does the algorithm perform poorly on those languages? One reason could be the relatively low amount of training data available for German and Swedish. By looking at table 3.1 it can be seen that Swedish and German are the two languages with the least amount of training data, both in terms of tokens and types.

The low amount of training data is shown in that the unsupervised algorithm had trouble extracting weights from the German and Swedish data sets. Only weights for 16 different edits could be extracted for German and 37 for Swedish. In comparison, 139 different weights were extracted for Hungarian, 78 for Icelandic and finally 92 for English. As described in section 3.2, 50 or more of each edit need to be observed among the training pairs before the recorded edit is used as a weight. This threshold exists in order to reduce noise. However, even with this threshold, and even considering the low amount of edit distance weights actually

<table>
<thead>
<tr>
<th></th>
<th>Acc</th>
<th>CER</th>
<th>Th</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>82.9</td>
<td>0.19</td>
<td>2.8</td>
</tr>
<tr>
<td>German</td>
<td>87.3</td>
<td>0.13</td>
<td>1.6</td>
</tr>
<tr>
<td>Hungarian</td>
<td>31.7</td>
<td>0.71</td>
<td>2.4</td>
</tr>
<tr>
<td>Icelandic</td>
<td>67.3</td>
<td>0.35</td>
<td>2.2</td>
</tr>
<tr>
<td>Swedish</td>
<td>79.4</td>
<td>0.22</td>
<td>2.7</td>
</tr>
</tbody>
</table>

Table 4.3: Results for supervised weighted edit distance with precalculated thresholds (Pettersson 2016). The results are given in normalisation accuracy (Acc) and character error rate (CER). The final column shows the edit distance threshold used.
extracted, the weights for German are apparently “noisy” enough to hinder the normalisation rather than help it.

As described in chapter 3, the process of extracting the unsupervised weights involves an approximate search in a lexicon. The algorithm tries to find training pairs by looking up word forms within an edit distance of 1 compared to the historical word forms. The model relies on correct training pairs being found in this step so that useful edits can be captured and weights extracted. However, there is no guarantee for how many normalisations performed in this step are actually correct. When looking closer at how the algorithm performed for German, it becomes apparent that of the normalisations performed in this first step, only 22% were correct. This can be compared to English where the same number was 48%. There is no wonder that the extracted weights hinder more than they help in such circumstances.

Another contributing factor to the poor performance for German is likely that the language already had the highest baseline accuracy at 84.7 percent. With such a high percentage of tokens already being identical to the gold standard, improving the accuracy further becomes difficult. The attempted normalisation needs a high precision to avoid making things worse instead of better.

This is borne out by the fact that the languages which show the most improvement in normalisation accuracy over the baseline are those with the lowest baseline accuracy, namely Hungarian and Icelandic.

4.3.4 Types of Errors

The dictionary resource plays a crucial part in any normalisation based on Levenshtein distance. If the right normalisation candidate is not present in the dictionary, a correct normalisation is impossible. If the dictionary resource used is “noisy” and contains unwanted tokens it can also have an adverse effect on normalisation accuracy.

As an example, during the Swedish normalisation the common preposition til (to) was normalised into the unrelated abbreviation tl (thallium) rather than the modern spelling till. This was the fourth most common error made for the Swedish data set, with this error being made 184 times. A correct normalisation of til would have increased the normalisation accuracy for Swedish with about half a percentage point. Not an insignificant figure considering the small margins involved.

Similarly, the English historical word form haue (have) is not normalised because it occurs in the British National Corpus which is used as dictionary resource for that language. It is found in a sonnet written in Early Modern English. This error is the fifth most common for the English normalisation.

Naturally, word forms missing from the dictionary resource also cause errors. As an example, the word schmelzt (melts) is consistently normalised as schmerzt (hurts), even though it should not have been changed at all. This is due to the fact that the Parole corpus which is used as a dictionary resource for German does not contain the word schmelzt.

Finally, mistakes in normalisation can also be made because the closest candidate in terms of edit distance is not the correct choice. The whole idea of using edit
distance for normalisation rests upon the intuition that the most similar word form to the historical one is likely to be the correct one. That is not always the case. Examples include *yowe* (you) being normalised to *owe* and *wyt* (with) being normalised to *wit*. Both errors are caused by the correct modern word form being further away in terms of edit distance than another, incorrect candidate. It can also be the case an incorrect candidate and a correct candidate have the same edit distance from the historical word form. As an example, *knowe* should be normalised into *know* rather than *known*. However, the edit distance to both is the same. These types of errors are the only ones that the addition of weights can help correct.
5 Conclusion and Future Work

This chapter presents conclusions drawn from the results of the thesis as well as suggestions for future work.

5.1 Conclusion

The aim of this thesis was to perform a multilingual evaluation of an unsupervised algorithm for weighted Levenshtein edits. The results were to be compared both to a normalisation method based on standard Levenshtein distance without weights, and to a method based on weights extracted from manually normalised parallel corpora. Another aim was to test the unsupervised algorithm in terms of generalisability.

The results of the evaluation have been mixed. The unsupervised normalisation method was, as expected, generally outperformed by the supervised method. However, the increase in normalisation accuracy for English and Hungarian was both interesting and unexpected. Whether the increase was due to the different edit distance thresholds used, suboptimal weights in the supervised weights file, or some minor differences in implementation is difficult to say.

The unsupervised algorithm performs better than, or similar to, unweighted Levenshtein distance for all languages except German. When investigating this phenomenon, it was found that only 22% of the normalisations from which the unsupervised weights were extracted, were in fact correct. It is therefore not strange that the unsupervised weights hinder more than they help. In order for useful weights to be extracted the algorithm must first be able to perform correct normalisations without weights and within a maximum edit distance of 1.

Using the unsupervised method for generating weights in this thesis seems to be generally worthwhile if aligned training data is not available and one would otherwise be normalising using edit distance without weights. However, an increase in normalisation accuracy is not guaranteed, and depending on the data set and language resources involved, it may even decrease slightly.

The unsupervised algorithm requires more training data than the supervised one, since only the fraction of the historical tokens for which a likely modern match can be found are used. On the other hand, a parallel corpus is not needed, and there is no need for manual normalisation of the training data. The training data consist of historical text in its original form. It should therefore be easier to gather a larger amount of training material.

The dictionary resource plays a very large part in the results of the normalisation, and unwanted word forms in the dictionaries caused a large amount of errors during the evaluation.

The results achieved in this thesis have been mixed enough that they cannot be said to be generalisable across the languages and data sets. With such a high
variance in the results, with the algorithm performing better than the supervised one for two data sets and worse than standard edit distance for another, it is clear that further testing is needed. However, while the results have been varied the method itself is language independent. The author believes that the results could be improved significantly with larger training corpora and by adapting the dictionary resources more to the normalisation task.

5.2 Future Work

Further multilingual evaluations of historical spelling normalisation would be interesting to perform. One important parameter for normalisation could not be set in an automated, unsupervised way in this thesis: the maximum edit distance. Since this parameter has such a large effect on the results it would be worthwhile to come up with an unsupervised method for generating it.

It would also be interesting to use other types of edit distance and different string similarity measures for unsupervised normalisation. Since historical spellings variations are phonologically similar, a similarity measure that takes phonetics into account, such as Editex (Zobel and Dart 1996) could prove fruitful.

Finally, another idea to explore is to iterate the steps of extracting edit distance weights and performing normalisation. Perhaps the normalisation results could be improved after several iterations as the weights were refined, similar to how the Expectation-Maximisation (Dempster et al. 1977) algorithm works.
Bibliography


