Spelling Normalization of English Student Writings

Yuchan HONG

Uppsala University
Department of Linguistics and Philology
Master’s Programme in Language Technology
Master’s Thesis in Language Technology
September 27, 2018

Supervisor:
Eva Pettersson, Uppsala University
Abstract

Spelling normalization is the task to normalize non-standard words into standard words in texts, resulting in a decrease in out-of-vocabulary (OOV) words in texts for natural language processing (NLP) tasks such as information retrieval, machine translation, and opinion mining, improving the performance of various NLP applications on normalized texts.

In this thesis, we explore different methods for spelling normalization of English student writings including traditional Levenshtein edit distance comparison, phonetic similarity comparison, character-based Statistical Machine Translation (SMT) and character-based Neural Machine Translation (NMT) methods. An important improvement of our implementation is that we develop an approach combining Levenshtein edit distance and phonetic similarity methods with added components of frequency count and compound splitting and it is evaluated as a best approach with 0.329% accuracy improvement and 63.63% error reduction on the original unnormalized test set.
6 Conclusion and Future Work

6.1 Conclusion .............................................................................. 29
6.2 Future Work ............................................................................. 30
Acknowledgements

First of all, I would like to express my very great appreciation to my supervisor Dr. Eva Pettersson, an excellent supervisor and a kind person who gave me many constructive suggestions and advice with inspiration and kind support in experimental data and tools leading me to the right direction of my experiments, as well as her detailed and valuable comments on my thesis. I would also like to offer my special thanks to Dr. Beáta Megyesi for helpful advice. I would also like to thank all the faculty in the master program of language technology of the Department of Linguistics and Philology for their warm guidance and assistance. Last but not least, I would like to thank my mother, family and friends for encouraging and helping me through my studies in Sweden.
1 Introduction

Spelling normalization is the task to normalize non-standard words into standard words in texts. It gains wide attention in natural language processing (NLP) as an effective preprocessing method to improve the performance of NLP tasks such as machine translation, data mining and information retrieval. Due to typing and cognitive errors, English student writings contain many non-standard words, mostly typos and misspellings, which needs to be normalized before further linguistic analysis and other NLP tasks.

1.1 Purpose

This thesis is inspired by SWEGRAM \(^1\) (Näsman et al., 2017) which is a web-based tool for automatic linguistic analysis of Swedish text, with a special focus on student writings, by providing a pipeline including tokenization, spelling normalization, part-of-speech tagging and syntactic analysis. The easy-to-use and user-friendly SWEGRAM is built to enable researchers with no computer skill to freely annotate their data and to do linguistic analysis and get frequency statistics from their texts. The aim of this thesis is to explore a spelling normalization approach for English student writings, which could be used as a preprocessing step in an English version of SWEGRAM. To focus on spelling normalization and expecting this step would improve the accuracy of latter linguistic analysis, we propose these research questions:

- How well do different approaches for spelling normalization perform for the task of normalizing English student writings?
- Could we improve the performance by adding frequency count and compound splitting?

1.2 Outline

In order to answer these research questions and conduct experiments accordingly, we present our work in the following parts.

In Chapter 2 we introduce spelling normalization and its use for historical texts and modern noisy texts as well as common methods for spelling normalization and related work.

In Chapter 3 we describe the data and tools used for spelling normalization in different steps in our experiments. In this part, we explain how the data are

\(^1\)http://stp.lingfil.uu.se/swegram/
chosen and used for different steps and give an overview of the tools used.

In Chapter 4 we present the methodology and experimental setup for different spelling normalization methods.

In Chapter 5 we present the experimental results for different spelling normalization methods, and explain and discuss the results in detail.

In Chapter 6 we conclude our findings and give possible suggestions for future work.
2 Background

2.1 Spelling Normalization

Spelling normalization is the task to normalize non-standard words into standard ones. Non-standard words occur in many text genres such as advertisements, SMS, texts on social media sites, historical texts and student writings. These non-standard words sometimes lead to misunderstandings for readers who are from different cultures or have another historical background. Moreover, these non-standard words have a negative impact on the performance of natural language processing (NLP) tools such as information retrieval and data mining (B. Han et al., 2013; Kong et al., 2014; Plank et al., 2014). In order to eliminate the negative impact, researchers investigate different methods for spelling normalization in various text genres.

2.1.1 Spelling Normalization of Historical Texts

With the development of digital humanities and an increasing need for digitalized historical documents, historical texts have gained large attention. However, it differs from modern data in that there are no standard writing conventions in spelling in historical texts. Spellings change not only in texts from different time periods, but also sometimes in texts from the same time period written by different authors due to dialect influence (Bollmann et al., 2011). The example below demonstrates spelling variation in 14th century English. The spelling variation is a challenge for NLP applications to process and generate annotated historical data and automatic data analysis. For instance, a tagger trained on 14th century English may not perform as well on 18th century English texts. A common approach to tackle this problem is to apply spelling normalization (Piotrowski, 2012) that is to map spelling variations in historical texts to modern standard forms.

<table>
<thead>
<tr>
<th>Original Sentences</th>
<th>Modern Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whan that Aprill,</td>
<td>When [that] April</td>
</tr>
<tr>
<td>with his shoures soote</td>
<td>with his showers sweet</td>
</tr>
<tr>
<td>The droghte of March</td>
<td>The drought of March</td>
</tr>
<tr>
<td>hath perced to the roote</td>
<td>has pierced to the root</td>
</tr>
<tr>
<td>And bathed every veyne</td>
<td>And bathed every vein</td>
</tr>
<tr>
<td>in swich licour,</td>
<td>in such liquor,</td>
</tr>
<tr>
<td>Of which vertu engendred</td>
<td>Of which virtue engendered</td>
</tr>
<tr>
<td>is the flour;</td>
<td>is the flower;</td>
</tr>
</tbody>
</table>

Table 2.1: First four lines of the General Prologue from The Canterbury Tales by Geoffrey Chaucer and their modern word-to-word translation. (Carlson, 2004)
Spelling normalization of historical texts has been explored and approached using various methods such as dictionary lookup, edit distance comparison, phonetic similarity comparison, machine translation and deep learning. Rayson et al. (2005) presented VARD (VARiant Detector) which contains a manual mapping scheme of spellings from 16th to 19th century English texts to modern spellings. The performance of VARD was compared with the MS-Word and Aspell checker evaluated on a test set of 17th century English texts and it was reported that VARD outperforms the spell checker by achieving one third more normalized tokens. Bollmann et al. (2011) tried a weighted Levenshtein-based normalization approach for Early New High German which are derived from two aligned version of the Martin Luther Bible in its 1545 and 1892 versions and showed a 91% exact match by this approach compared to 65% exact match of the old bible text before normalization. Pettersson, Megyesi, and Nivre (2013) explored a Levenshtein-based approach using context-sensitive, weighted edit distance combined with compound splitting. This approach was evaluated for Early Modern Swedish and showed a normalization accuracy of 86.9%, clearly outperforming the baseline of 64.6% for the unnormalized version of the evaluation corpus. Pettersson, Megyesi, and Tiedemann (2013) tried an SMT approach to normalizing historical texts, and achieved an increase in normalization accuracy from 64.8% to 83.9% for historical Icelandic texts, and from 64.6% to 92.3% for historical Swedish texts. Bollmann and Søgaard (2016) explored a neural network architecture of deep biLSTM network on character-level for normalization and evaluated their method on Early New High German showing an increased accuracy compared to the baseline of unnormalized texts and an average accuracy of 80.55%.

2.1.2 Spelling Normalization of Noisy Modern Texts

User-generated texts are common and widespread nowadays in social media sites and blogs like Twitter and Facebook. Due to user creativity and Internet trends, noisy modern texts are generated consisting of non-standard casual language breaking the ordinary rules for spelling, grammar and pronunciation. According to Clark and Araki (2011), spelling errors and irregular language in social media could be grouped into eight categories, as illustrated in Table 2.2.

<table>
<thead>
<tr>
<th>Error Categories</th>
<th>Error Examples</th>
<th>Normalized Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviation (shortform)</td>
<td>nite, sayin, gr8</td>
<td>night, saying, great</td>
</tr>
<tr>
<td>Abbreviation (acronym)</td>
<td>lol</td>
<td>laugh out loud</td>
</tr>
<tr>
<td>Typing error/misspelling</td>
<td>wouls, ridiculous</td>
<td>would, ridiculous</td>
</tr>
<tr>
<td>Punctuation omission</td>
<td>im , dont</td>
<td>I’m, don’t</td>
</tr>
<tr>
<td>Non-dictionary slang</td>
<td>that was well mint</td>
<td>That was very good</td>
</tr>
<tr>
<td>Wordplay</td>
<td>that was soooooo great</td>
<td>that was so great</td>
</tr>
<tr>
<td>Censor avoidance</td>
<td>sh1t, f***</td>
<td>shit, fuck</td>
</tr>
<tr>
<td>Emoticons</td>
<td>:) , &lt;3</td>
<td>smiling face, heart</td>
</tr>
</tbody>
</table>

Table 2.2: Noisy Modern Texts Categories and Examples

Spelling normalization of noisy modern texts is important for tasks such as information retrieval, machine translation and opinion mining. Studies on spelling normalization of noisy texts are commonly focusing on preprocessing
approaches. Clark and Araki (2011) explored Casual English Conversion System (CECS) as a high-quality and user educational preprocessing system for normalizing noisy social media texts. They manually compiled a database of 1,043 phrase and word entries for noisy items lookup and an educational constraint of non-natives’ comprehension of informal written English is added in the system to furthermore achieve a linguistic and cultural study. To evaluate CECS, a test set of 100 sentences from Twitter were translated to Japanese using Google Translate and Systran. Before and after CECS preprocessing, the proportion of non-translated words decreased from an average of 3.34 to 0.86. Aw et al. (2006) proposed a phrase-based statistical model for SMS normalization and discussed the difference between word-based SMT and phrase-based SMT. 5000 SMS were evaluated and showed a BLEU score of 0.80720 for the model which outperforms the baseline BLEU of 0.5784 for raw SMS without normalization. Baldwin et al. (2015) explored deep learning methods for Twitter lexical normalization and named entity recognition and found that deep learning methods and methods based on lexicon-augmented conditional random fields (CRFs) achieved best results by a F1 score of 0.8421 evaluated on randomly sampled tweets from December 2014 to February 2015.

2.1.3 Spelling Normalization of English Student Writings

English as a world language is widely spread and learned throughout the world and students who study English as their second language usually encounter problems of misspellings and grammar errors. Using NLP-based models could be a valuable tool for English teaching and studying in ESL (English as a Second Language) and EFL (English as a Foreign Language) since it helps to prompt feedback in English acquisition such as spelling errors, grammatical errors and writing quality (N.-R. Han et al., 2010). However, these models tend to be trained on data from native English speakers with well-formed and correct texts. N.-R. Han et al. (2010) presented a manually error-annotated English learner corpus from Chungdahm English Learner Corpus of native Korean students and explored the preposition error detection and correction using a maximum-entropy-based model and a language model trained on the error-annotated learner corpus and a corpus of well-written and constructed texts by native English people respectively. They found that the same model trained on an error-annotated corpus produced by English learners outperformed the native corpus on a large scale even if the error-annotated corpus is five times smaller than the native corpus. Okada (2005) found that Japanese EFL learners have spelling errors occurring in word-initial and word-final position based on spelling error corpora. Although spell checkers are well developed and widely used, spelling error annotated corpora for ESL/EFL teaching and learning are not common, even though they are needed and beneficial in the process of second language acquisition.

2.2 Common Methods for Spelling Normalization

Spelling errors are classified into two categories by Kukich (1992) as typographic and cognitive errors. Typographic errors are mainly caused by careless typing on the keyboard which appears for example as a substitution or transposition
of letters in a word, such as wrongly typed 'what' to 'waht'. Cognitive errors on the other hand, occur when a person does not know the right spelling of a word, thus spelling it according to the word’s pronunciation. Therefore these kinds of errors occur mostly between words having high phonetic similarity, such as 'whether' with its misspelling 'wether'. A fundamental model for the spelling normalization task is the noisy channel model (Shannon, 1948), which consist of two channels, one source model and one channel model, and the two formulas could be illustrated as Formula1 and Formula2.

\[ \text{Formula1} : \text{argmax}_w P(w|s) \]

\[ \text{Formula2} : \text{argmax}_w P(s|w) * P(w) \]

Given an alphabet E, a string s composed of letters in E, and a dictionary of D. The s is the misspelling of a string w, in which s is not present in D and w is present in D. To find the best w for s, we need to calculate the formula in Formula2, so we have a source model P(w) and a channel model P(s | w) for the noisy channel model in the spelling normalization task. Under this model, we would expect the probability for "what" given "what" (P(what|what)) to be very high, and the probability for "what" given "waht" (P(what|waht)) to be relatively high, and the probability for "what" given "which" (P(what|which)) to be low. Levenshtein edit distance and phonetic similarity methods are posed on the noisy channel model to deal with the spelling normalization problem, besides, machine translation and deep learning methods are also claimed to have a positive impact on the spelling normalization task.

2.2.1 Levenshtein Edit Distance Method

Levenshtein edit distance (Levenshtein, 1966) is a string metric to calculate the minimum letter edit steps between two strings which originally allows insertion, deletion and substitution of letters, whereas the Damerau-Levenshtein distance (Damerau, 1964) also allows transposition of two adjacent letter. Examples could be illustrated as:

'cloud' -> 'coid' (1 substitution of 'l' for 'o')
'too' -> 'to' (1 deletion of 'o')
'had' -> 'have' (1 substitution of 'v' for 'd' and 1 insertion of 'e')
'waht' -> 'what' (1 transposition of 'a' and 'h')

The mathematical formula of the Levenshtein edit distance could be illustrated in Figure 2.3 where dist refers to the edit distance, i refers to the ith letter in the former string, and j refers to the jth letter in the later string.

Damerau (1964) presented that up to 80% of the spelling corrections based on Levenshtein edit distance could be solved within an edit distance of one and
\[ 
\begin{align*}
\text{dist}(0, 0) &= 0 \\
\text{dist}(i, 0) &= i \\
\text{dist}(0, j) &= j \\
\text{dist}(i, j) &= \min \left\{ 
\begin{array}{l}
\text{dist}(i-1, j) + 1 & \text{deletion} \\
\text{dist}(i, j-1) + 1 & \text{insertion} \\
\text{dist}(i-1, j-1) + \begin{cases} 0 & \text{if } i = j \\ 1 & \text{otherwise} \end{cases} & \text{equality} \\
\end{array} \right. 
\right. 
\end{align*} 
\]

Figure 2.1: Levenshtein Edit Distance (Pettersson, 2016)

many early algorithms and studies on Levenshtein-based spelling normalization are based on this assumption. Later improvements are achieved by estimating probabilities and weights for different edit operations and allow multiple edit operations. Pettersson, Megyesi, and Nivre (2013) presented a weighted Levenshtein method with single-character weights and context-sensitive weights calculated by comparing the frequency of each edit and the frequency of the specific source characters left unchanged. Normalization candidates which have more than one edit distance compared to the source string are taken into consideration, and with the context-sensitive weights, this approach achieved 79.1% accuracy compared with the baseline of 64.6% on unnormalized texts evaluated on 600 sentences evenly and randomly chosen from 12 historical Swedish court and church records.

2.2.2 Phonetic Similarity Comparison Method

Phonetic similarity comparison methods are mostly implemented based on some phonetic algorithms which create phonetic representations for words. Some examples of phonetic algorithms and how the word form ‘spell’ is represented for each algorithm are given in Table 2.3.

<table>
<thead>
<tr>
<th>Phonetic Algorithms</th>
<th>Representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoundEx (Odell, 1956)</td>
<td>‘s104’</td>
</tr>
<tr>
<td>Metaphone (Philips, 1990)</td>
<td>‘SA’</td>
</tr>
<tr>
<td>NYSIIS (Rajkovic and Jankovic, 2007)</td>
<td>‘SPL’</td>
</tr>
<tr>
<td>New York State Identification and Intelligence System</td>
<td>(‘SPL’, ‘)’)</td>
</tr>
<tr>
<td>DoubleMetaphone (Philips, 2000)</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3: Example of phonetic algorithms representations for word ‘spell’.

The idea is that words with the same pronunciation would share the same representation based on one phonetic algorithm. Toutanova and Moore (2002) presented an approach to combine phonetic information with the noisy channel model which resulted in a best error reduction of 46% tested on a set of around 10,000 misspellings. To deal with misspellings due to cognitive errors, to choose normalization candidates with the same phonetic representation as the misspellings would be a feasible idea.
2.2.3 SMT and NMT Methods

Statistical machine translation (SMT) is a method driven by parallel linguistic data to find the most probable word or phrase in a target language for a given word or phrase in the source language. SMT-based methods in spelling normalization regard spelling normalization as a translation task with the misspellings and typos as source words and normalized words as target words. However, this method requires large parallel data for training and tuning, and in the spelling normalization task, it means that large amounts of misspellings and their corresponding normalized forms are needed to improve the performance of SMT normalization. Pettersson, Megyesi, and Tiedemann (2013) proposed a character-based SMT approach which regards the translation task at a character level rather than at a word level, using character-aligned parallel data for training and tuning. The results showed that with a set of 1,000 Icelandic historical and modern spelling pairs, the normalization accuracy is 76.5% compared with a set of 33,888 token pairs with a normalization accuracy of 83.9%. The character-based SMT is thus proved to achieve comparatively good results with much less parallel data than would be needed for traditional SMT tasks.

With explosive research on deep learning, neural networks have been introduced to the spelling normalization field as well. Korchagina (2017) presented a comparative study on rule-based translation, character-based SMT, and character-based neural machine translation (NMT) on spelling normalization of 15th to 16th century German texts, and found that NMT showed the best accuracy and that the accuracy trend to be higher with more parallel data. Bollmann et al. (2017) presented several encoder-decoder architectures including recurrent neural networks (RNN) and long short-term memory (LSTM) and a multi-task learning (MTL) architecture to realize learning pronunciation and achieved a 2% increase in performance for spelling normalization of Early New High German texts.

2.2.4 Compound Splitting

Compounds are common in many languages especially Germanic languages such as German, Swedish and Danish, NLP tasks such as machine translation and spelling normalization are reported to have worse performance due to compounds. Previous studies on translation between German and English have presented this problem (Koehn and Knight, 2003; Popović et al., 2006), and Stymne (2008) presented that phrase-based SMT with compound splitting as a preprocessing step improved the translation from Swedish to English with a reduction of untranslated words by 50%. Compound splitting has a positive impact on spelling normalization as well. Pettersson, Megyesi, and Nivre (2013) presented a historical normalization task with a compound splitter developed by Stymne (2008) which split the compounds into their compound parts and search for the normalized spelling of each part of the compound. Although it showed no change on the automatic accuracy, it detected a small increase in the amount of correctly normalized words when using the compound splitter.
2.3 SWEGRAM

SWEGRAM (Näsman et al., 2017) is a web-based tool for automatic annotation and quantitative analysis of Swedish achieved by a pipeline consisting of tokenization, sentence segmentation, spelling normalization, POS tagging, and syntactic analysis as dependency parsing. It is easy-to-use and user-friendly which allows users to upload one or more files, and then an annotated output file in CoNLL-U tab-separated format\(^1\) is generated presenting annotated data including information of tokens, normalized tokens, lemma, part-of-speech (POS) tags and dependency parsing symbols. It also provides quantitative analysis that automatically analyse the number of tokens, lemmas, sentences, POS, and spelling errors, and provide readability measures, and information on the average length of various units. It is an open source tool available for all individuals especially beneficial to researchers in humanities and social sciences for annotating and generating their own annotated data.

\(^1\)http://universaldependencies.org/format.html
3 Data and Tools

In order to explore various spelling normalization methods for English students' writings and select our best performing normalization method, five different approaches are implemented:

- Damerau Levenshtein Edit Distance Comparison
- Metaphone Similarity Comparison
- The Combination of Damerau Levenshtein and Metaphone Comparison with or without Frequency Count and Compound Splitting
- A Character-based SMT Approach using GIZA++ toolkit \(^1\) (Koehn et al., 2007)
- A Character-based NMT Approach using seq2seq \(^2\) (Britz et al., 2017)

Three corpora are used:

- Uppsala Student English Corpus (USE) \(^3\)
- WordNet \(^4\) (Miller, 1995)
- A Subset of the British National Corpus (BNC) \(^5\) (Consortium et al., n.d.)

These corpora are involved in relevant steps of our spelling normalization approaches. A subset of USE was manually corrected by us with regard to spelling and is used as training and tuning data for the character-based SMT and NMT approaches, and as test data for all spelling normalization approaches. A lookup dictionary generated from WordNet is used for detecting tokens which are misspellings and typos (or refers to out-of-vocabulary words). Finally a subset of BNC is added to further enlarge the detection dictionary and to be used for frequency counts. Detailed information about data and tools are explained in the following part.

\(^1\)http://www.statmt.org/moses/

\(^2\)https://google.github.io/seq2seq/

\(^3\)http://www.engelska.uu.se/research/english-language/electronic-resources/use/

\(^4\)https://wordnet.princeton.edu/

\(^5\)http://stp.lingfil.uu.se/ evapet/dighum/hist-datasets.html
3.1 Data

3.1.1 WordNet

WordNet is a large lexical database for English which is freely and publicly available to download online. It contains more than 118,000 different word forms and more than 90,000 word senses, divided according to the syntactic categories of noun, verb, adjective, and adverb (Miller, 1995). To generate a lookup dictionary for detecting out-of-vocabulary (OOV) words in original students' writings, only word terms from WordNet are kept and the word sense part is removed.

3.1.2 British National Corpus (BNC)

BNC is a written and spoken language corpus representing a wide cross-section of British English from late 20th century containing 100 million words. A web-interface version of BNC is available for free access containing basic search queries for words and collocation searching. Since BNC contains large amounts of copyright material, to use the offline version of BNC, users have to agree to the terms of Licence\(^6\). A subset of BNC containing approximately 2 million words is adapted in this thesis for further building the OOV lookup dictionary and collecting frequency statistics for a frequency count.

3.1.3 Uppsala Student English Corpus (USE)

In order to design and test different approaches for spelling normalization of English student writings, an English student writings collection and its normalized gold standard collection are needed. USE is an English writing collection of Swedish university students ranging from the years 1999-2001 set up by Ylva Berglund and Margareta Westergren Axelsson aimed at conducting research into the process and results of English learning and acquisition. The corpus consists of 1,489 essays of 1,221,265 words written by 440 Swedish university students of English from three different levels covering different topics. Three different levels of 'a', 'b', 'c' stands for first-term, second-term and third-term, and different topics are collected as 'a1' to 'a5', 'b1' to 'b8' and 'c1'. A subset of USE is selected and collected for this thesis as illustrated in Table 3.1.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Numbers of Essays</th>
<th>Numbers of Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set (a1)</td>
<td>111</td>
<td>93,694</td>
</tr>
<tr>
<td>Tuning set (a2)</td>
<td>58</td>
<td>50,161</td>
</tr>
<tr>
<td>Testing set (a3)</td>
<td>14</td>
<td>11,407</td>
</tr>
</tbody>
</table>

*Table 3.1: Statistics of the USE subset*

The training, tuning and test sets are first-term essays with the following topics:

**Training set (a1):**
"English, my English."-describe their experience of English and evaluate their language proficiency

\(^6\)http://www.natcorp.ox.ac.uk/docs/licence.html
Tuning set (a2):
Argumentation. Students argue for or against a statement concerning a topical issue.

Test set (a3):
Reflections. Students reflect on the medium of television and its impact on people, or on related issues of their choice.

Since a large amount of misspellings and typos occurs in first-term English student writings, only first-term essays are selected in our experiments and these three topics are chosen because they are the first three topics in the first-term essays.

However, USE does not contain the normalized version of Swedish university students English writing, so we process a manual normalization throughout the subset and produce a spelling normalized parallel version as well. The steps and principles are:
1. Reading one essay at a time and mark the misspellings and typos
2. Correct the misspellings and typos based on our general knowledge, context information and the help of an online Cambridge Dictionary\textsuperscript{7}
   e.g. correct ‘oppurtunity’ to ‘opportunity’, ‘belive’ to ‘believe’
   e.g. correct ‘with a croud of people’ to ‘with a crowd of people’ rather than wrongly correct it to ‘with a proud of people’
3. Sometimes the students write Swedish words in the essays. Leave the Swedish words unchanged, since this thesis concerns English spelling normalization
   e.g. leave ’godis’ (‘candy’ in English) as unchanged
   It might be careless writing of Swedish university students to write Swedish words in a English essay, but this occasion occur rarely.

3.2 Tools

3.2.1 Moses
Moses \textsuperscript{8} (Koehn et al., 2007) is an open source toolkit for statistical machine translation which allows users to automatically train translation models for many translation directions, in which parallel translation data are needed. In the traditional SMT translation process, it regards a whole source sentence as input and does word or phrase alignment to find the best translation into a target sentence. It is illustrated in Figure 3.1 that a Swedish sentence ‘Jag är student.’ is a source sentence and the SMT processes and translates it to an English sentence as ‘I am a student.’ To design a character-based SMT for spelling normalization of English student writings, the translation direction is from misspellings and typos to normalized corrected spellings, and the input is an original token and the system does character alignment regarding each letter as a translation unit to find the best translation into a normalized token. It is illustrated in Figure 3.2 that a misspelling ‘croud’ is character aligned and translated into a correctly normalized word as ‘crowd’.

\textsuperscript{7}https://dictionary.cambridge.org
\textsuperscript{8}http://www.statmt.org/moses/
3.2.2 Seq2seq

Seq2seq\(^9\) (Britz et al., 2017) is an open source encoder-decoder framework for Tensorflow that could be used for machine translation, summarization, conversational modeling and image captioning, documented by standard Python code and easy to build upon due to its module-based structure. For the implementation of machine translation, seq2seq works as reading an input source sentence from left to right and encoding the sequences into hidden states and then the decoder decodes them into a target translation sentence. In the spelling normalization task, we implement the character-based NMT approach so the encoder reads an input token consisting of white-spaced characters and the decoder outputs normalized tokens.

\(^9\)https://google.github.io/seq2seq/
4 Experimental Setup

Since the implementation steps for methods based on Levenshtein edit distance and phonetic similarity comparison differ from those of the character-based SMT and NMT methods, the experimental setup is divided into three main parts to explain the different steps respectively.

4.1 Edit Distance and Phonetic Similarity

The main steps for spelling normalization using the methods based on Levenshtein edit distance and phonetic similarity are illustrated in Figure 4.1 and explained further below.

![Figure 4.1: The main steps using the methods based on Levenshtein edit distance and phonetic similarity](image)

4.1.1 Building a Lookup Dictionary and Detecting OOV Words

As mentioned in Chapter 3, two English corpora (WordNet and a subset of BNC) are used to build a lookup dictionary for detecting misspellings and typos. A piece of Python code is written to read WordNet line by line and select the keyword for each line removing the explanation. Then the tokens in the subset of BNC are added into the dictionary with a frequency count. If the token is the first word of a sentence with an uppercased first letter, it is changed to its lower case version to avoid a decrease in OOV words detection due to a difference in casing. The frequency count will be introduced and explained in section 4.1.3.

After building the lookup dictionary, we process the step of detecting OOV words. For all tokens in an original English student essay, we look it up in the
dictionary, and add it into an OOV words list if it is not in the lookup dictionary to mark it as a possible misspelling for further normalization. However, there are some tokens containing digits and proper nouns, e.g. '1999', '15-years', 'Uppsala', which are not included in the dictionary so they would be added to the OOV words list for further processing, while the normalization results for these words tend to be a failure for they remain unchanged in the gold standard. Therefore, when detecting OOV words, we first check if the token contains digits or if the token is not the first word of a sentence but it contains at least one upper case letter. Otherwise we add the token to the OOV words list.

4.1.2 Generating Normalization Candidate Set

To generate normalization candidates for tokens in the OOV words list, first an algorithm based on Ratcliff and Metzener (1988) named 'the gestalt approach' is implemented using Python code with the `difflib` Python module\(^1\). The initial idea is to find the longest contiguous subsequence that contains no blank or whitespace to the left and to the right respectively, and with further development, it allows to compare a word with a list of words and return the best match or topX match with ranked similarity score. For instance, the returning results for some OOV words with `get-close-match` top8 are:

'B Brittish':

'British', 'skittish', 'rightish', 'Britishes', 'iritish', 'iritis', 'kittenish', 'grittiest'

'wahlt':

'wht', 'wat', 'wah', 'aht', 'whut', 'whit', 'what', 'whet', 'what'

'compere':

'compete', 'compare', 'comer', 'copper', 'coppered', 'cooper', 'coopered', 'computer'

It shows that the correct normalization (with blue fill textcolor and underline) is not always in the top1 position with the best similarity score, and some even in a relatively back position. Hence an improved design for selecting candidates is important and necessary.

4.1.3 Selecting Normalization Candidates

Damerau Levenshtein edit distance and Metaphone phonetic similarity comparison approaches are implemented in this part respectively and in a combined version.

Damerau Levenshtein Edit Distance Comparison

After generating normalization candidates, a candidate set for each OOV word is available for normalization selection. Damerau Levenshtein edit distance allows insertion, deletion, substitution, and transposition of two adjacent letters. The

\(^1\)https://docs.python.org/3.6/library/difflib.html
candidate set and the selection results for some OOV words with implementation of Damerau Levenshtein edit distance are:

'B Brittish'


'waht'

'wht':1, 'wat':1, 'wah':1, 'aht':1, 'whut':2, 'whit':2, 'whet':2, 'what':1

'compere'

'compete':1, 'compare':1, 'comer':2, 'copper':2, 'coppered':2, 'cooper':2, 'coopered':2, 'computer':3

The best match (matches) in blue fill textcolor and underline after processing Damerau Levenshtein edit distance are showed above. Candidates with the same edit distance occurs as well which increase the difficulty of selecting normalization candidates.

Metaphone Similarity Comparison

The Metaphone algorithm (Philips, 1990) is initially an improvement of the Soundex algorithm and it returns an intentionally approximate phonetic representation of an input word. The candidate set and the selection results for some OOV words with implementation of Metaphone similarity comparison are:

'B Brittish' -> 'BRTX':

'B Brittish': 'BRTX', 'skittish': 'SKTX', 'rightish': 'RTX', 'Brittishs': 'BRTXS', 'irtish': 'IRTX', 'iritis': 'IRTS', 'kittenish': 'KTNX', 'griottest': 'KRTST'

'waht' -> 'WT':

'wht': 'T', 'wat': 'WT', 'wah': 'W', 'aht': 'AT', 'whut': 'WT', 'whit': 'WT', 'whet': 'WT', 'what': 'WT'

'compere' -> 'KMPR':

'compete': 'KMPT', 'compare': 'KMPR', 'comer': 'KMR', 'copper': 'KPR', 'coppered': 'KPRT', 'cooper': 'KPR', 'coopered': 'KPRT', 'computer': 'KMPTR'

The best match (matches) in blue fill textcolor and underline after processing Metaphone similarity comparison are shown above. As for the Damerau-based approach described above, the case where several candidates with the same phonetic representation occur is a problem here too, increasing the difficulty in selecting the best normalization candidate.

Combination of Damerau Levenshtein and Metaphone Comparison

Since the Metaphone comparison approach shows no improvement compared to the Damerau Levenshtein edit distance approach in selecting normalization
candidates which should be limited to close to one candidate for each OOV word, but the candidate sets returned by the Damerau Levenshtein edit distance and Metaphone algorithm are disjoint, combining these two approaches might have better results as to limit the normalized candidates to a best one candidate for each OOV word.

Since cognitive errors in spelling errors take up a relatively big portion, candidates with similar phonetics are taken into consideration first. Our implementation is to assign a value of zero for a candidate which has the similar pronunciation with the original OOV word, and a value of one for a candidate which does not, then with Damerau Levenshtein edit distance comparison, update the edit distance to the value of each candidate. To be specific, for the OOV word 'Brittish', in its normalization candidate set, the candidate 'British' has the same phonetic representation of 'BRTX' as the original OOV word 'Brittish'. Accordingly, a value of zero is assigned to the candidate 'British'. In the next step, the candidate has a Damerau Levenshtein edit distance of one, so a value of one is added to zero, resulting in a final value of one for the candidate 'British'. The candidate set and the selection results for some OOV words with implementation of the combination of Damerau Levenshtein and Metaphone similarity comparison are:

'B Brittish':
'British':1 (0+1), 'skittish':3 (1+2), 'rightish':4 (1+3), 'Britishs':3 (1+2), 'Ir tish':4 (1+3), 'iritis':5 (1+4), 'kittenish':5 (1+4), 'griottiest':4 (1+3)

'waht':
'wht':2 (1+1), 'wat':1 (0+1), 'wah':2 (1+1), 'aht':2 (1+1), 'whut':2 (0+2), 'whit':2 (0+2), 'whet':2 (0+2), 'what':1 (0+1)

'compere':
'compete':2 (1+1), 'compare':1 (0+1), 'comer':3 (1+2), 'copper':3 (1+2), 'coopered':3 (1+2), 'coopered':3 (1+2), 'computer':4 (1+3)

A substantial improvement in selecting one best candidate is shown for the combination of the two approaches. Further and detailed discussion of the evaluation results are presented in Chapter 5.

Frequency Count

Since the number of selected candidates with a minimum value presented in the examples above for each OOV word are sometimes more than one, apart from randomly choosing a normalization candidate, a better solution would be to select one final normalization candidate based on the highest frequency in a corpus. To develop the frequency count component, in the process of building the lookup dictionary, each word in the lookup dictionary initially obtains a value of one, when adding words from WordNet and the subset of BNC into the lookup dictionary. For words that are already present in the lookup dictionary, the value if updated by adding one to the current frequency count.
Compound Splitting

Since compounds are common in Swedish, they are commonly occurring in USE, e.g. 'childrenlitterature', 'schoolyear'. After observation in USE, we find that the Damerau Levenshtein edit distance is mostly more than three for these compounds, since there are no corresponding normalized words in the lookup dictionary, and their candidate set thus only contains normalization candidates of their sub-words, such as 'children' and 'literature'. To implement the compound splitter, if the best match or the selected candidate has a value bigger than or equal to three, this OOV word is regarded as a compound and it is split into two parts based on the best match. An example of compound splitting for an OOV word 'schoolyear' is:

'schoolyear' -> 'school'
'school' -> 'school'
'year' -> 'year'
'schoolyear' -> 'school year'

First, a best match candidate 'school' for 'schoolyear' is selected by Damerau Levenshtein, or Metaphone algorithm, or the combination of the two approaches, and the edit distance from 'schoolyear' to 'school' is four which is bigger than three, so the OOV word 'schoolyear' is regarded as a compound. Second, 'schoolyear' is splitted into two parts based on the best match of 'school' as 'school' and 'year'. Third, the rest part 'year' is proceeded for the spelling normalization steps of generating its normalization candidate set and selecting the best normalization candidate. Finally, the OOV compound 'schoolyear' is correctly normalized and splitted into 'school year' with a whitespace between the two parts.

4.2 Character-Based SMT Approach

The parallel USE training, tuning and test data presented in Chapter 3 are plain text files with one token per line. For the character-based SMT approach, character level data are needed. Hence, the USE data sets are modified by adding whitespace between each character of the original tokens. For instance, to change 'waht' into 'w a h t'. The Moses toolkit with all its standard components are involved in this implementation. To build the character-based SMT system based on the USE data sets, first of all, a language model of character 5-grams is trained on the normalized training data set. Pettersson, Megyesi, and Tiedemann (2013) presented a language model of character 10-grams which performs well, however, in our experiment, setting a language model of character 10-grams returns a core dumped error until the language model is changed to character 5-grams. Then a training step is processed on the parallel training data sets run in character alignment, proceeding character extraction and scoring, creating a lexical reordering table and a Moses configuration file. Followed by a tuning step on the parallel tuning data sets, resulting in a ini file with trained weights for translation. Finally the test step to perform spelling normalization as translation and generating a normalized test file based on the original test data. Five SMT
systems are built using this approach, one with Moses default reordering parameters, one without reordering parameters, and the other three with different reordering parameters. The different character-based SMT systems are:

- SMT1 \textit{wbe-msd-bidirectional-fe-allff} (default)
- SMT2 \textit{hier-mslr-bidirectional-fe-allff}
- SMT3 \textit{phrase-mslr-bidirectional-fe-allff}
- SMT4 \textit{hier-leftright-forward-f-allff}
- SMT5 \textit{without reordering}

The reordering model parameters initially change the sequence of words or phrases in an input source sentence in the translation process, which contains five main parts as \textit{modeltype}, \textit{orientation}, \textit{directionality}, \textit{language}, and \textit{collapsing}. In the spelling normalization task, the reordering model parameters in SMT change the sequence of characters in an input unnormalized word. \textit{Modeltype} consists of three optional parameters as \textit{web}, \textit{phrase} and \textit{hier}, in which \textit{web} determines the orientation of two words based on word alignments at training time and phrase alignments at decoding time, \textit{phrase} based on phrase alignments at both training and decoding, while \textit{hier} determines the orientation based on a combination of several phrases. \textit{Orientation} consists of four optional parameters determining the classes of orientation used as \textit{mslr}, \textit{msd}, \textit{monotonicity}, \textit{leftright}, in which \textit{mslr} takes four orientations into consideration of \textit{monotone}, \textit{swap}, \textit{discontinuous-left}, \textit{discontinuous-right}, \textit{msd} considers three orientations of \textit{monotone}, \textit{swap}, \textit{discontinuous}, \textit{leftright} considers \textit{left} and \textit{right} orientations. \textit{Directionality} consists of three optional parameters as \textit{backward}, \textit{forward}, \textit{bidirectional}, determining if the orientation should be based on the previous or the next phrase, in which \textit{bidirectional} considers both \textit{backward} and \textit{forward} models. \textit{Language} consists of two optional parameters as \textit{fe} and \textit{f} determining which language the model should be based on, in which \textit{fe} considers both the source and target language and \textit{f} considers only source language. \textit{Collapsing} consists of two optional parameters of \textit{allff} and \textit{collapseff} determining how the scores are treated, in which \textit{allff} treats the scores individually.

4.3 Character-Based NMT Approach

To implement the character-based NMT approach, seq2seq is applied and the same preprocessing of adding whitespace between letters of each token in the USE data sets is proceeded as for the character-based SMT approach described in Section 4.2. To realize the spelling normalization task using the NMT approach, first of all, two vocabulary sets are generated on the parallel characterized USE training data sets. Then a training step is proceeded where an encoder works,
and a training step is set for this step with different sizes of the \textit{yml} file setting different parameters as illustrated in Table 4.1. Followed by the step of making predictions and then the decoding part where a decoder interprets the hidden states into normalized character sequences on the original test data set. Four character-based NMT systems are built with training steps and sized model file as:

- NMT1 10,000 steps small \textit{yml} model size
- NMT2 50,000 steps medium \textit{yml} model size
- NMT3 10,000 steps large \textit{yml} model size
- NMT4 50,000 steps large \textit{yml} model size

The differences of three \textit{yml} files with small, medium, and large model sizes is shown in Table 4.1 as below.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Small \textit{yml}</th>
<th>Medium \textit{yml}</th>
<th>Large \textit{yml}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Units</td>
<td>128</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>Encoder Cell</td>
<td>GRUCell</td>
<td>GRUCell</td>
<td>LSTMCell</td>
</tr>
<tr>
<td>Encoder Layers</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Decoder Cell</td>
<td>GRUCell</td>
<td>GRUCell</td>
<td>LSTMCell</td>
</tr>
<tr>
<td>Decoder Layers</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

\textbf{Table 4.1:} Different parameters in three \textit{yml} model

These \textit{yml} files for different model sizes are example configurations and have a large number of available hyperparameters for users to tune. The increasing of the training steps, the number of units and the number of layers result in the increasing of training time for the NMT systems.
5 Evaluation and Results

Spelling normalization results for the different approaches are evaluated on the USE test data set by two evaluation measures: **accuracy** and **error reduction**.

\[
\text{Accuracy} = \frac{\text{Number of Correct}}{\text{Number of Correct} + \text{Number of Incorrect}}
\]

\[
\text{Error Reduction} = \frac{\text{Number of Normalized}}{\text{Number of Incorrect}}
\]

The **accuracy** and **error reduction** are used for evaluating all three approaches. **Accuracy** is to calculate the total number of tokens after normalization which are identical to the gold standard of the test data set, measuring the similarity of normalized texts with gold standard texts. **Error reduction** is to calculate the number of corrected normalized tokens after normalization divided by the number of unnormalized tokens in the original test data set, measuring the effectiveness of normalization.

5.1 Results for the Combination of the Levenshtein-based Approach and the Phonetic Approach

Experiments on combining the Levenshtein-based and the phonetic approach tested on the USE test data set measured by **accuracy** and **error reduction** are listed in Table 5.1.

<table>
<thead>
<tr>
<th>Components</th>
<th>Accuracy</th>
<th>Error Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>99.496%</td>
<td>n/a</td>
</tr>
<tr>
<td>GT1</td>
<td>99.737%</td>
<td>45.45%</td>
</tr>
<tr>
<td>GT8+DL+FC</td>
<td>99.606%</td>
<td>18.18%</td>
</tr>
<tr>
<td>GT8+MP+FC</td>
<td>99.614%</td>
<td>20%</td>
</tr>
<tr>
<td>GT8+DL+MP+FC</td>
<td>99.807%</td>
<td>60%</td>
</tr>
<tr>
<td>GT8+DL+MP</td>
<td>99.789%</td>
<td>56.36%</td>
</tr>
<tr>
<td>GT8+DL+MP+FC+CS</td>
<td>99.825%</td>
<td>63.63%</td>
</tr>
</tbody>
</table>

Table 5.1

Baseline=Unnormalized version of the test data set.
GT1=Get-close-match top1 from the generated normalization candidate set.
GT8 = Get-close-match top8 from the generated normalization candidate set.
DL = Damerau Levenshtein edit distance approach.
MP = Metaphone similarity comparison approach.
FC = Frequency count.
CS = Compound splitting.

With a 0.241% improvement over the baseline, the get-close-match top1 component with one best match in generating the normalization candidate set shows relatively good results, but when extending the top1 match to get-close-match top8 with eight matches and selecting a best match of normalization with Damerau Levenshtein or Metaphone comparison adding frequency count, the accuracy drops by 0.131% and 0.123% respectively, referring to that selecting a best match within eight candidates by merely Damerau Levenshtein or Metaphone comparison does not outperform the basic get-close-match top1 component. However, when these two components are combined, it outperforms get-close-match top1 by 0.07% improvement on accuracy and 10.91% improvement on error reduction.

Adding the frequency count component also shows a slight improvement in accuracy and error reduction by which the GT8+DL+MP+FC has 0.018% and 3.64% higher score than that of the GT8+DL+MP, but the improvement is not settled, for when the frequency count is not included, a random selection is performed if there is still one more best candidate selected by GT8+DL+MP. An example of when frequency count are needed would be:

'wich' -> 'witch':1, 'which':1
'waht' -> 'wat':1, 'what':1

After generation by GT8 and selecting by DL+MP, two best normalization candidates are left, and to choose one best candidate (which should be the word with fill blue textcolor), frequency count is added to lookup the frequency of each candidate based on a subset of BNC, and the best candidate for the two OOV words is correctly selected by frequency count, while in the occasion with random choice, the correct rate might reduce 50%. The frequency counts for candidates of these two example OOV words are:

'wich' -> 'witch':11, 'which':7759
'waht' -> 'wat':2, 'what':3805

Furthermore, including the compound splitting component in the approach also shows a positive impact on the results by a 0.018% and 3.63% improvement compared to the same approach without compound splitting. Examples of the effectiveness of compound splitting are:

'schoolyears' -> 'school years'
'childrenlitterature' -> 'children literature'
'popartists' -> 'pop artists'
'holidaytrips' -> 'holiday trips'
However, it also has a problem in detecting OOV compounds. Since it is implemented in the normalization candidate selection step, if the value of the best match is bigger than or equal to three, the candidate is tested for compound splitting. It works well for most of the OOV compounds which are composed by two words longer than three letters such as ‘schoolyears’, but for compounds such as ‘infact’ (‘in fact’ as correctly normalized) which could have a best normalization candidate with an edit distance of one for ‘infect’, this word would not be considered for compound splitting, and would thus get an incorrect normalization.

5.2 Results for the Character-Based SMT Approach

For the character-based SMT approach for spelling normalization of English student writings, five systems with different reordering model parameters and without reordering are built (detailed setup is presented in Section 4.2). Table 5.3 shows the results on the USE test data set:

<table>
<thead>
<tr>
<th>Systems</th>
<th>Accuracy</th>
<th>Error Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>99.496%</td>
<td>n/a</td>
</tr>
<tr>
<td>SMT1</td>
<td>99.506%</td>
<td>18.2%</td>
</tr>
<tr>
<td>SMT2</td>
<td>99.661%</td>
<td>32.73%</td>
</tr>
<tr>
<td>SMT3</td>
<td>99.661%</td>
<td>32.73%</td>
</tr>
<tr>
<td>SMT4</td>
<td>99.661%</td>
<td>32.73%</td>
</tr>
<tr>
<td>SMT5</td>
<td>99.719%</td>
<td>41.81%</td>
</tr>
</tbody>
</table>

Table 5.2

It is shown that with a reordering model parameter of \textit{wbe-msd-bidirectional-fe-allff}, SMT1 system has a very small improvement of 0.01% compared to the baseline (original test set), and the other SMT systems returned the same accuracy and error reduction but achieve a slight improvement over SMT1 of 0.059% accuracy and 14.52% error reduction. The different parameters between SMT1 and the rest of the SMT systems is in the modeltype part where SMT1 used \textit{wbe} based on word alignments and the others used phrase and hier based on phrase alignments. In character-based SMT, it means that the SMT1 model determines the orientation of two letters based on character alignments, and the other SMT models determines the orientation of several letters. However, other parameter changes result in no visible changes in measuring scores as well as in manual inspection in different normalization results from different SMT systems. It might be because the training and tuning parallel data are not spelling error pairs but are ordinary Swedish university students’ English writings that contain misspellings and typos, resulting in that by regarding the original texts as source texts and normalized texts as target texts, there is not a large amount of differences, leading to the consequence that SMT is not sensitive enough to the misspellings and typos.

However, by switching off the reordering model, SMT5 system achieves an improvement over all other SMT systems with a 0.222% accuracy and a 41.81%
The reordering model concerns the sequence change between characters in an input token, e.g. whether 't e h' should be rearranged to 't h e'. It might have some positive impact on normalizing typographic errors rather than cognitive errors, and meanwhile it might also lead to unnecessary transpositions between characters. This is a possible reason for the fact that SMT5 without reordering performs better than the other SMT systems with different reordering models.

5.3 Results for the Character-Based NMT Approach

For the character-based NMT approach for spelling normalization of English student writings, four systems with different training steps and model sizes are built. Detailed setup is presented in section 4.3. Table 5.4 shows the results on the USE test data set:

<table>
<thead>
<tr>
<th>Systems</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>99.496%</td>
</tr>
<tr>
<td>NMT1</td>
<td>93.899%</td>
</tr>
<tr>
<td>NMT2</td>
<td>95.899%</td>
</tr>
<tr>
<td>NMT3</td>
<td>94.364%</td>
</tr>
<tr>
<td>NMT4</td>
<td>99.579%</td>
</tr>
</tbody>
</table>

Table 5.3

According to the accuracy, it is shown that the former three character-based NMT systems we implemented have a negative impact on the normalization results, meanwhile, the enlargement of the model size and increasing of training steps have a relatively big and positive influence on the normalization results. After manually observing the NMT normalization results, we find that many output words decoded by the NMT systems contain ill-decoded characters such as 'responssiiity', 'representtttesee' which should be 'responsibility' and 'representatives'. By viewing back to the early training steps’ output, we observe a similar occasion that when the NMT systems encode the character sequences to hidden states and then decode the hidden states to normalized character sequences of correct tokens, it learns and calculates through each training step, and the initial output of less training steps as 100 are ‘rrrr’ for ‘room’ for an example. Hence, the unsatisfactory experimental results of the character-based NMT systems might be due to not having enough training steps. Moreover, similar to the character-based SMT systems, the lack of parallel spelling error pairs might also be one reason. The NMT4 system with 50,000 training steps and large \textit{yml} model file achieves the best performance in all four NMT systems and improves the normalization accuracy with 0.083% compared to the baseline (original test set), showing that the increasing of training steps and enlarging of the \textit{number of units} and the \textit{number of layers} parameters have a positive influence on the performance of the character-based NMT system using seq2seq.
6 Conclusion and Future Work

6.1 Conclusion

Spelling normalization as a task for normalizing non-standard words into standard ones has a positive impact on the performance of many NLP tasks such as information retrieval, opinion mining and machine translation. Inspired by SWEGRAM and different spelling normalization methods of historical texts and noisy modern texts, we aimed at exploring different approaches for spelling normalization of English student writings, and answering our research questions of how well do different approaches for spelling normalization perform for the task of normalizing English student writings and could we improve the performance by adding frequency count and compound splitting.

In this thesis, we implemented three main approaches for spelling normalization of English student writings: the Levenshtein-based and phonetic similarity-based approach, the character-based SMT approach, and the character-based NMT approach and conducted corresponding experiments and evaluations on the USE test set. For the Levenshtein-based and phonetic similarity-based approach, three steps are implemented: detecting OOV words in a lookup dictionary based on WordNet and a subset of BNC, generating the candidate set for each OOV word, and selecting the best normalization candidate. Different approaches of Get-close-match, Damerau Levenshtein edit distance, Metaphone algorithm, and the combination of the Damerau Levenshtein edit distance and Metaphone algorithm are implemented and two components of a frequency count and a compound splitting are added into the approaches to be evaluated. The combination of Damerau Levenshtein edit distance and Metaphone algorithm approach with the frequency count and the compound splitting components showed the best performance among the three main approaches with an improved accuracy of 0.329% and an improved error reduction of 63.63% compared to the baseline of the original unnormalized test set. The experiments on the character-based SMT approach with five different systems in reordering showed that the system without reordering model achieved an improved accuracy of 0.223% and an improved error reduction of 41.81% compared to the baseline. For the character-based NMT approach, a better performance is achieved by the NMT4 system with the largest training step and model size with an improved accuracy of 0.083% compared to the baseline.

In conclusion, the approaches we implemented and the experiments we conducted basically satisfy our expectation for the spelling normalization task of English student writings and clearly answer our research questions. More ideas about possible improvement and future work is discussed in section 6.2.
6.2 Future Work

Since the approach combining Levenshtein edit distance and the phonetic Meta-phone algorithm does not include a weighted Levenshtein calculation but only considers the similar phonetic representation and minimum Damerau Levenshtein edit distance, it to some extent ignores the correct normalization candidate with a bigger Levenshtein edit distance. Further development of combining a weighted Levenshtein-based approach with phonetic similarity comparison could result in a possible improvement for the spelling normalization task.

Furthermore, this thesis is inspired by SWEGRAM which is a pipeline for automatic annotation and quantitative analysis of Swedish. A similar pipeline for English could be further explored and developed based on our spelling normalization approach for English student writings, and it could be extended to a pipeline consisting of tokenization, spelling normalization, part-of-speech tagging, lemmatization, syntactic parsing using UDPipe and quantitative analysis for frequency statistics for automatic data annotation and linguistic analysis.


Han, Bo, Paul Cook, and Timothy Baldwin (2013). “Lexical normalization for social media text”. ACM Transactions on Intelligent Systems and Technology (TIST) 4.1, p. 5.

Han, Na-Rae, Joel R Tetreault, Soo-Hwa Lee, and Jin-Young Ha (2010). “Using an Error-Annotated Learner Corpus to Develop an ESL/EFL Error Correction System.” In: LREC.


32
In: Meeting of the ACL on Interactive Poster and Demonstration Sessions, pp. 177–180.


