Detecting spelling variants in non-standard texts

Fabian Barteld

Institut für Germanistik / Language Technology Group, Department of Informatics
Universität Hamburg
firstname.lastname@uni-hamburg.de

Abstract

Spelling variation in non-standard language, e.g. computer-mediated communication and historical texts, is usually treated as a deviation from a standard spelling, e.g. *2mr* as a non-standard spelling for *tomorrow*. Consequently, in normalization – the standard approach of dealing with spelling variation – so-called non-standard words are mapped to their corresponding standard words. However, there is not always a corresponding standard word. This can be the case for single types (like emoticons in computer-mediated communication) or a complete language, e.g. texts from historical languages that did not develop to a standard variety. The approach presented in this thesis proposal deals with spelling variation in absence of reference to a standard. The task is to detect pairs of types that are variants of the same morphological word. An approach for spelling-variant detection is presented, where pairs of potential spelling variants are generated with Levenshtein distance and subsequently filtered by supervised machine learning. The approach is evaluated on historical Low German texts. Finally, further perspectives are discussed.

1 Introduction

Spelling variation is a well-known feature of non-standard language, e.g. computer-mediated communication (CMC) and historical texts (Baron et al., 2009; Eisenstein, 2013). One problem is that this variation decreases the utility of unannotated corpora, e.g., by reducing the recall for queries and distorting keyword frequencies (Baron et al., 2009). Furthermore, the variation makes it harder to annotate this data automatically since the number of out-of-vocabulary (OOV) words is higher when compared to the same amount of standardized data. In addition, during training time, instances of the same morphological word appear as different types thereby distributing the information about one morphological word over these types.

The predominant way to deal with spelling variation is normalization, i.e. non-standard words are mapped to a corresponding standard word, or a canonical form. In this thesis proposal, we pursue an alternative approach: the task of spelling-variant detection, i.e. instead of mapping non-standard or historical words to a standard form as in normalization, the aim is to detect spelling variants in a set of types without reference to a canonical form. Therefore, this task can be applied in cases where no canonical form exists. The detected spelling variants can then be used to mitigate the problems caused by spelling variation that were described above. An approach for detecting spelling variants is presented and evaluated on Middle Low German (GML), a group of German dialects from between 1200 and 1650. These dialects developed into Low German, a dialect group of German that has not undergone standardization. Therefore, there exists no contemporary variant of Low German with standardized orthography that could be used as target language.

After having discussed related work in Section 2, we elaborate on the task of spelling-variant detection (Section 3) and introduce an approach using binary classification (Section 4). We present first experiments on generating candidate pairs (Section 4.1) and filtering these pairs using a supervised machine learning approach (Section 4.2). Finally, we conclude with an outlook on further planned research in the framework of this thesis.

Proceedings of the Student Research Workshop at the 15th Conference of the European Chapter of the Association for Computational Linguistics, pages 11–22, Valencia, Spain, April 3-7 2017.

2 Related work

In normalization – also called standardization (Ljubešić et al., 2014) or canonicalization (Jurish, 2010a) – spelling variation is seen as a deviation from a given standard. In the case of CMC this is the standardized language as used in newspapers and regarding historical texts this is the corresponding contemporary standard language. This kind of normalization has been criticized as a "lossy translation" for CMC data (Gimpel et al., 2011) and it has been shown that it is not capable to deal with all peculiarities that appear in non-standard texts.

An example for this is that normalization cannot deal with differences in the usage of a word between the standard and the non-standard domain that result in different labels – a tagger trained on the standard domain will apply the wrong tag. Consequently – as Yang and Eisenstein (2016) show – the accuracy of tagging historical texts benefits from combining domain adaptation and normalization.

Such differences in the usage of a word are also visible in semantic changes. Bollmann et al. (2012) therefore distinguish between normalization and modernization: While in the first a historical word is mapped to its modern cognate, in the second the historical word is more loosely translated. They give the example of the Early New High German (1350-1650) word *vrlaub* 'permission' which would be normalized to its Modern German cognate *Urlaub* 'vacation' but modernized to the Modern German word *Erlaubnis* 'permission'. Another, more subtle example for a lossy normalization, would be *Kopf* and *Haupt*. Both denote 'head' in Modern German, but *Haupt* is only used in exalted language. However, in the Middle High German period (1050-1350), *houbet* the cognate of *Haupt* was the most commonly used word while the cognate of *Kopf* was mainly used in descriptions of battles (Fritz, 2006). Such differences will not be relevant for tasks like POS tagging. Still, this example shows the lossy nature of normalization to a modern standard.

Another limitation of normalization is that it can only deal with items that have a corresponding standard word. This can be solved by creating an artificial standard for the specific phenomenon and normalizing towards this standard. Dipper (2011) has shown that using an artificial standardized version of Middle High German leads to better results training and applying POS and morphological taggers to these texts. In CMC, items like emoticons have no corresponding standard form and require a special treatment when normalizing these texts. E.g., for the shared task of normalizing Twitter data (Baldwin et al., 2015) only all-alphanumeric tokens are normalized. This excludes tokens like *=)*, *:)* and *:-)* from the normalization. One way to deal with variation in emoticons is again the normalization to an artificial standard, cf. the manual mapping of different emoticons to synsets used by Hogenboom et al. (2015). Hence, normalizing to an artificial standard solves the problems of normalization. However, this introduces the need to develop the standard first.

These problems do not appear when detecting spelling variants without the reference to a standard. The detection of spelling variants has been applied in a minor strand of work on automatically annotating non-standard texts. For POS tagging historical texts, the knowledge about spelling variants has been used to substitute OOV words with possible spelling variants – this improved the accuracy of POS taggers trained on the non-standard data (Logačev et al., 2014; Barteld et al., 2015). A similar approach has been used by Gadde et al. (2011) for SMS text. In all of these approaches the knowledge about spelling variants is used independently from the tagger, either in a pre-processing step to reduce the amount of spelling variation before training and/or tagging or as a post-processing step to correct tagging errors.

Alternatively, specialized tools can be developed that directly use the knowledge about spelling variation. This approach has been followed by Kestemont et al. (2010) and Barteld et al. (2016) for lemmatization of historical texts.

With the exception of Acharyya et al. (2009), who present a clustering approach based on context similarity that also takes surface similarity into account, spelling-variant detection has not been addressed independently of a specific task like POS tagging. However, Acharyya et al. (2009) only present an anecdotal evaluation of resulting clusters.

Similar to the task of detecting pairs of spelling variants is the detection of pairs of standard and non-standard words appearing in noisy text and the detection of cognate pairs. Usually, surface similarity, contextual similarity or a combination of both is used for these tasks. Gouws et al. (2011)

use the bigrams that appear before and after a word to compute distributional similarity and string kernels to measure surface similarity to create a dictionary of non-standard and corresponding standard types.

3 Defining spelling variation and spelling-variant detection

In order to define spelling variation, we distinguish between *type* and *morphological word*. Morphological words are the inflected word forms of a language. This distinction is similar to the distinction between *morphological word* and *lexeme*: A lexeme like the English verb *(to) be* appears as different morphological words in texts, e.g. *am* and *was*. In lemmatization, the task is to assign the same lemma to the different morphological words of the same lexeme, thereby abstracting over inflectional differences. Similarly, the aim of spelling-variant detection is to detect the types that belong to the same morphologicial word. However, while the morphological words belonging to the same lexeme differ with respect to their morphology, the types that belong to the same morphological word only differ in their spelling.

Consequently, the notion morphological word can be operationalized by combining POS, morphological information, lemma and word-sense disambiguation: Each combination of these attributes is a morphological word. For example {*Personal Pronoun, Nominative Sg., I*} is a morphological word of English. In texts, these abstract word forms are realized by using types. Types are the words t of a language $L \subseteq \Sigma^*$ for some given alphabet Σ. For a standardized language, a unique mapping from a morphological word to a type is expected.[1] For instance, {*Personal Pronoun, Nominative Sg., I*} is usually realized as 'I'. In GML texts, the corresponding morphological word can be realized using the types given in Example 1.

(1) yck, ick, jk, ik, yk, jck, jc, ic

This example shows *i*, *j* and *y* that appear in general mostly interchangable in GML manuscripts and prints. Only bi-graphs like *ei* are an exception to this rule. Niebaum (2000) gives an overview over the graphematic inventory of GML. He lists the grapheme <*i*> and amongst others *j* and *y* as variants. He states that these are often used instead of *i* to disambiguate the spelling next to letters like *m* and *n*. However, the example of 'I' shows that they are used interchangeably in other contexts as well. Such variation can be easily modelled in a rule-based approach. We will compare our approach with a rule-based approach developed for GML.

Variation as in Example 1 is what we define as spelling variation in a broad sense: the realization of a morphological word using different types. There is spelling variation even in standardized texts. One example is spelling errors that lead to a variation. But there is also real spelling variation in standardized language. One example for this rare case where two different types are used for the same morphological word in standardized language is the co-existence of *ß* and *ss* in Modern German which leads to the spelling variants *Fuß* and *Fuss* '(the) foot'. This is, however, negligible for standard texts: only 7% of the morphological words appearing more than once in the Tiger corpus (Brants et al., 2004), a corpus consisting of German newspaper texts, show variance, i.e., are realized by more than one type.[2]

In non-standard texts, there is more variation: In the English Twitter texts used as the training data for the W-NUT 2015 shared task on normalization (Baldwin et al., 2015), 57% of the morphological words show variation.[3] This can be reduced to 16% by lowercasing every type.

Historical texts, for which the amount of data available is often extremely small, also show a lot of variation. In the GML texts from the Reference Corpus Middle Low German (Peters and Nagel, 2014) that are used for the experiments in this paper[4], 58% of the morphological words show variation. However, less of this variation is due to differences in the case as in the Twitter data: If every

[1] Not the other way around, due to ambiguity. The German type *Bank* can either denote the nominative singular of *bench* or *bank*. However, the German morphological word *bench* (nominative singular) is always realized as *Bank*.

[2] As the Tiger corpus is not annotated with word-sense disambiguation, this overestimate the actual amount of spelling variation in the corpus as e.g. the types *Bänke* ('benches') and *Banken* ('banks') are incorrectly treated as the same morphological word {*Noun, Nominative Pl., Bank*}. Tokens tagged with FM, XY, OA, PTKVZ or one of $., $, and $(have been excluded for the calculation.

[3] As the W-NUT 2015 data is neither annotated with POS, morphology nor lemma, we use the normalization annotation: Each normalization is treated as a morphological word. Tokens containing non-alphanumeric characters have been excluded as they were not normalized.

[4] We use the release 2016-08-23 of the corpus (`http://hdl.handle.net/11022/0000-0001-B002-5`).

type is lowercased still 54% of the morphological words show variation.

These numbers quantify the internal – or synchronic, cf. Piotrowski (2012) – variation in the corpus and thereby indicate the amount of the data sparsity problem. Consequently, they give an indication of how hard it is to develop tools for this language variant. This is different than the usually reported number of OOV types with respect to a dictionary of the corresponding standard. The latter only indicates how promising it is to apply tools developed for the standard to the language variant in question.

Normalization does not deal with spelling variation directly but with non-standard words, a term used by Sproat et al. (2001), who introduced the task of normalization. For historical text, the non-standard words are historical words or historical forms of contemporary words – the standard words are contemporary words. This approach only looks at diachronic variation and not at synchronic variation as defined above. Internal variation in the data is only dealt with indirectly by mapping the non-standard types to a corresponding standard type. Hence, it resembles a translation task, a framework in which normalization has been approached (Kobus et al., 2008; Scherrer and Erjavec, 2016). The task of detecting spelling variants shifts the attention towards the internal variation and resembles an information retrieval task where the aim is to detect unordered pairs of types like GML {*jc*, *ik*} which are used to realize the same morphological word. More formally, given a set of types $L \subseteq \Sigma^*$, the aim is to retrieve all pairs of types that are spelling variants, i.e. the set $SV \subseteq \{\{t_1, t_2\} | t_1 \in L, t_2 \in L, t_1 \neq t_2\}$. SV defines the spelling-variant relation.

As has been noted, spelling variation as defined in this paper is a broad cover term for different types of variation that appear between types for which the spelling-variant relation holds. In order to give an overview over the phenomena that have to be covered for spelling-variant detection, the following constructed pair of GML sentences illustrates three different types of spelling variation.

(2) Do he komen was van deme kloster
DO he ghecomen was uan dem kloster
when he come.PPTC was from the cloister
'when he had been coming from the cloister'

{*Do*, *DO*} and {*van*, *uan*} from Example 2 illustrate spelling variation in the narrow sense, i.e. two types for which the same pronunciation is assumed. However, spelling variation in the broad sense also covers {*deme*, *dem*} where the final <e> is assumed to correspond to the pronunciation of a final [ə]. Finally, there is morphological variation as in {*komen*, *ghecomen*} where the types differ by the spell-out of the morphological marker *ghe* in the participle *komen* '(to) come'.

A clear-cut distinction between those three types of variation is not always possible: {*deme*, *dem*} could also be a spelling variant in the narrow sense, as it cannot be decided for a single instance if there actually was a difference in the pronunciation. Furthermore, the difference between {*deme*, *dem*} could also be classified as morphological variation, treating the *e* as the overt dative marker.

As has been pointed out above, we cover errors under the term spelling variation as well. While errors are usually defined as a deviation from a norm (Brill and Moore, 2000), in the case of the lack of a norm, we define them as a type of variant that is unlikely to appear, it may even appear only once. The GML corpus, for instance, contains one instance of *gesprok* as the past participle of *speak*, whereas other instances of the past participle are realized with the suffix *en*. In this example it is likely that this suffix was to be realized as an abbreviation, i.e. a dash over the *k*, and was simply forgotten.

Besides actual spelling errors, for historical texts another source of spelling variation are errors in the transcription (done manually or with optical character recognition). These lead to variants that should be discovered by the algorithm as well.

4 Approaches towards spelling-variant detection

For the experiments presented in this paper, we use five texts from the Reference Corpus Middle Low German (Peters and Nagel, 2014). The corpus is annotated with POS, morphological information, lemma and word-sense disambiguation. In order to exclude temporal and spatial variants, the texts are taken from the same dialect region and roughly from the same time (about 1500 AD). The texts consist of 36,269 tokens. We use two texts that constitute about 80% of the tokens ('Buxteh. Ev.', 19,644 tokens and 'Griselds', 9,057 to-

kens) for training, the other three texts ('Veer Koeplude', 4,691 tokens, 'Agneta Willeken', 2,269 tokens and 'Reval Tot.', 608 tokens) are each split into two halves, using the first halves as development and the second halves as test set. Pairs of types that are instances of the same morphological word, i.e. they have the same POS, morphology and lemma containing word-sense information, are extracted from these texts. Tokens, for which the annotation groups together types that are not spelling variants were removed from the the corpus. This includes text that is struck through in the manuscript. Such tokens were always annotated with the tag OA ('no annotation'). This reduces the number of training tokens to 26,915, the size of the development set to 3,393 tokens and the size of the test set to 3,396 tokens. We report precision, recall and F-score for the set of spelling-variant pairs extracted from the test set that did not appear in the training data. These are 68% of the pairs. This way of splitting the data into training and test set makes the task harder than directly splitting the set of spelling-variant pairs as the amount of pairs containing rare words will be higher in this setting. However, this way of evaluating the task will give a more realistic estimation of the performance for applications like POS tagging for which the main task of spelling-variant detection is exactly to identify variants of OOV words in new texts.

We approach spelling-variant detection in two steps: First, we generate pairs of spelling variants, then we employ a supervised binary classifier to filter out overgenerated pairs. The generation step is employed in order to reduce the number of pairs that have to be classified. Without this step, for a given set of types L each of the $\binom{n}{2}$ pairs $\{t_1, t_2\} \in L \times L$ would have to be classified, which is computationally intractable for large sets. This pair generation step needs to be fast while having a high recall for actual spelling variants.

4.1 Candidate-pair generation

For candidate-pair generation, we rely on surface similarity between the types. From the set L, we generate all pairs of types for which the Levenshtein distance (Levenshtein, 1966) is below a given threshold s. There are several efficient approaches for detecting all types from L for which the distance is below s, some have even been proposed in the context of normalization, e.g. anagram hashing (Reynaert, 2009). We use a Levenshtein automaton (Schulz and Mihov, 2002) to retrieve all the pairs of types from L that have a distance below $s \in \{1, 2, 3, 4\}$. Table 1 shows recall, precision and F-score as well as the average number of candidate pairs per type (arithmetic mean and standard deviation) for the different values of s. The numbers show that most of the spelling variants have a distance smaller than or equal to 3. Going to distance 4 improves recall from 0.97 to 1 but also increases the average size of generated candidate pairs from about 83 to 261 per type. The precision is always very low, even at a Levenshtein distance of 1 where the average number of predicted variants is slightly below 2. This is due to the fact that many types do not have spelling variants.

For cognate recognition as well as mining pairs of standard and non-standard words, variants of a weighted Levenshtein distance have been used in order to increase recall and precision, e.g. by Hauser and Schulz (2007) for detecting historical variants of modern words and Gomes and Lopes (2011) in cognate detection. These approaches usually employ quasimetrics, i.e. the used metrics are not necessarily symmetric as the weights learned for edit operations are learned in one direction – $i \rightarrow y$ may have a different cost than $y \rightarrow i$. This makes sense in the context of normalization and cognate detection as the comparisons made in these cases are directed as well, e.g., types from older stages of a language are compared to the modern variant. However, in the case of spelling-variant detection the weights for both directions should be equal because next to the pair {*ghecomen*, *komen*} from Example 2 the pair {*ghekomen*, *comen*} exists where the difference *c* $\leftrightarrow$ *k* appears in opposite directions.

We use an undirected version of the measure SPSim by Gomes and Lopes (2011) as a baseline for our experiments. This measure employs substitution patterns (SP), i.e. segments of mismatches from the alignment of the types in the candidate pair with their left and right context. Example 3 shows a pair of spelling variants and the corresponding undirected SP with a context of length 2 denoted by the triple (left context, {pair of mismatched characters}, right context). The context is padded with *$* at the beginning and the end of the type.

(3) maria, marien
('ri', {'a', 'en'}, '$$')

Lev	R	P	F1	AVG	SD
1	0.58	0.12	0.20	1.85	2.42
2	0.88	0.02	0.04	15.99	20.06
3	0.97	0.00	0.01	83.39	84.78
4	1.00	0.00	0.00	261.41	202.86

Table 1: Recall and Precision using the Levenshtein distances
Levenshtein distance (Lev), Recall (R), Precision (P), F-score (F1), Number of candidates per type: average (AVG) and standard deviation (SD)

Lev	R	P	F1	AVG	SD
1	0.48	0.18	0.26	1.11	1.51
2	0.65	0.09	0.15	2.93	3.92
3	0.66	0.05	0.10	4.54	6.11
4	0.67	0.05	0.09	5.08	6.58

Table 2: Results using undirected SPSim (0.9)
Levenshtein distance (Lev), Recall (R), Precision (P), F-score (F1), Number of candidates per type: average (AVG) and standard deviation (SD)

The measure is trained on positive examples. When applying SPSim, SPs that appeared in the training data get a cost of 0, otherwise their cost is the edit distance between the mismatched segments. Furthermore, the context is generalized, i.e. when a mismatch segment appears in the training data with at least two different contexts, the mismatch will always get a cost of 0 regardless of the context.

Using this measure, pairs of types where all the changes are known get the maximal similarity of 1. This allows for improving the precision without losing on recall by setting a high threshold on the similarity for cognate detection. The results using the undirected version of SPSim for identifying spelling variants in the GML test set can be seen in Table 2. The threshold of 0.9 produced the best results on the development set.

While there is an improvement in F-score from 0.20 to 0.26, the precision is still very low (0.18). One reason for this is that the training data contains a lot of very generic substitutions that are learned by SPSim. The following SPs are the SPs that are learned from the training data and involve a single 'a': ($\emptyset$, {'a', 'o'}, $\emptyset$), ($\emptyset$, {'a', 'u'}, $\emptyset$), ($\emptyset$, {'a', 'e'}, $\emptyset$) and ($\emptyset$, {'a', 'en'}, $\emptyset$).

With this generic set of SPs, two types like *dach* '(the) roof' and *doch* 'but' differing only in *a* vs. any other vowel except for an *i* will have a similarity of 1. The pattern ($\emptyset$, {'a', 'u'}, $\emptyset$) is learned from the two spelling variants {*sundighe*, *sandige*} and {*ghehat*, *ghehut*}. However, the first pair is likely to be an error in the original manuscript, the second example is an error in the gold annotation, leading to a wrongly learned pattern.

In order to make the classification more robust against such noise in the data, a more complex weighting scheme for SPs than 0 and 1 should be used. We follow the approach that Ciobanu and Dinu (2014) apply to cognate recognition by training a binary classifier on positive and negative examples for spelling variants to filter out overgenerated candidate pairs.

4.2 Filtering overgenerated candidate pairs

For filtering out overgenerated candidate pairs, we apply a binary classifier that is trained on positive and negative examples of pairs of types. We experiment with two different kinds of features: surface features – representing similarities and differences in the strings – and context features.

As surface features we use undirected SPs as defined in the previous section as well as paired character n-grams around mismatches (Ciobanu and Dinu, 2014), and all paired character n-grams (Ciobanu and Dinu, 2015) extracted from the aligned sequences, see Example 4.

(4) maria, marien
2-grams: {$m, $m}, {ma, ma}, ...,
{ia, ie}, {a_, en}, {_$, n$}
2-grams(mis): {ia, ie}, {a_, en}, {_$, n$}

For the n-grams we test all combinations of lengths in $\{1, 2, 3\}$. Similarly, for the SPs we use context sizes of $\{0, 1, 2\}$. Furthermore, we combine the n-grams with SPs.

As context feature, we use the cosine similarity between dense vector representations (vec) obtained using positive pointwise mutual information with singular value decomposition on a larger unannotated set of GML texts (1,730,614 tokens) than the annotated texts used for the experiments. As suggested by Levy et al. (2015), we have tested different hyperparameters for the creation of the dense vectors: dimensions ($\{125, 250, 375, 500\}$), context windows ($\{2, 5\}$) and frequency thresholds ($\{10, 25, 50, 75, 100\}$). We have also combined the context feature with the best performing surface features and the surface features with the best performing context feature.

For classification a Support Vector Machine (SVM) is trained. We use a radial basis function (RBF) kernel and train the model with the Weka (Witten et al., 2011) wrapper for LibSVM (Chang and Lin, 2011) doing a grid-search over the values $\{1, 2, \ldots, 5\}$ and $\{10^{-2}, \ldots, 10^{2}\}$ for the hyper-parameters c and γ on the development set.

The classifier is trained on positive and negative examples. As positive examples, we use all the pairs of spelling variants appearing in the training data (1834 pairs). In order to obtain negative examples, we extract pairs of types with a Levenshtein distance of 1 and of 2 that do not appear with the same annotation using only types that appear at least 10 times in the training data. The frequency threshold is used to reduce the probability that the pair is actually a pair of spelling variants that – due to ambiguity of the types – did not occur as the same morphological word in the training data. The negative pairs are sampled randomly to obtain the same number of negative and positive pairs. In the sampling procedure, we prefer pairs with a lower Levenshtein distance.

We apply the trained classifier to all candidate pairs obtained using the generation process described in the previous section using the Levenshtein distances $1, 2$ and 3. Table 3 shows relevant results.

Overall, all of the features lead to an improvement in F-score over the best F-score obtained using the Levenshtein distance (0.20) and the undirected SPSim (0.26). Combining the different types of surface features did not improve the results.

Differing from the result obtained by Ciobanu and Dinu (2015) for discriminating between cognates and borrowings, using only n-grams around mismatches leads to better overall result than using only n-grams in terms of F-score (0.38 against 0.36), but using all n-grams leads to a slightly better recall (0.43 against 0.42). Both features lead to better results than using SPs, which lead to an F-score of 0.34. However, the differences between these three feature types are small and are not stable across different splits of the dataset.

Using only context features, the results are comparable to the results with surface features, regarding the F-score (0.36). However, this F-score results from a higher recall and a lower precision. A context size of 2, a small frequency threshold (10) and the dimensions 500 and 375 lead to the best results on the data set.

Combining surface and context features results in the best F-score (0.42) using this approach. However, in experiments with vectors obtained from a smaller subcorpus (739,576 tokens), adding the context features led to no improvement over using only surface features.

Regarding the generation method, the best F-scores are obtained using a Levenshtein distance of 1. The increase in recall obtainable by adding further candidate pairs corresponds to a larger drop in precision.

Finally, we compare our approach with a rule-based approach. For this, a set of 26 rules developed by linguists for the purpose of reducing the spelling variation in GML texts is used to detect spelling variants. The rules consist of regular expressions and substitutions. They are applied in a fixed order.[5] Example 5 gives an exemplary rule. Example 6 gives the set of spelling variants for the personal pronoun in first person singular (Engl. 'I') and the remaining variants after applying the rules showing that the number of variants for this morphological word is reduced from 8 to 2 by the rule-based approach.

(5) /ck?(?!h)/ → /k/

(6) {ik, ic, ick, jc, jck, jk, yck, yk} → {ik, jk}

All pairs of types that are mapped to the same form by applying these rules are considered spelling variants. With this approach a slightly better F-score as with SPSim is obtaind (0.29), see Table 4. However, it is outperformed by the machine learning approach. By simply taking the union of the sets of spelling variants obtained using the rules and the best binary classification model, we obtain the best F-score (0.45), see Table 4.

5 Conclusion and future work

In this paper, we presented the task of spelling-variant detection and preliminary results of an approach using supervised machine learning, which was evaluated on Middle Low German texts. The

[5]The script applying these rules to the data has been created by Melissa Farasyn in the project 'Corpus of Historical Low German' (CHLG; `http://www.chlg.ac.uk/index.html`) and contains rules by Melissa Farasyn with additions by Sarah Ihden and Katharina Dreessen both from the project 'Reference Corpus Middle Low German/ Low Rhenish (1200-1650)'.

Lev	Features	R	P	F1	AVG	SD	c	γ
1	n-gram(mis): 1, 2, 3	0.42	0.34	0.38	0.62	0.86	2	10^{-1}
1	n-gram: 1, 2, 3	0.43	0.31	0.36	0.68	0.91	3	10^{-1}
1	SP: 0, 1	0.37	0.31	0.34	0.60	0.84	2	10^{0}
1	vec: 500, 2, 10	0.52	0.28	0.36	0.83	1.08	4	10^{-1}
1	vec: 500, 2, 50, n-gram(mis): 1, 2, 3	0.47	0.37	0.42	0.62	0.87	2	10^{-1}
1	vec: 375, 2, 25, n-gram(mis): 1, 2, 3	0.47	0.38	0.42	0.62	0.86	2	10^{-1}
2	vec: 500, 2, 10, n-gram: 3, n-gram(mis): 1, SP: 2	0.58	0.17	0.26	1.48	1.81	4	10^{-1}

Table 3: Recall and Precision for the binary classification approach
Levenshtein distance (Lev), Recall (R), Precision (P), F-score (F1), Number of candidates per type: average (AVG) and standard deviation (SD), hyperparameters for the SVM (c, γ)

Method	R	P	F1	AVG	SD
Rule	0.19	0.67	0.29	0.20	0.56
SVM	0.47	0.38	0.42	0.62	0.86
SVM+Rule	0.52	0.39	0.45	0.67	0.93

Table 4: Recall and Precision for the rule-based approach and the combination with the best binary classification
Recall (R), Precision (P), F-score (F1), Number of candidates per type: average (AVG) and standard deviation (SD)

Lev	R	P	F1	AVG	SD
1	0.36	0.03	0.05	3.10	6.62
2	0.65	0.00	0.01	48.49	84.93
3	0.83	0.00	0.00	299.57	368.07
4	0.92	0.00	0.00	913.19	785.16

Table 5: Recall and Precision using the Levenshtein distance for English Twitter data
Levenshtein distance (Lev), Recall (R), Precision (P), F-score (F1), Number of candidates per type: average (AVG) and standard deviation (SD)

results obtained are better than using a variant of the trainable edit distance SPSim that was developed for cognate detection. Furthermore, this approach outperformed a rule-based approach with rules developed by linguists for the GML data. Still, the overall F-score obtained is low. In the proposed thesis, we will focus on various ways to improve these results (Section 5.1). In addition, we will extend the scope of the approach (Section 5.2) and use extrinsic evaluations (Section 5.3).

5.1 Improving spelling-variant detection

Improving the precision of the generator seems like a promising way to improve the results, as the drop in precision when going from Levenshtein distance 1 to 2 for generating candidate pairs, led to worse results regarding the F-score.

We will also look into ways to improve the context features, e.g., by using vector representations obtained from the skip-gram (Mikolov et al., 2013) or other models. As for historical texts the amount of texts available is often very limited, we will experiment with ways to improve the obtained vector representations from smaller data sets, e.g., by taking surface similarity into account as Acharyya et al. (2009) do in their clustering approach, by improving the representations of rare words (Sergienya and Schütze, 2015) which are especially important in spelling-variant detection or by using only the most frequent types in the context (Gimpel et al., 2011).

5.2 Extending the scope

For this paper, we limited the data used for training and evaluation to GML texts from only one dialect region and the same time. In future work, we will expand the scope of variant detection. We will add data from other dialect regions and time spans, which will add dialectal and diachronic variation as well – which is common for historical corpora that contain heterogeneous texts.

We will also extend the experiments to other kinds of non-standard data, especially CMC texts. We did first experiments with Twitter data using the data of the W-NUT 2015 normalization shared task (Baldwin et al., 2015) and treated all types that are normalized with the same type as spelling variants. Table 5 shows results for generating candidate pairs using the Levenshtein distance. There is a difference when comparing these results to the results obtained for the GML data (see Table 1): while the number of candidates generated using the Levenshtein distance is higher, at the same time the recall is lower. Therefore, we plan to experiment with other ways of generating candidate pairs in order to reduce the number of pairs that have to be classified, e.g., by using a distributional thesaurus (Riedl and Biemann, 2013). A surface

based way to improve the recall is to use rules to simplify the non-standard words, e.g. by reducing the number of character repetitions to a maximum of 3 (Han and Baldwin, 2011) thereby detecting spelling variants like {*loool*, *looooooooool*}.

Another difference between contemporary CMC texts and historical data is the amount of text that is available. Therefore, e.g., using context representations should give better results for this type of data than for the GML data.

In this paper, spelling-variant detection has been approached on the type level. However, there is variation on the token level as well (Jurish, 2010b). For example, the dative singular of the name *Maria* appears as *maria* and as *marien* in the GML data, whereas the nominative singular only appears as *maria*. Therefore, *marien* is a spelling variant of *maria* in *do he comen was to maria* 'when he came to Maria', but not a spelling variant for *maria* in *maria hett gesegt* 'Maria said'. One way to approach spelling-variant detection on the token level is to rank spelling variants generated for the type. One possible starting point for this is the system presented by Roark and Sproat (2014) to expand abbreviations. Similarly to the approach presented here, one of their models uses an SVM to evaluate possible expansion candidates. This model also includes features related to the token context, as the abbreviation expansion is done for specific tokens.

Furthermore, for the experiments presented here, we used texts that have been tokenized manually. This removed spelling variation that involves white space. E.g. the GML word for 'kingdom' appears as *koninckryke*, *konnick ryke* and *konynck ryke* in the texts. In order to detect this kind of variation, tokenization has to be combined with spelling-variant detection or spelling-variant detection has to be extended to token n-grams.

5.3 Applications

Apart from an intrinsic evaluation of spelling-variant detection as in this paper, we will also evaluate it extrinsically. Next to the approaches that use detected spelling variants to improve the accuracy of POS tagging and lemmatization, we will employ spelling variants in other tasks.

One of these tasks is normalization. While we presented spelling-variant detection as an alternative to normalization in the absence of an existing standard, it should also be usable to complement normalization as normalization has to deal with spelling variation in non-standard words while mapping these to standard words. For instance, Jin (2015) uses an approach for normalization, where for non-standard words, firstly, normalization candidates are generated, and, secondly, the most probable of these candidates is selected. The candidate generation used in the original approach cannot generate the correct candidate for spelling variants of non-standard words that did not appear in the training data, e.g., *you are* as a normalization candidate for *urr* will not be generated if only *ur* as a non-standard variant of *you are* is known from the training data. Similarly to the approach that Barteld et al. (2016) used to improve the lemmatization of non-standard texts, the knowledge that *urr* is a spelling variant of *ur* could be used to generate the candidate *you are* and thereby improve the coverage of the generator.

Detected spelling variants could also be used as the basis for an artifical standard that can then be used as the target for normalization, where no standard exists. A simple first approach for this would be to transform the spelling variant relation into a clustering by using the symmetric transitive closure and take the most frequent form for each cluster as the standard form.

Another use case that we are interested in is the detection of annotation errors in a corpus. One approach to do this is to use variation n-grams (Dickinson and Meurers, 2003) to detect potential errors. In this approach, variation in the annotation of identical n-grams is used to detect annotation errors. Spelling variation, however, leads to the situation that two identical n-grams on the level of the morphological word appear as different n-grams on the observable type level (cf. Example 2) which affects the recall of this approach. We want to employ spelling-variant detection to identify n-grams that are identical on the level of the morphological word but differ on the type level before employing the variation n-gram method for annotation error detection.

Acknowledgements

This work has been supported by the German Research Foundation (DFG), grant SCHR 999/5-2. I would like to thank the anonymous reviewers for their helpful remarks.

References

Sreangsu Acharyya, Sumit Negi, L. V. Subramaniam, and Shourya Roy. 2009. Language independent unsupervised learning of short message service dialect. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(3):175–184.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135. Association for Computational Linguistics.

Alistair Baron, Paul Rayson, and Dawn Archer. 2009. Word frequency and key word statistics in historical corpus linguistics. *Anglistik: International Journal of English Studies*, 20(1):41–67.

Fabian Barteld, Ingrid Schröder, and Heike Zinsmeister. 2015. Unsupervised regularization of historical texts for POS tagging. In *Proceedings of the Workshop on Corpus-Based Research in the Humanities (CRH)*, pages 3–12.

Fabian Barteld, Ingrid Schröder, and Heike Zinsmeister. 2016. Dealing with word-internal modification and spelling variation in data-driven lemmatization. In *Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 52–62. Association for Computational Linguistics.

Marcel Bollmann, Stefanie Dipper, Julia Krasselt, and Florian Petran. 2012. Manual and Semi-automatic Normalization of Historical Spelling—Case Studies from Early New High German. In *Proceedings of the 11th Conference on Natural Language Processing (KONVENS 2012), LThist 2012 Workshop*, pages 342–350.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2(4):597–620.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27.

Maria Alina Ciobanu and Liviu P. Dinu. 2014. Automatic detection of cognates using orthographic alignment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 99–105. Association for Computational Linguistics.

Maria Alina Ciobanu and Liviu P. Dinu. 2015. Automatic discrimination between cognates and borrowings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 431–437. Association for Computational Linguistics.

Markus Dickinson and Detmar W. Meurers. 2003. Detecting errors in part-of-speech annotation. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 107–114.

Stefanie Dipper. 2011. Morphological and Part-of-Speech Tagging of Historical Language Data: A Comparison. *Journal for Language Technology and Computational Linguistics (JLCL)*, 26(2):25–37.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369. Association for Computational Linguistics.

Gerd Fritz. 2006. *Historische Semantik*. Metzler, Stuttgart and others, 2nd edition.

Phani Gadde, L. V. Subramaniam, and Tanveer A. Faruquie. 2011. Adapting a WSJ trained part-of-speech tagger to noisy text: Preliminary results. In *Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data*, pages 5:1–5:8.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47. Association for Computational Linguistics.

Luís Gomes and José G. P. Lopes. 2011. Measuring Spelling Similarity for Cognate Identification. In Luis Antunes and H. Sofia Pinto, editors, *Progress in Artificial Intelligence*, Lecture Notes in Computer Science 7026, pages 624–633. Springer, Berlin, Heidelberg.

Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 82–90. Association for Computational Linguistics.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 368–378. Association for Computational Linguistics.

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

Alexander Hogenboom, Daniella Bal, Flavius Frasincar, Malissa Bal, Franciska De Jong, and Uzay Kaymak. 2015. Exploiting Emoticons in Polarity Classification of Text. *Journal of Web Engineering*, 14(1-2):22–40.

Ning Jin. 2015. Ncsu-sas-ning: Candidate generation and feature engineering for supervised lexical normalization. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 87–92. Association for Computational Linguistics.

Bryan Jurish. 2010a. Comparing canonicalizations of historical german text. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 72–77. Association for Computational Linguistics.

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

Mike Kestemont, Walter Daelemans, and Guy De Pauw. 2010. Weigh your words—memory-based lemmatization for Middle Dutch. *Literary and Linguistic Computing*, 25(3):287–301.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing sms: are two metaphors better than one ? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 441–448. Coling 2008 Organizing Committee.

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

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association of Computational Linguistics*, 3:211–225.

Nikola Ljubešić, Tomaž Erjavec, and Darja Fišer. 2014. Standardizing Tweets with Character-Level Machine Translation. In Alexander Gelbukh, editor, *15th International Conference CICLing 2014, Proceedings, Part II*, Lecture Notes in Computer Science 8404, pages 164–175. Springer, Berlin, Heidelberg.

Pavel Logačev, Katrin Goldschmidt, and Ulrike Demske. 2014. POS-tagging historical corpora: The case of Early New High German. In *Proceedings of the Thirteenth International Workshop on Treebanks and Linguistic Theories (TLT-13)*, pages 103–112.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR) 2013, Workshop Track*.

Hermann Niebaum. 2000. Phonetik und Phonologie, Graphetik und Graphemik des Mittelniederdeutschen. In *Sprachgeschichte. Ein Handbuch zur Geschichte der deutschen Sprache und ihrer Erforschung*. De Gruyter, Berlin, Boston, 2nd edition.

Robert Peters and Norbert Nagel. 2014. Das digitale ‚Referenzkorpus Mittelniederdeutsch / Niederrheinisch (ReN)‘. *Jahrbuch für Germanistische Sprachgeschichte*, 5(1):165–175.

Michael Piotrowski. 2012. *Natural Language Processing for Historical Texts*. Synthesis Lectures on Human Language Technologies 17. Morgan & Claypool Publishers.

Martin Reynaert. 2009. Parallel identification of the spelling variants in corpora. In *Proceedings of the Third Workshop on Analytics for Noisy Unstructured Text Data 2009*, pages 77–84.

Martin Riedl and Chris Biemann. 2013. Scaling to large3 data: An efficient and effective method to compute distributional thesauri. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 884–890. Association for Computational Linguistics.

Brian Roark and Richard Sproat. 2014. Hippocratic abbreviation expansion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369. Association for Computational Linguistics.

Yves Scherrer and Tomaž Erjavec. 2016. Modernising historical Slovene words. *Natural Language Engineering*, 22(6):881–905.

Klaus Schulz and Stoyan Mihov. 2002. Fast string correction with Levenshtein-automata. *International Journal of Document Analysis and Recognition*, 5:67–85.

Irina Sergienya and Hinrich Schütze. 2015. Learning better embeddings for rare words using distributional representations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 280–285. Association for Computational Linguistics.

Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Ian H. Witten, Eibe Frank, and Mark A. Hall. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Burlington, MA, 3rd edition.

Yi Yang and Jacob Eisenstein. 2016. Part-of-speech tagging for historical english. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1318–1328. Association for Computational Linguistics.