Developing a Stemmer for German Based on a Comparative Analysis of Publicly Available Stemmers

Leonie Weißweiler **Alexander Fraser**
Center for Information and Language Processing
LMU Munich
weissweiler@cis.lmu.de fraser@cis.lmu.de

Abstract

Stemmers, which reduce words to their stems, are important components of many natural language processing systems. In this paper, we conduct a systematic evaluation of several stemmers for German using two gold standards we have created and will release to the community. We then present our own stemmer, which achieves state-of-the-art results, is easy to understand and extend, and will be made publicly available both for use by programmers and as a benchmark for further stemmer development.

1 Introduction

1.1 The stemming task

In Information Retrieval (IR), an important task is to not only return documents that contain the exact query string, but also documents containing semantically related words or different morphological forms of the original query word. (Manning et al., 2008, p.57)

This is achieved by a stemming algorithm.

> A stemming algorithm is a computational procedure which reduces all words with the same root (or, if prefixes are left untouched, the same stem) to a common form, usually by stripping each word of its derivational and inflectional suffixes (Lovins, 1968).

Thus, the purpose of a stemmer is not to find the morphologically correct root for a word, but merely to reduce it to a form it shares with all words that are sufficiently semantically related to be considered relevant to a search engine query for one of them. The exact nature of that form is irrelevant.

1.2 Motivation

Stemming for German is naturally a task that attracts less attention then stemming for English. There are, however, a number of different available stemmers for German, the most popular of which are the Snowball German stemmer, developed by the team of Martin Porter, and the stemmer developed by Jörg Caumanns (Caumanns, 1999). Available stemmers are fairly different in terms of their algorithms and their approaches to stemming, with solutions ranging from recursive stripping of just a few characters to identifying prefixes and suffixes from a pre-compiled list. Of all the stemmers presented here the Snowball stemmer is the only one for which an official implementation is available. For the others, the implementations that are used in NLP toolkits are by third parties, and, as we will show, sometimes contain flaws not intended by the original authors.

At the same time, we are not aware of any comprehensive evaluation of stemmer performance for German. The main goal of this paper is therefore to supply such a study in order to enable NLP programmers to make an informed choice of stemmer. We also want to improve existing stemmers and therefore present a new state-of-the-art stemmer, which we will make freely available in Perl, Python and Java. So a secondary goal is to make a clean and simple implementation of our stemmer available for programmers. Finally, we will release the two gold standards we have developed, which can act as a benchmark for future stemmer development work.

1.3 Summary of Work

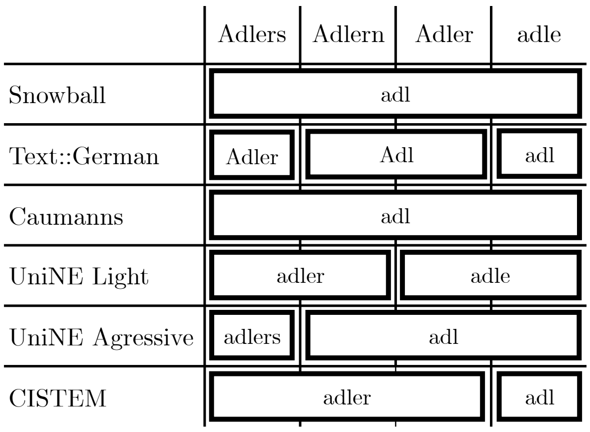
We looked at five available stemmers for German and compared their algorithms.

We then automatically compiled two different gold standards from the morphological information in the CELEX2 data (Baayen et al., 1995) for German. They aim to represent two slightly different opinions on what stemming should be. One was compiled by stripping away morphemes that had not been assigned their own wordclass and the other using the wordform to lemma matching in the CELEX2 data.

We then evaluate the stemmers on the two gold standards, computing precision, recall and f-measure in a cluster-based evaluation that evaluated performance based on which words were stemmed to the same stem (and not how the stems actually looked, which is not relevant in most applications of stemming, as we discussed above).

Based on the results of our evaluation, we developed a new stemmer called CISTEM which is simpler and more aggressive than the previously existing stemmers. We show that CISTEM performs better than the previously existing stemmers.

2 Existing stemmers for German and related work



	Adlers	Adlern	Adler	adle
Snowball	adl	adl	adl	adl
Text::German	Adler	Adl	Adl	adl
Caumanns	adl	adl	adl	adl
UniNE Light	adler	adler	adle	adle
UniNE Agressive	adlers	adl	adl	adl
CISTEM	adler	adler	adler	adl

Figure 1: Comparison of all stemmers using the words ”Adler” (eagle), ”Adlers” (eagle, genitive case), ”Adlern” (eagles, dative case), ”adle” (inflected form of ”to ennoble”)

2.1 German Stemmers

In this section we provide an overview of the German stemmers that we studied, briefly outlining their availability and the algorithms used. We show the differences between them with the example shown in Figure 1, where we stemmed the word ”Adler” (eagle). We show the stem produced and the other words reduced to the same stem for each stemmer. All stemmers except Text::German have the same preprocessings steps which are lowercasing the word and replacing umlauts with their normalized vowel versions (e.g., ü is replaced with ue). These steps will therefore not be mentioned below.

2.1.1 Snowball

In 1996, Martin Porter developed the Snowball stemmer for English (Porter, 1980). It became by far the most widely used stemmer for English. The Snowball team has developed stemmers for many European languages, which are included as a set in important natural language processing toolkits such as NLTK (Bird et al., 2009) for Python or Lingua::Stem for Perl.

The Snowball German stemmer is an adaptation of the original English version and thus restrains itself to suffix-stripping. It defines two regions R1 and R2, where R1 ”is the region after the first non-vowel following a vowel, or is the null region at the end of the word if there is no such non-vowel” and R2 is defined in the same way, with the difference that the definition is applied inside of R1. After defining R1 and R2 Snowball deletes a number of suffixes if they appear in R1 or R2. It does not do this recursively but instead in three steps, in each of which at most one suffix can be stripped. The first two steps strip fairly common suffixes like ”ern” or ”est”, while the third step strips derivational suffixes, e.g. ”isch” or ”keit”, which are fairly uncommon.

In our example, the Snowball stemmer correctly places ”Adlers” (eagle, genitive case), ”Adlern” (eagles, dative case) and ”Adler” (eagle) together in the stem ”adl”. However, it also incorrectly stems ”adle”, which is the first person singular of ”adeln” (to ennoble) to ”adl”. This is because the length restriction on how short stems can become is defined in terms of R1 and R2, as explained above, and in this example, R1 for all four words is the part after ”adl”.

2.1.2 Text::German

The stemmer in the Perl CPAN Module Text::German was, as far as we could find out, developed in 1996 at the Technical University of Darmstadt by Ulrich Pfeifer, following work by Gudrun Putze-Meier for which no reference is available. It is not currently actively supported. We made a number of efforts to contact both scientists but were unsuccessful.

What sets Text::German apart from the other stemmers examined here is the fact that it strips prefixes, and that it uses small lists of prefixes, suffixes and roots to identify the different parts of a word. Although the implementation in CPAN has significant flaws, the idea is novel and produced good results, as can be seen in section 3.3.

While the behaviour of Text::German is at times difficult to understand due to its binary-encoded rules, we think that its performance on our example is primarily due to two factors. One is that ”ers” is not in its list of suffixes, which is why ”Adlers” is stemmed to itself. The other is that it does not lowercase stems, which results in ”adle” (correctly) being stemmed seperately.

2.1.3 Caumanns

The stemmer proposed by (Caumanns, 1999) is unique in two ways. One is that it uses recursive suffix stripping of the character sequences ”e”, ”s”, ”n”, ”t”, ”em”, ”er” and ”nd”, which are the letters out of which every declensional suffix for German is built. The other is that it strips ”ge” before and after the word, which makes it one of the two stemmers that stem prefixes. It also substitutes ”sch”, ”ch”, ”ei” and ”ie” with special characters so they are not separated and replaces them back at the end of the stemming process.

In our example, the Caumanns stemmer conflates all four words to the same stem ”adl”. This is because of the recursive suffix stripping and because its length constraint is not producing words shorter than three characters, which is why ”adle” was stemmed to ”adl” which is exactly three characters long.

2.1.4 UniNe

The UniNE stemmer, developed by (Savoy, 2006) from the University of Neuchatel in 2006, has an aggressive and a light stemming option.

Light Option The light option merely attempts to strip plural morphemes. After the standard Umlaut substitutions, it strips one of ”nen”, ”se”, ”e” before one of ”n”,”r” and ”s” or one of ”n”,”r” and ”s” at the end of the word. As only one of these options can take effect, it is a very conservative stemmer.

In the ”Adler” example, the stemmer stems ”Adlers” and ”Adlern” to ”adler” and ”Adler” and ”adle” to ”adle”. It does not go further because it removes at most two letters and doesn’t strip suffixes recursively.

Aggressive Option The aggressive option goes through a number of suffix stripping steps, which always depend on the length of the word. The difference with the other stemmers is that UniNE has two groups of stripping operations and at most one out of each group is executed. Also, its conditions for stripping ”s” and ”st” are very similar to those of the Snowball stemmer, which defines a list of consonants that are valid s- and st-endings respectively and have to occur before the ”s” or ”st” so that the consonant in question is stripped.

This stemmer’s main problem in our example is that it stems ”Adlers” to itself because ”r” is not included in its list of valid s-endings which have to occur before ”s” for it to be stripped.

2.2 Evaluation studies

The literature on the comparative evaluation of stemmers for German is relatively sparse. (Braschler and Ripplinger, 2003) compared the NIST stemmer and the commercial Spider stemmer with two baselines of simply not stemming and morphological segmentation based on unsupervised machine learning and morpho-syntactic analysis. They found precision improvements of up to 23 percentage points and recall improvements of up to 12 percentage points for the NIST stemmer over no stemming compared to 20 percentage points improvement in precision and 30 percentage points in recall for the commercial Spider Stemmer. (Savoy, 2006) tested their UniNE stemmer and the Snowball German stemmer in an information retrieval system and found that the UniNE stemmer improved the Mean Average Precision (MAP) by 8.4 percentage points while the Snowball stemmer improved it by 12.4 percentage points against a baseline without any stemming.

Our evaluation is based on two gold standards which we will make publicly available, allowing them to act as a benchmark for future work on German stemming.

3 Evaluation

3.1 Runtime Analysis

The runtimes that can be seen in table 3.1 are averaged over 10 runs of each stemmer. The Snowball implementation used was our own implementation in Perl which we did in order to better compare the Snowball stemmer to the others (it should be noted that the official implementation of the stemmer is in Martin Porter’s own programming language Snowball, compiled to C code, which will therefore, in practice, be much faster than imple-

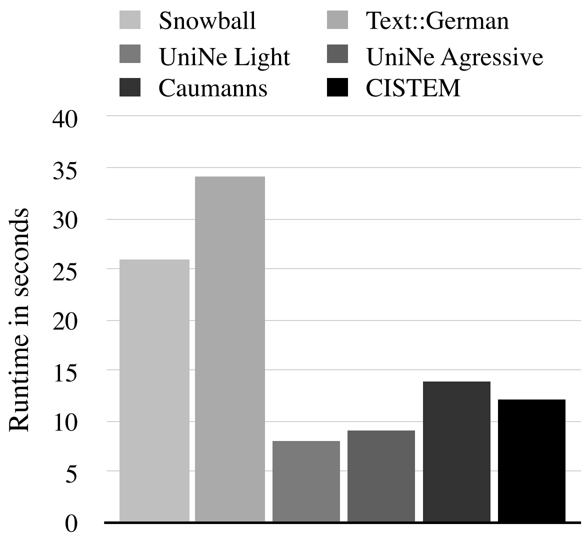


Figure 2: A comparison of stemmer runtimes. 624029 words were stemmed by each stemmer using a single threaded Perl 5.8.18 program on a Xeon x7560 2,26 GHz Processor running openSUSE

mentations in Perl). For the UniNE stemmer, we used the implementation in the CPAN module Lingua::Stem::UniNE::DE, with slight modifications of our own with regards to the use of a module, and for the Caumanns Stemmer we used our own Perl implementation, which was fairly difficult to implement because the paper (Caumanns, 1999) doesn't clearly state a definitive algorithm, instead describing main ideas and then making suggestions for improvements.

To assess average runtime, we then stemmed a corpus of 624029 words on each stemmer using a single threaded Perl 5.8.18 program on a Xeon x7560 2,26 GHz running openSUSE ten times and computed the mean runtime. As can be seen in the table, the runtimes of the Caumanns, UniNE and CISTEM stemmers are fairly similar, while Snowball takes about twice and Text::German nearly three times as long.

3.2 Gold standard development

We compiled two different gold standards. The reason for this is that exactly which words belong to the same stem is something that is difficult for people to agree on. The question of whether, for example, ”billig” (cheap) belongs together with ”billigen” (to approve) seen from an IR perspective, is a difficult one because the adjective ”billig” also exists in the sense of ”something worthy of approval”. Therefore, our hope is that the two gold standards will capture the different ends of this spectrum where one end, when in doubt, puts words in a cluster together and the other doesn't. Having two gold standards capturing this distinction enables us to be more objective in our evaluation.

For the first gold standard, we used the morphological information in CELEX 2. It gives the flat segmentation into morphemes and annotates each morpheme with its word class, and X if no word class applies. This should be equivalent to the distinction between lexical and grammatical morphemes. We then stripped the morphemes annotated with X to form the stem. For the second gold standard, we simply used the fact that every wordform in CELEX2 is assigned a lemma, and used that lemma as the wordform's stem. In each case, we then grouped the wordforms by stem according to the principle that the exact stem is irrelevant as long as the cluster makes sense. The resulting gold standards are 30951 stems large in the case of gold standard 1 and 47852 stems for gold standard 2. From each, we took a random sample of 1000 stems and used those as gold standards for our evaluation. To avoid overfitting, we changed the samples several times while developing CISTEM, including after the end of development for the final evaluation.

As you can see in Figure 3.2, there are differences between the gold standards. For the ”absurd” example, gold standard 2 classified ”absurd” as a different lemma than ”absurditäten” (absurdities) and thus put them in two seperate stems while gold standard 1 sees them as having the same stem. The difference is even more pronounced in the second example, where the first gold standard has one stem for ”relativier” (relative), one for ”Relativismus” (a theory in philosophy), one for ”Relativität” (the general noun for relative) and one for ”relativistisch” (relative, but only in the context of Einstein's theory of relativity).

From an information retrieval point of view, one would consider ”Relativismus” and ”relativistisch” as belonging in one stem that relates to the theory of Relativity, and the other two stems as belonging in another stem. Overall, gold standard 2 is much more likely to seperate words into several different stems while gold standard 1 is more likely to group them into a single stem. This makes sense considering gold standard 2 thinks in lemmata, e.g. in a dictionary one would like to have seperate entries for ”Relativismus” and ”Relativität” while gold standard 1 groups them together because neither ”ismus” nor ”tät” are lexical stems that can be assigned a word class.

This confirms our hopes that the two gold standards would capture two ways of looking at stemming. Gold standard 1 represents a more aggressive-stemming-friendly view and gold standard 2 a more conservative one. Personally, we consider gold standard 1 on the whole to be more suitable for stemmer evaluation, but arguments could also be made for the opposite point of view. For this reason, both gold standards are included in the following evaluation.

3.3 Evaluation

We stemmed the Celex2 corpus. We then went through each of the stems from the gold standard (1000 stems large) and matched them with a stem from the stemmed corpus depending on how many of the words belonging to these two stems matched. For each stem of the gold standard, we computed precision, recall and f1-measure and then computed the average of each of those metrics to form the overall evaluation results for that gold standard. The results can be seen in table 2 and are illustrated in Figures 3(a), 3(b) and 4.

3.4 Results

The most surprising result of our evaluation was that the difference between the two gold standards was not as pronounced as we expected, considering that they represent the two ends of the spectrum of what one wants a stemmer to do. The two gold standards agree on the best stemmer and the worst stemmer in terms of precision, recall and f-measure. As can be seen in Figure 4, these measures differ for the three middle ranked stemmers Snowball, Text::German and UniNE Aggressive.

The difference between the two gold standards is shown most clearly in their assessment of the Snowball stemmer. This is to be expected as the stripping of clearly derivational suffixes like ”lich” or ”ung” matches the stemming concept of gold standard 1 quite closely, where suffixes like these suffixes are removed. This explains why, while Snowball achieved the lowest precision on both gold standards, the gap to the next best precision is much lower in gold standard 1 (just 0.59 percentage points) than in gold standard 2 (6.37 percentage points), where Snowball’s aggressive stemming is much more likely to affect precision negatively.

Being one of the more conservative stemmers, Text::German scores significantly higher on gold standard 2. On gold standard 1, it achieves fourth place in both recall and f-measure and third place in precision of the existing stemmers. We attribute this mainly to the fact that Text::German stems at most one suffix from a small list which doesn’t include derivational suffixes, which is guaranteed to hurt recall on gold standard 1 because gold standard 1 requires more than just very conservative suffix stripping. This same fact results in a relatively good score on gold standard 2, achieving the second place (when compared with previously existing stemmers) in all metrics. We think that the main problem that hurts its performance on both gold standards should be that Text::German identifies prefixes from a list, nearly all of which are clearly lexical, and strips them according to a complicated set of rules. From an IR standpoint, the stripping of lexical prefixes which clearly change the word’s meaning is suboptimal.

The Caumanns stemmer achieves first place (with respect to already existing stemmers) in both gold standards in recall and f-measure. The gap to the other stemmers’ values is about 3 percentage points on both gold standards. An interesting point here is that while precision is significantly higher than recall on gold standard 1, the opposite is true for gold standard 2. This points to the Caumanns stemmer having achieved a middle line between both gold standards’ concepts of stemming. The stemmer is more conservative than gold standard 1 and more radical than gold standard 2. This, together with the large gap in performance to the competitors, makes the Caumanns stemmer the best stemmer for German we have seen so far.

The light option of the UniNE stemmer, as expected, scores last in recall and f-measure while having the highest precision on both gold standards. The gap between precision and recall is larger in gold standard 1 (more than 30 percentage points) than in gold standard 2 (more than 25 percentage points). As the express goal of the light option is to merely strip affixes denoting plural, the lack of recall is naturally more pronounced in gold standard 1 because it requires stemmers to be more radical in order to score well. The overall bad performance is not surprising, as stemming entails more than just stripping plural suffixes.

The agressive option of the same stemmer, on the other hand, achieves mediocre results, com-

Gold standard 1	**Gold standard 2**
• absurderen absurdestem [...] absurditäten absurdität	• absurderen absurdestem absurder absurden [...] • absurditäten absurdität
• relativem relatives [...] relativistischerer [...] relativität relativitäten	• relativieret relativiertest [...] • relativität relativitäten • Relativismus • relativistischsten relativistischen [...]

Table 1: Two examples for the differences between the two gold standards

Table 2: Evaluation results of different stemmers using our two gold standards, each of which is for the same 1000 stems (note that CISTEM is our new stemmer which will be introduced later in the paper)

Gold standard 1

Stemmer	Snowball	Text::German	Caumanns	UniNE Light	UniNE Aggressive	CISTEM
Precision	96.17%	97.56%	96.76%	**98.39%**	97.37%	96.83%
Recall	83.78%	79.29%	89.43%	67.69%	80.29%	**89.73%**
F1	89.55%	87.48%	92.95%	80.20%	88.01%	**93.15%**

Gold standard 2

Stemmer	Snowball	Text::German	Caumanns	UniNE Light	UniNE Aggressive	CISTEM
Precision	85.89%	96.00%	92.26%	**96.43%**	94.50%	92.43%
Recall	86.61%	86.97%	96.17%	70.91%	83.81%	**96.45%**
F1	86.25%	91.27%	94.17%	81.72%	88.83%	**94.40%**

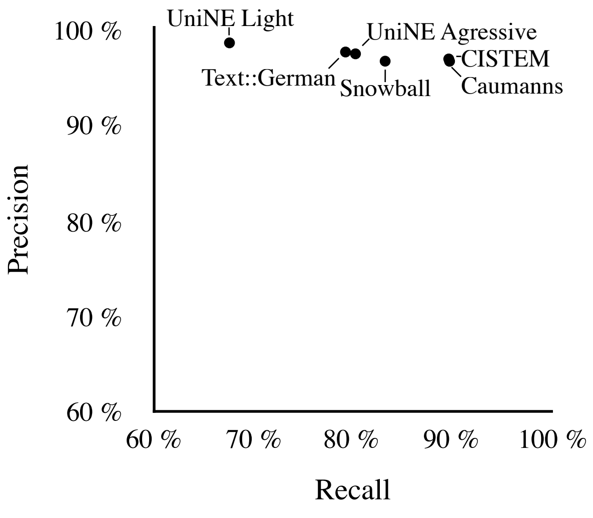
ing in third place in f-measure on both gold standards. While it does strip suffixes in several steps, it doesn't do so recursively, which is why it makes sense that the performance is about as good as the Snowball stemmer, which has a similar approach. It also explains that the performance is somewhere in the middle of all the existing stemmers as we have seen that recursive suffix stripping in general performs best and one-time stripping worst, because UniNE Aggressive's approach is located in between these two ideas.

The clearest lesson to be drawn from this analysis is that the problem of existing stemmers is in recall. Precision is relatively similar for every stemmer, only varying by 2.22 percentage points while recall varies by 21.04 percentage points in gold standard 1. The discrepancy is similarly pronounced in gold standard 2, where precision varies by 10.54 percentage points and recall by 25.54 percentage points. Because of the nature of f1-measure, recall therefore decides the stemmer's f-measure ranking: in gold standard 1, the recall order from best to worst exactly mirrors that of f-measure and in gold standard 2, only the positions of Snowball and UniNE Aggressive, the two middle stemmers, are reversed. The mean precision in gold standard 1 is 97.13% and the mean recall is 82.04%. The mean precision in gold standard 2 is 92.62% and the mean recall is 87.45%. Not only does recall vary much more, it is also consistently much lower than precision.

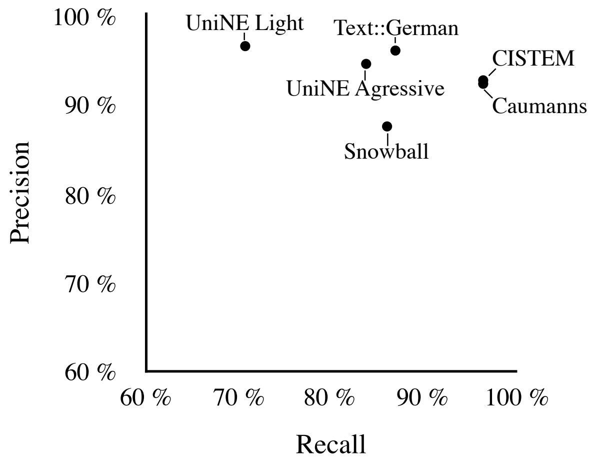
4 Development

4.1 CISTEM development

Following the insight that recall is the most promising area for stemmer development, we focused on improving recall over existing stemmers with CISTEM. As starting point, we used the Caumanns stemmer, as it was the best performing stemmer of our evaluation, and tried to improve on it. We tried several changes and evaluated each of them seperately to improve f-measure. One feature of the Caumanns stemmer that we deleted was the substitution of ”z” for ”x”, which improved precision slightly in



(a) Gold standard 1



(b) Gold standard 2

Figure 3: Precision- Recall values on the two gold standards

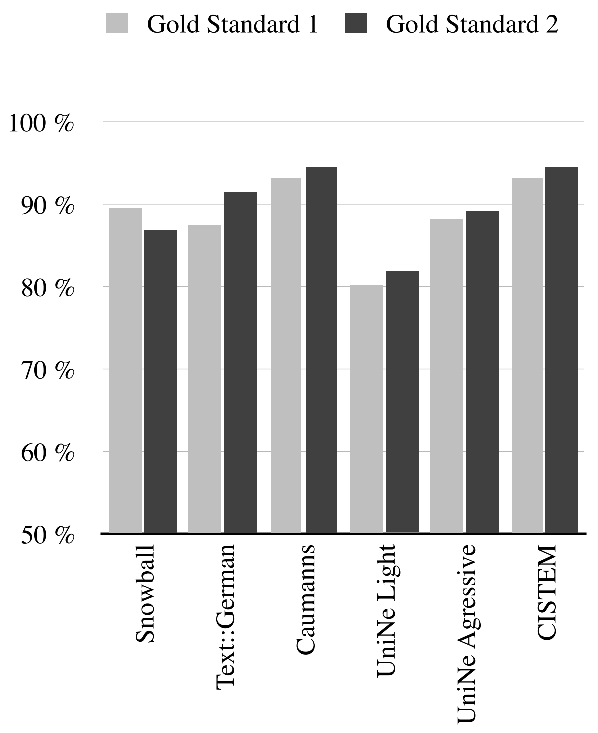


Figure 4: F1-measures on both gold standards

gold standard 1 and changed no other metrics. We also found that stripping ”ge” before and after the word after the suffix stripping, as proposed by Caumanns, didn’t work well. The version of this that delivered the best performance was stripping ”ge” as a prefix, before the suffix stripping and only if the remaining word is at least four characters long. This is consistent with the requirement for suffix stripping that the resulting word needs to be at least four characters long, while the Caumanns stemmer undercuts that requirement by removing ”ge” after the suffix stripping without checking the length of the result. Interestingly, introducing a new variable that measures true length (necessary because substitutions of multiple characters by one character, e.g. ”ei” by ”%” make the word shorter than it actually is) hurt performance quite clearly. We also deleted the substitution of ”ch” by ”§” because we found it hurt recall on gold standard 2 and changed nothing on gold standard 1. The length constraint on the stripping of ”nd”, which was at least five remaining characters in the Caumanns stemmer, was changed to at least six, which doesn’t only improve performance but also makes the algorithm simpler as ”nd” is now stripped in the same step as ”em” and ”er”. Our other contribution was to give the steps a definitive order, which had not been clear in the Caumanns paper and led to subtle flaws in third-party implementations we tried.

The resulting algorithm, which can be seen in Figure 5, is simpler than the Caumanns stemmer and easy to understand and implement. We will also offer a context-insensitive version which ignores case for ”t”-stripping because the original Caumanns stemmer’s performance is drastically worse when using a corpus of only lowercase words, which might be necessary in some contexts, but would lead to the stemmer never stripping ”t”.

4.2 Final Evaluation

CISTEM shows slight improvements over the Caumanns stemmer in both precision and recall. The difference in recall is more pronounced, which is

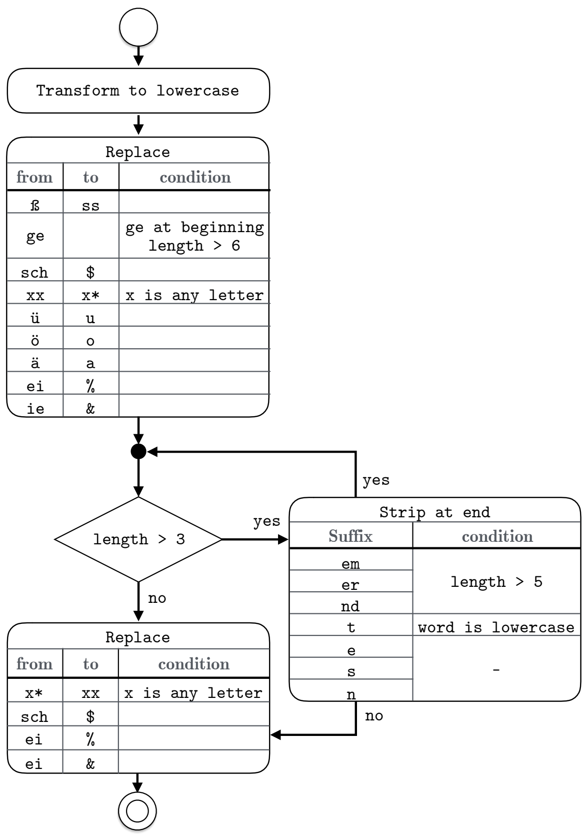


Figure 5: The CISTEM Algorithm

consistent with our goal of removing some constraints of the Caumanns stemmer to improve recall.

If we look back to the example in Figure 1, we can see that CISTEM stems the four words correctly. It stems ”adle” to ”adl”, which is the same stem that Caumanns assigned it, but stems the other three words to ”adler” because the length requirement for stripping ”er” is that the resulting stem will be longer than five characters (not four characters).

The main advantage of CISTEM over other stemmers available is that we have a definitive algorithm shown in Figure 5. The algorithm is bug free, the order is fixed and we will make it available in a range of programming languages to prevent flawed third-party implementations.

We hope that our new stemmer CISTEM will be useful in a wide range of applications. In addition, stemming-based segmentation of German has recently been shown to be effective in reducing vocabulary in neural machine translation (Huck et al., 2017). So we will additionally provide a version of the algorithm which segments words, rather than stemming them.

5 Conclusion

We presented two gold standards for stemming which represent two different views on stemming. We then evaluated five existing stemmers for German on those gold standards and discussed the results. Finally, we presented our own stemmer, which improves on the stemmer of Caumanns and achieves state-of-the-art results on both gold standards.

One of the main problems in stemmer development is the divide between the stemmers that are published and those that are actually used in NLP applications. The Snowball stemmer continues to be most widely used because it is the default stemmer for most NLP libraries and offers stemmers for a wide range of European languages.

For this reason, we will publish official implementations in a range of programming languages, starting with Perl, Python and Java. We are also planning to release our gold standards in the hope that they will be used in further work on stemming for German. The code and gold standards will be made available at `https://www.github.com/LeonieWeissweiler/CISTEM`, and we hope to also be included in some standard NLP packages in the future. Other future work would be to find other ways of building a gold standard for stemming in order to have one definitive gold standard where words are clustered exactly as they should be for information retrieval. We were often obstructed in our development by having to show improvements in both gold standards for every change, which could be avoided by having just one gold standard. Another more unconventional idea would be implementing a small rule-learning system that suggests new rules for the stemmer based on their effectiveness in matching a gold standard or when used actively in a working IR system.

Acknowledgments

This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 644402 (HimL). This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 640550).

References

Baayen, R. H., Piepenbrock, R., and Gulikers, L. (1995). *The CELEX Lexical Database (Release 2) on [CD-ROM]*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Sebastopol, CA.

Braschler, M. and Ripplinger, B. (2003). Stemming and decompounding for German text retrieval. In Sebastiani, F., editor, *Advances in Information Retrieval: 25th European Conference on IR Research*, ECIR 2003, pages 177–192, Berlin. Springer.

Caumanns, J. (1999). *A Fast and Simple Stemming Algorithm for German Words*. Technical Report Nr. tr-b-99-16. Freie Universität Berlin, Fachbereich Mathematik und Informatik.

Huck, M., Riess, S., and Fraser, A. (2017). Target-side word segmentation strategies for neural machine translation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, Copenhagen, Denmark.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11(1).

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Savoy, J. (2006). Light stemming approaches for the French, Portuguese, German and hungarian languages. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 1031–1035, New York. ACM.