Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)

A Probabilistic Morphology Model for German Lemmatization

Christian Wartena
Hochschule Hannover
Expo Plaza 12, 30539 Hannover, Germany
christian.wartena@hs-hannover.de

Abstract

Lemmatization is a central task in many NLP applications. Despite this importance, the number of (freely) available and easy to use tools for German is very limited. To fill this gap, we developed a simple lemmatizer that can be trained on any lemmatized corpus. For a full form word the tagger tries to find the sequence of morphemes that is most likely to generate that word. From this sequence of tags we can easily derive the stem, the lemma and the part of speech (PoS) of the word. We show (i) that the quality of this approach is comparable to state of the art methods and (ii) that we can improve the results of Part-of-Speech (PoS) tagging when we include the morphological analysis of each word.

1 Motivation

In the following we present a simple approach to lemmatization of German texts and compare the results with a number of other easily available tools.

The motivation was twofold: while lemmatization seems to be a core task in analyzing text, in most standard Python packages like Stanford's Natural Language Toolkit (NLTK), no lemmatization for German is available. Mainly for teaching undergraduate students we wanted to have a simple tool, that gives linguistically correct lemmata for all words and is also easy to use and install on a Python notebook server. In the second place, we wanted to investigate, whether a careful splitting of a word into a stem and suffix can improve the treatment of unknown words in a standard trigram PoS tagger, in which the PoS otherwise is guessed on the base of the final letters of a word.

Lemmatization is a core task in analyzing text. Nevertheless it did not receive as much attention as e.g. PoS tagging. Rule based systems can reach a very high accuracy for morphological analysis. However, existing systems are not always available and the construction of a lexicon and morphological rules is a very tedious task. In practice therefor often simple heuristic rules or a dictionary lookup are used, even if the quality of the tools is not even known. Especially for information retrieval the quality of lemmatization seems not to be very important and some studies suggest that any form of heuristic stemming, mixing up inflectional and derivational morphology can be used (Kettunen et al., 2005; Moral et al., 2014) though some other studies contradict these findings (Braschler and Ripplinger, 2004).

In the following we present an approach to morphological analysis based on computing the most likely sequence of morphemes for a given word. In section 2 we present the details of this method. In subsection 2.4 we show how we can use the results in a PoS tagger. In section 3 we discuss related work and alternative approaches and finally, section 4 compares the results on lemmatization and PoS tagging.

2 Method

Given a word $w = a_1 \dots a_n$ we try to find the most likely sequence of morpheme tags $s = t_1 \dots t_k$ that generates w. We cannot use a standard Hidden Markov Model and the Viterbi algorithm to find the most likely sequence s since we do not have a segmented list of output observations. However, the solution presented here is very similar to a Hidden Markov Model and the computation of the most likely tag sequence is almost identical to the Viterbi Algorithm. We define the most likely tag sequence s that generates w as

$$\max_{k,s \in T^k} \prod_{i=3}^{k} p(t_i \mid t_{i-2}t_{i-1}) \cdot p(a_{l_i m_i} | t_i) \qquad (1)$$

where T is the set of all tags, $0 \leq l_i \leq m_i \leq n$ for each i and $w = a_{l_3 m_3} \cdot \ldots \cdot a_{l_k m_k}$.

Since here every state is dependent on two previous states we call the model second order model and we have to add two start states to every sequence. We also add a final state that generates the empty string to each tag sequence. We add one final state for each PoS tag. This will allow us, to compute the most likely tag sequence for each PoS.

Using dynamic programming we can find the optimal tag sequence efficiently. Using a first order model, for a string $w = a_1 \dots a_n$ we define the probability that a prefix $a_1 \dots a_j$ of w is tagged with a tag sequence in which the last tag is t as

$$\vartheta(t,j) = \max_{r,s,i} \left(\vartheta(s,i) \cdot p(a_{ij} \mid t) \cdot p(t \mid s)\right) \quad (2)$$

for each $t \in T$ and $2 \leq j \leq n$ where $s \in T$, $2 \leq i \leq j$, a_{ij} denotes the substring $a_i \dots a_j$ from w and

$$\vartheta(t,1) = p(t \mid \text{START}). \quad (3)$$

The equation can easily be extended for a second order model but the becomes slightly more complicated. When we compute $\vartheta(t,j)$ we get the well-known Trellis diagram. When we extend the algorithm with a backpointer, we can easily find the optimal tag sequence for a given word.

Consider e.g. the word *Sorgen*, that can either be the plural of the noun *Sorge*, the infinitive of the verb *sorgen*, or a finite form of the same verb. Now, for each of the corresponding final states we can compute the most likely tag sequence that generates the word *sorgen*. In our data we thus find the following tag sequences:

$$s_{\text{inf}} = \text{None}, \text{START}, \text{VV}, \text{SUF\_INF}, \text{END\_VVINF}$$
$$s_{\text{fin}} = \text{None}, \text{START}, \text{VV}, \text{SUF\_FIN}, \text{END\_VVFIN}$$
$$s_{\text{nn}} = \text{None}, \text{START}, \text{NN}, \text{SUF\_NN}, \text{END\_NN}$$

The probability that the sequence s_{nn} generates the word is computed as follows: $p(\textit{Sorgen}, s_{\text{nn}}) = p(\text{NN} \mid \text{None}, \text{START}) \cdot p(\text{SUF\_NN} \mid \text{START}, \text{NN}) \cdot p(\text{END\_NN} \mid \text{NN}, \text{SUF\_NN}) \cdot p(\text{‘sorge’} \mid \text{NN}) \cdot p(\text{‘n’} \mid \text{SUF\_NN}) = e^{-1.60854} \cdot e^{-1.46985} \cdot e^{0.0} \cdot e^{-7.82129} \cdot e^{-1.43789} = e^{-12.33757}$. Similarly, we find $p(\textit{sorgen}, s_{\text{fin}}) = e^{-10.77681}$ and $p(\textit{sorgen}, s_{\text{inf}}) = e^{-10.63024}$.

Since there could be several sequences generating *sorgen* and ending in END_NN, the probability $p(\textit{Sorgen}, s_{\text{nn}})$ is not the probability that the word is generated by a noun sequence. To compute that probability we would need an equivalent of the forward algorithm, that computes the sum of all probabilities leading to one state instead of the maximum. However, in practice alternative paths turn out to be completely nonsense (since the model is highly over generating) or extremely unlikely (and thus do not change anything).

Given the large amount of training data for words, and the fact that for each substring we can assume that it is generated by one of the open class morphemes, we do not need any interpolation or smoothing and use the trigram probabilities directly.

Finally, we use also case information and multiply the found probability with the probability that a word of the found class is capitalized or not.

2.1 Unknown Words

For each string we can assume that it is an unseen instance of an open morpheme class. Currently we defined by hand, which classes are the open classes, but this also can quite easily be guessed from the number of hapax legomena in each class.

For a morpheme $m = a_1 \dots a_n$ we can estimate the probability that an unseen morpheme is generated by a given morpheme tag using the probability that a morpheme ending on a given suffix is generated by that class. We compute these suffixes probabilities on infrequent morphemes, assuming that morphemes not observed in the test data are more similar to infrequent than frequent to morphemes. If not enough observations are available for suffixes of length n we use the probabilities for suffixes of length $n-1$. To compute the probabilities of the shorter suffixes we exclude all morphemes ending on one of the longer suffixes for which we had enough observations. E.g. if we use the probability $p(\text{noun} \mid \textit{ung})$, for bigrams we use $p(\text{noun} \mid \textit{ng} \text{ and not } \textit{ung})$ rather than $p(\text{noun} \mid \textit{ng})$.

For longer unknown words we need to be sure, that an analysis using several known morphemes is preferred over the analysis as one unknown morpheme. Especially long nouns should be much more likely to be a noun compound, consisting of two or more known stems than being a completely unseen stem. Thus we also use the probability $p(n \mid t)$ that a morpheme of length n is generated by t. We compute this probability on infrequent morphemes again. Finally, we use the probability $p_{hap}(t)$ that the tag produces a hapax legomenon. Thus we approximate the probability of an unseen morpheme $m = a_1 \dots a_n$ given a tag t as

$$p(m \mid t) \approx p(a_{n-2}a_{n-1}a_n \mid t) \cdot p(n \mid t) \cdot p_{hap}(t). \quad (4)$$

2.2 Generating training data

The most critical part in the development of the analyzer is the generation of training data. We generate the training data from the Tiger Corpus (Brants et al., 2002). Here we find a lemma and a PoS for each word form. The basic idea now is to split the word form in the stem, that can easily be derived from the lemma, and prefixes and suffixes. E.g. the word *geplant* with lemma *planen* and stem *plan* can be split up in *ge*, *plan* and *t*. For the affixes we assign tags based on the given PoS. Thus, in the present example we generate

```
[('ge', 'PREF_PP'), ('plan', 'VV'), ('t'
    , 'SUF_PP'), ('', 'END_VVPP')]
```

In many cases we end up with much more complicated sequences. We restrict the decomposition of words to inflectional morphology except for noun compounds, comparatives and superlatives of adjectives and adjectives derived from participles. Thus we have sequences like

```
[('auf', 'PTKVZ'), ('ge', 'PREF_PP'), ('
    schreck', 'VV'), ('t', 'SUF_PP'), ('
    ', 'END_VVPP')]
[('amtier', 'VV'), ('end', 'PRESPART'),
    ('en', 'SUF_ADJ'), ('', 'END_ADJA')]
[('ordnung', 'NN'), ('s', 'FUGE'), ('krä
    ft', 'NN_VAR'), ('en', 'SUF_NN'), ('
    ', 'END_NN')]
```

We use several language dependent heuristic rules to split up each word. In German the stem often is not a part of the surface form. In most of these cases we can find a variant of the stem by searching a substring that starts and ends with the same consonants. E.g. for the word *jüngeren* (younger) with stem *jung* (young) we find:

```
[('jüng', 'ADJ_VAR'),('er', 'ADJ_COMP')
    ,('en', 'SUF_ADJ'),('', 'END_ADJA')]
```

In addition now the substitution *jüng/jung* will be stored for the adjective class. These substitutions will be used later to reconstruct the stem and lemma of an analyzed word.

In total we used 52 final tags (i.e. tags encoding the PoS of a word and not corresponding to any morpheme) and 75 real morpheme tags.

The morpheme classes obtained in this way are very rough and result in a massively overgenerating model. E.g. for some verbs the past participle is formed without the prefix *ge*. Thus the model allows the morpheme tag SUF_PP without having seen the morpheme PREF_PP before and independent of the verb, since no distinction is made between verb classes needing the prefix and those that do not have this prefix in the past participle. Thus even a form like *lauft* could be analyzed as `[('lauf', 'VV'), ('t', 'SUF_PP'), ('', 'END_VVPP')]`. However, it turns out that for analyzing there is only a limited number of cases, where this causes problems.

2.3 Lemmatization

Once the most likely sequence of tags is found, a small set of rules is used to generate the correct lemma. These rules mainly deal with the generation of an infinitive from a stem and with the application of the stored substitutions for irregular stems and stems with Ablaut (vowel gradation).

2.4 Part of Speech Tagging

The usual way to analyze German or English is to start with part-of-speech tagging and then to analyze each word according to the found PoS. It is now tempting to investigate, whether the other way around works as well: instead of using observed probabilities we could use the probabilities as computed by the morphological analysis. To be precise: we computed $p(w,t)$ for a word and a tag before. In a standard trigram tagger (see e.g. Brants (2000)) we need the probability $p(w \mid t)$. This probability can be computed easily by using the fact that $p(w \mid t) = \frac{p(w,t)}{p(t)}$.

The approach has the advantage that we get much better statistics for inflectional variants of infrequent words. On the other hand we lose a lot of information on specific word forms. For some words only certain forms are frequently used, and others are infrequent or even not existent. E.g. the noun *Ärger* (trouble, annoyance) does not have a plural form. Consequently, the form *ärgere* only can be a verb form (from *ärgern*, to annoy). Using the morphological analysis described above, we will nevertheless find an analysis as noun as well.

In the following we will use two variants: in the first variant we use only the probabilities computed by the morphological analysis. In the second variant we will use the observed probabilities and use the morphological analysis for words that were not observed in the training data. Here we treat words seen once and twice as unseen words as well.

We base the transition probabilities on trigram statistics over tags. Here we use linear interpolation to avoid zero probabilities and set the smoothed probability $p^*(t_n \mid t_{n-2}t_{n-1}) = 0.95 \cdot p(t_n \mid t_{n-2}t_{n-1}) + 0.04 * p(t_n \mid t_{n-1}) + 0.01 \cdot p(t_n)$.

2.5 Implementation

We implemented all algorithms in pure Python. The script to generate training data from the Tiger corpus and the classes to train and apply the morphological analysis and the PoS tagging are available on Github[1] and PyPI[2].

In order to speed up the computation, we compute the analysis for 2000 frequent words immediately after training and store the results in the model file as well. Here we exclude all analyses resulting in a PoS tag that was never observed for that word. This also slightly improves the results.

In the following we will call this tagger *Hanover Tagger* or short, *HanTa*, and refer to the version using observed probabilities for all words that were seen at least three times in the training data as *HanTa (hybrid)*.

In the package available on GitHub there are functions to analyze a single word, to tag a sentence or to tag and lemmatize a sentence at once. The user can also choose to get only the PoS tag, the lemma, or a morphological analysis.

3 Related Work and Alternative Tools

At first glance lemmatization seems to be an easy task. Nevertheless, for most languages, at least for German, we need some morphological analysis to find correct lemmata. State of the art methods for morphological analysis are still rule based. In the first place here the work of Koskenniemi (1983) has to be mentioned. For German this approach was used in the SMOR tool (Schmid et al., 2004).

Besides the rule based approaches there are several attempts to derive a morphological model for a language in a complete unsupervised way. An example of this approach is Morfessor (Creutz and Lagus, 2007), that in fact uses an underlying model for morphology that is very similar to ours. For a recent overview of unsupervised learning of morphology we refer to (Goldsmith et al., 2017).

Only a few studies deal with the possibility to learn lemmatization or morphology in general from annotated data. Kanis and Müller (2005) and Jongejan and Dalianis (2009) learn rules from a lemmatized corpus to transform an inflected word form to a lemma. Gashkov and Eltsova (2018) obtain good results for German by a full-form dictionary and applying analogy for unknown words: basically, for an unknown word form the word with the longest common suffix is searched and then the transformation associated with that word is applied. Gesmundo and Samardžić (2012) propose to annotate words with the type of rule, needed to transform the full form to a lemma, thus reducing lemmatization to a tagging task. A similar idea is followed by Chrupala et al. (2008) who define classes that correspond to mappings from word forms to lemmata and train a classifier to classify words accordingly. This approach is extended by Müller et al. (2015) who use more features and conditional random fields for classifying morphemes. To some extend our model resembles this approach. The main differences are (i) that we learn on segmented data (and thus have to produce such data before learning) and (ii) that Müller et al. (2015) learn the transformation needed to produce a lemma as well, while we need a small language specific, rule based component that produces a lemma from the list of morphemes found.

Our goal is not to improve on state of the art morphological analysis but just to have an easy tool that gives results that can be used in further tasks and to provide an alternative for lemmatization tools that are easily available and therefor used frequently. In the following we thus compare the results of lemmatization to those obtained by the TreeTagger, Spacy and GermaLemma. For testing PoS tagging we use the same tools and in addition an own implementation of a standard second order Hidden Markov Model, using suffix statistics to guess the output probabilities for unseen words.

The TreeTagger (Schmid, 1999) is a PoS tagger based on a second order Hidden Markov Model (or trigram model) extended with decision trees to use more contextual information and dictionaries of prefixes and suffixes to improve the basic model. The standard model for German was trained on a manually tagged newspaper corpus.

Spacy (ExplosionAI GmbH, 2019) is a state of the art tool based on deep learning for tokenization, PoS tagging and named entity recognition. Spacy also provides lemmatization. While other modules are based on trained artificial neural networks, lemmatization is rule based. We used release 2.1.4.

GermaLemma (Konrad, 2017) is a tool that combines a full form lexicon, extracted from the Tiger Corpus, an algorithm for splitting compounds and morphological rules from the Pattern package (The CLiPS (Computational Linguistics & Psycholinguistics) research center, 2018). GermaLemma

[1]`https://github.com/wartaal/HanTa`

[2]`https://pypi.org/project/HanTa/`

requires that the lemmatization is preceded by PoS tagging.

4 Evaluation

Since our focus is on the development of a practical tool for lemmatization, that can be used as a component in a larger pipeline, we will use large corpora, in which many words occur many times, instead of word lists for evaluation.

4.1 Data

As mentioned before we use the Tiger Corpus for training. We used version 2.2 which consists of $50 \cdot 10^3$ sentences or $0.9 \cdot 10^6$ tokens. For cross validation we split the corpus into 10 contiguous parts, as was also done by Giesbrecht and Evert (2009) and is considered to be a slightly harder and more realistic setting than taking every tenth sentence, since every part now gets sentences from different texts. In addition we use a list of the most frequent verb forms extracted from the DeReKo Corpus (Stadler and Wegstein, 2016) to train the morphology model and to make sure, that at least the most frequent verb stems are seen in the training phase.

For evaluation, besides the Tiger Corpus, we use TüBA D/Z and the Hamburg Dependency Treebank. The Hamburg Dependency Treebank (HDT) (Foth et al., 2014) is very interesting for our purpose since it consists of texts from a different domain. Tiger and TüBA D/Z consist of daily newspaper texts, while HDT uses texts from `heise.de` with news and background articles on anything related to computer hard and software. Here we observe the use of a different vocabulary and sometimes deviations from standard German spelling, like writing compounds as words separated by blanks. We use part A of the corpus, which was manually annotated and checked for consistency. This part consists of 102 000 sentences or $1.87 \cdot 10^6$ tokens. We use HDT for evaluation of PoS Tagging. The lemmata provided cannot be used for evaluation, since for compounds only the head is given as a lemma.

TüBa D/Z (Telljohann et al., 2004) is a manually annotated newspaper corpus of a similar size (104.787 sentences or $1.96 \cdot 10^6$ tokens). TüBA D/Z uses a slightly different tagset than Tiger: TüBA D/Z has a different tag for pronominal adverbs (which we just can replace to compare results) and it distinguishes between two different forms of attributive indefinite pronouns (with and without determiner) while Tiger just has one. Thus, we remove this distinction when evaluating results trained on the Tiger annotation scheme. With regard to the lemmata, TüBa D/Z uses a #-sign to mark the boundary of separable prefixes and sometimes adds disambiguating PoS information to the lemma. Both are removed. In some cases (especially for adjectival nouns) several possibilities for the lemma are listed and separated by a pipe symbol. Here we keep the whole string as it is.

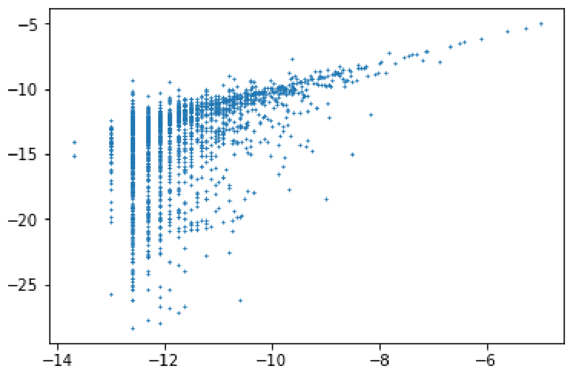

Figure 1: Observed (x-axis) vs. predicted probabilities (y-axis) (here displayed as the natural logarithm of the probabilities) for 2337 word-tag pairs.

4.2 Lemmatization

First, we compare the values of the predicted probabilities with the observed probabilities. For this purpose we take every 10th word of a list of all word forms occurring at least 3 times. This results in a list of 2337 words. For each of these words we compare the probability for the most probable observed tag with the probability estimated for that tag. The Pearson correlation between the two methods is 0.455 indicating a low correlation between the observed and predicted values. Especially for infrequent word forms the estimates are much too low (see Figure 1). This situation is not completely unwanted: we will predict non-zero probabilities for many word forms not present in the corpus. Consequently, some observed probabilities have to become smaller.

4.2.1 Quantitative Analysis

The morphological analysis gives a ranked list of possible PoS tags for each word. We use precision, recall and Mean Reciprokal Rank (MRR) to evaluate these rankings, computed for one fold from the 10-fold cross validation division of the tiger corpus. Here we do not take into account the words

Table 1: Mean Reciprokal Rank on the prediction of the PoS on 10% of the Tiger corpus, using 90% as training data. The prediction is only based on the morphological analysis, not taking into account information from surrounding words.

	all words	unknown words
HanTa	0.955	0.900
HanTa (hybrid)	0.962	0.900

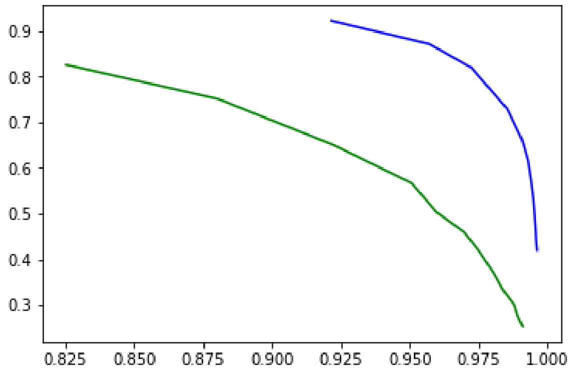

Figure 2: Precision and recall of the prediction of the PoS on 10% of the Tiger corpus. The green line gives the results for unknown words, while the blue (upper) line corresponds to all words.

around each word and for an ambiguous word we thus always predict the most likely tag.

Figure 2 shows the precision-recall curves for the tagger using only predicted probabilities. The MRR for known and unknown words for both variants of the tagger are given in Table 1.

Despite the low correlation of the observed and computed probabilities, the ranking of the results seems to be almost identical.

Next, we test the accuracy of lemmatization of HanTa, the TreeTagger, Spacy and GermaLemma on the Tiger and the TüBa Corpora. We use GermaLemma here in combination with our own Trigram Tagger implementation (GerTriTa). For GerTriTa and HanTa we use 10-fold cross validation on the Tiger Corpus. However, GermaLemma is trained on the Tiger Corpus. Thus, here we cannot really use the results. The same holds for Spacy that is trained on Tiger as well.

Since the correct lemmatization for many closed class elements is unclear and arbitrary (e.g. in Tiger the lemma of the determiner *das* is *der* while the TreeTagger generates the lemma *das*, which we do not want to consider as incorrect) we evaluate on the open class words only. Results are given in Table 2.

Table 2: Accuracy of lemmatization on the Tiger corpus. Values in brackets are obtained by evaluating on the training data

	all	unknown
HanTa	**96.98 ±0.24**	86.00 ±0.68
HanTa (hybrid)	**97.12 ±0.25**	86.00 ±0.68
TreeTagger	96.12	
Spacy	(87.46)	
GermaLemma	(97.79 ±0.20)	(96,38 ±0.37)

Table 3: Accuracy of lemmatization on the TüBa D/Z corpus.

HanTa	92.98
HanTa (hybrid)	93.06
TreeTagger	**93.59**
Spacy	86.60
GermaLemma	92.23

We also evaluated the lemmatization on the TüBa D/Z Corpus. This corpus was not used in development or training of any of the compared tools (as far as we know) and therefore is much better suited for evaluation. The results are given in Table 3.

4.2.2 Error Analysis

For the HanTa lemmatizer we clearly see two main sources of errors. In the first place many plural forms of long (unknown) nouns are not correctly analyzed as a stem and a plural suffix. E.g. the word *Plattenläden* (Record shops), occurring in TüBa D/Z but not in Tiger gets the lemma *Plattenläden*, since the analysis as one long unknown word is slightly more probable than the analysis as a compound, which would have enabled HanTa to correctly lemmatize the word as *Plattenladen*. Also for a simple word like *Volkslieder* (Folk songs) HanTa preferred the analysis as one large unknown noun over the analysis *Volk+s+lied+er*. This is partly also caused by the quite low probability of the suffix *er*. Here it could help to have more fine grained classes that would give a higher probability for the suffix *er* after certain nouns.

The second source of errors is formed by adjectival nouns and especially present participles that are used as nouns, like *Lehrende* (*teaching person*). We did not code this type of nouns in any special

Table 4: Accuracy of PoS tagging on the Tiger corpus.

	all	unknown
HanTa	96.52 ±0.33	**88.98 ±1.04**
HanTa (hybrid)	**96.96 ±0.34**	**88.98 ±1.04**
GerTriTa	**96.94 ±0.31**	88.04 ±0.72
TreeTagger	95.08	

way in the training data, but just coded them as one nominal morpheme. In the Tiger corpus lemmata are not assigned uniformly to these type of nouns. E.g. the word *Andersdenkende* (dissenting person) is lemmatized as *andersdenkend* , the word *Asylsuchenden* (asylum seeking person) is lemmatized as *Asylsuchender* (with strong masculine flexion) and *Wohlhabenden* (wealthy person) as *wohlhabende* (with weak flexion). In the TüBa D/Z corpus these type of nouns have three lemmata (one for each gender), separated by a '|'-sign in case the gender is underspecified and the lemma with the corresponding gender marking, if the gender is clear. Thus *Süchtigem* (addicted person, dative masculine singular) gets the lemma *Süchtiger*.

Most other lemmatizing errors are caused by ambiguity and the assignment of the wrong PoS. E.g. the word *überzeugt* (convinced) has to be lemmatized as *überzeugen* if it is a past participle, but it has to be lemmatized as *überzeugt* if it is a past participle used in an adjectival way (at least according to the annotation principles of Tiger and TüBa D/Z; see e.g. (Lenz, 1993) and (Eisenberg, 1994, p. 71) for a discussion on the status of German participles). Finally, the frequent words *möchte* and *möchten* (would like) are lemmatized incorrectly in Tiger as *möchten*, and thus learned incorrectly by HanTa, while they are correctly lemmatized as *mögen* in TüBa D/Z.

4.3 Part of Speech Tagging

For the evaluation of the PoS tagging based on the tag probabilities found by the lemmatizer we use two corpora: the TüBa D/Z treebank and the manually corrected part (part A) of the Hamburg Dependency Treebank. Especially, the latter one is interesting since its text are not from daily newspapers like the data from Tiger and TüBa D/Z.

The results for evaluating PoS tagging with 10-fold cross validation on the Tiger Corpus are given in Table 4. The results on TüBa D/Z and HTB are given in Table 5.

Table 5: Accuracy of PoS tagging on the TüBa D/Z and HDT Corpora.

	Tüba	HTB
HanTa	95.07	93.80
HanTa (hybrid)	**95.54**	**94.29**
GerTriTa	93.19	92.97
TreeTagger	94.81	92.87
Spacy	93.38	92.75

Table 6: Top 10 most frequent errors. The first column gives the correct PoS tag, the second column the predicted PoS and the last column the proportion this error has to the total number of errors.

PoS	Predicted PoS	Perc.
NN	NE	10.20 %
NE	NN	5.36 %
KOKOM	APPR	5.11 %
VVFIN	VVINF	4.23 %
NE	FM	3.09 %
ADV	ADJD	2.76 %
NN	ADJA	2.47 %
FM	NE	1.78 %
VVFIN	VVPP	1.69 %
KOUS	PWAV	2.63 %

4.3.1 Error Analysis

Finally, we have a more detailed look of the errors that HanTa makes on the TüBa-D/Z corpus. Table 6 shows the 10 most frequent errors. We see that there are some frequent ambiguous words, like *als* (as, than) and *wie* (as) that are hard to classify and already were reported by Giesbrecht and Evert (2009) to be a main source of errors. For most verbs the infinitive and first and third person present tense plural are identical and in many cases the correct class cannot be determined without syntactic analysis. Furthermore, there are many problems with proper nouns (NE). Here HanTa has difficulties to decide whether an unknown word is a proper noun (NE), a foreign word (FM) or a common noun (NN). In addition, Tiger and TüBa-D/Z also differ in the distinction between common nouns and proper nouns. E.g. the words *Osteuropa* (eastern Europe), *Bundesnachrichtendienst* (Federal intelligence office) and *EU-Kommission* (EU commission) are classified as proper names in Tiger but as common names in TüBa-D/Z.

Table 7: Average runtime of analyzing the first 1000 sentences from TüBa D/Z. All results are averages from 7 runs.

Tagger	Time
TreeTagger	2.85 s ± 0.501 s
Spacy	14.5 s ± 1.61 s
HanTa	10.7 s ± 0.102 s
HanTa (hybrid)	6.94 s ± 0.162 s
HanTa incl. lemm.	37.6 s ± 0.794 s
HanTa (hybrid) incl. lemm.	31.9 s ± 1.26 s

4.4 Run Time

We measured the time needed to tag and/or lemmatize the first 1000 sentences from TüBa D/Z in a Jupyter Notebook on a Laptop with one Intel i7 2.7 GHz Processor and 8.0 GB RAM. The results are given in Table 7. Currently the results of the morphological analysis of each word is not stored. So after PoS Tagging of the whole sentence the words have to be analyzed again for lemmatization. Moreover only probabilities for each PoS and not the lemmata are stored in the model, so for lemmatization each word has to be analyzed, which is clearly reflected in the run time. We report results for tagging only (i.e. analyzing each word only once) and for tagging and lemmatization.

5 Discussion

Looking at the lemmatization, we see that our approach gives surprisingly good results: the approach in fact is quite naive, the morphological classes are too coarse-grained and the model is massively overgenerating and allowing for all kind of nonsense analyses. Nevertheless, in most cases the correct PoS and the correct lemma is predicted. On the Tiger corpus HanTa is even slightly better than the TreeTagger, on the TüBa D/Z Treebank the TreeTagger outperforms HanTa with half a percent. GermaLemma gives the best results on Tiger. However, GermaLemma uses a dictionary derived from the Tiger corpus, thus a comparison on these data is not fair. On TüBA D/Z GermaLemma does not perform very well, but this is due to the bad performance of the trigram PoS tagger that was used to provide GermaLemma with the PoS tags it needs. The results from Spacy in both experiments are much behind all other approaches.

HanTa's accuracy on lemmatization (97.12 %) at first glance seems to be below the results of LEMMING reported by Müller et al. (2015) (98.10 %). However, these results cannot be compared directly. In the first place, the reported result is on one split from the Tiger corpus, but it is unclear, whether it is a contiguous or a random split. More important, we excluded all closed class words from the evaluation. Since most closed class words occur very frequently and are easy to lemmatize, including these words will improve the results.

In the evaluation of the PoS Tagger the first remarkable observation is the result from the TreeTagger that is noticeable below the evaluation results of Giesbrecht and Evert (2009). A possible source of difference could be the version of the Tiger corpus. Probably, Giesbrecht and Evert used version 1 of the Tiger corpus that consists of $0.7 \cdot 10^6$ tokens.

Here again the results from Spacy stay behind the other taggers. Interestingly, the baseline trigram tagger is almost as good as HanTa on the Tiger corpus, but on the Hamburg Dependency Treebank HanTa outperforms the baseline clearly. Thus, indeed, the careful splitting of a word into its stem and suffix has an advantage over just using the last letters of a word to guess its PoS.

6 Conclusion and future work

In this paper we have presented a simple approach to German lemmatization. We have evaluated the lemmatization on three different large corpora and shown that the results are close to results that can be obtained by state of the art tools and methods. Furthermore, we have shown, that the use of HanTa's morphological analysis for unknown words in PoS tagging is more useful than using arbitrary length suffixes to guess the PoS. The PoS tagging using morphological analysis even outperforms other widely used PoS taggers.

In order to make HanTa a useful tool, we will work on the speed of the analysis, which is now clearly below that of most other tools evaluated here. Small improvements on the quality can be achieved by further development of the script generating the training data. Here e.g. a better treatment of adjectival nouns could help. Most interestingly, however, would be to see the effect of using more fine grained morpheme classes, including information on number, gender, tense, etc.

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories.*, volume 168.

Thorsten Brants. 2000. Tnt: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, pages 224–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martin Braschler and Bärbel Ripplinger. 2004. How effective is stemming and decompounding for german text retrieval? *Information Retrieval*, 7(3):291–316, Sep.

Grzegorz Chrupala, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with Morfette. In *LREC 2008*.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3.

Peter Eisenberg. 1994. *Grundriß der deutschen Grammatik. 3., überarbeitete Auflage*. Metzler, Stuttgart/Weimar.

ExplosionAI GmbH. 2019. German: Available pretrained statistical models for german. `https://spacy.io/models/de`. Accessed: 2019-06-24.

Kilian Foth, Arne Köhn, Niels Beuck, and Wolfgang Menzel. 2014. Because size does matter: The hamburg dependency treebank. In *Proceedings of the Language Resources and Evaluation Conference 2014 / European Language Resources Association (ELRA)*. Universität Hamburg.

Alexander Gashkov and Mariia Eltsova. 2018. Lemmatization with reversed dictionary and fuzzy sets. In *SHS Web of Conferences*, volume 55, page 04007. EDP Sciences.

Andrea Gesmundo and Tanja Samardžić. 2012. Lemmatisation as a tagging task. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 368–372, Jeju Island, Korea, July. Association for Computational Linguistics.

Eugenie Giesbrecht and Stefan Evert. 2009. Is part-of-speech tagging a solved task? an evaluation of pos taggers for the german web as corpus. In *Proceedings of the fifth Web as Corpus workshop*, pages 27–35.

John A Goldsmith, Jackson L Lee, and Aris Xanthos. 2017. Computational learning of morphology. *Annual Review of Linguistics*, 3:85–106.

Bart Jongejan and Hercules Dalianis. 2009. Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 145–153, Suntec, Singapore, August. Association for Computational Linguistics.

Jakub Kanis and Luděk Müller. 2005. Automatic lemmatizer construction with focus on oov words lemmatization. In *International Conference on Text, Speech and Dialogue*, pages 132–139. Springer.

Kimmo Kettunen, Tuomas Kunttu, and Kalervo Järvelin. 2005. To stem or lemmatize a highly inflectional language in a probabilistic ir environment? *Journal of Documentation*, 61(4):476–496.

Markus Konrad. 2017. Lemmatization of german language text. `https://datascience.blog.wzb.eu/2017/05/19/lemmatization-of-german-language-text/`. Accessed: 2019-06-24.

Kimmo Koskenniemi. 1983. *Two-level morphology: A general computational model for word-form recognition and production*, volume 11. University of Helsinki, Department of General Linguistics Helsinki.

Barbara Lenz. 1993. Probleme der kategorisierung deutscher partizipien. *Zeitschrift für Sprachwissenschaft*, 12(1):39–76.

Cristian Moral, Angélica de Antonio, Ricardo Imbert, and Jaime Ramírez. 2014. A survey of stemming algorithms in information retrieval. *Information Research: An International Electronic Journal*, 19(1):n1.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, Lisbon, Portugal, September. Association for Computational Linguistics.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May. European Language Resources Association (ELRA).

H. Schmid, 1999. *Improvements in Part-of-Speech Tagging with an Application to German*, pages 13–25. Springer Netherlands, Dordrecht.

Heike Stadler and Werner Wegstein. 2016. Encoding a derewo word-list. In *Varianz und Vielfalt interdisziplinär: Wörter und Strukturen*, pages 65–73. Institut für deutsche Sprache.

Heike Telljohann, Erhard Hinrichs, and Sandra Kübler. 2004. The tüba-d/z treebank: Annotating German with a context-free backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May. European Language Resources Association (ELRA).

The CLiPS (Computational Linguistics & Psycholinguistics) research center. 2018. pattern.de. `https://www.clips.uantwerpen.be/pages/pattern-de`. Accessed: 2019-06-23.