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

Evaluating Off-the-Shelf NLP Tools for German

Katrin Ortmann **Adam Roussel** **Stefanie Dipper**
Department of Linguistics
Fakultät für Philologie
Ruhr-Universität Bochum
ortmann|roussel|dipper@linguistics.rub.de

Abstract

It is not always easy to keep track of what tools are currently available for a particular annotation task, nor is it obvious how the provided models will perform on a given data set. In this contribution, we provide an overview of the tools available for the automatic annotation of German-language text. We evaluate fifteen free and open source NLP tools for the linguistic annotation of German, looking at the fundamental NLP tasks of sentence segmentation, tokenization, POS tagging, morphological analysis, lemmatization, and dependency parsing. To get an idea of how the systems' performance will generalize to various domains, we compiled our test corpus from various non-standard domains. All of the systems in our study are evaluated not only with respect to accuracy, but also the computational resources required.

1 Introduction

An extensive number of NLP tools are available nowadays for the automatic analysis of natural language data. The vast majority of these tools have been developed for English, though, and often take advantage of specific properties of the English language, such as the fact that English sentences show a rather fixed word order (so Markov models work well), or that English does not have a rich inflectional morphology (so lemmatizers are not a major concern). As a result, it is often not clear to what extent these tools are applicable to further languages, and often no pre-trained models are provided for languages other than English.

Similarly, these tools are mostly evaluated only on English language data, but the results for English may not be transferable to other languages. Depending on the language, different kinds of annotations can be necessary, which are not relevant for English.

Moreover, most tools are trained and tested on standard (newspaper) language, but different registers of a language can differ significantly, e.g. with respect to syntax or lexicon (Biber and Conrad, 2009). Performance of tools trained on standard language drops considerably when these tools are applied to data from other registers, such as social media data.

And finally, the efficiency of many freely available systems, i.e. the time and computing resources they require for the annotation task, can become a problem in the context of practical applications or if large amounts of text need to be analyzed.

The goals of this work are, first, to determine what freely-available systems exist for the linguistic analysis of German texts. Second, we want to assess the accuracy of their output in various non-standard domains, including formal (Wikipedia), semi-formal (sermons), and informal (movie subtitles) contexts. We will evaluate the systems in the fundamental NLP tasks of sentence segmentation, tokenization, part-of-speech (POS) tagging, morphological analysis, lemmatization, and dependency parsing, and we will provide an overview of the computational resources each system requires for these tasks.

The remainder of this paper is structured as follows: Section 2 gives an overview of related work. Section 3 introduces the data used in the study and the gold standard annotation. The NLP tools tested in this study are introduced in Section 4. In Section 5, we describe our experimental setup for the evaluation of the selected systems with respect to both accuracy and speed. We conclude with a discussion of the results in Section 6 and 7.

2 Related Work

The Association for Computational Linguistics (ACL) provides state-of-the-art results for a range of NLP tasks, such as POS tagging, named entity recognition, parsing, paraphrase identification, or question answering.[1] However, the majority of results reported here are based on English data only, and even for common tasks like POS tagging, only results for English and French are available. This, of course, does not mean that POS tagging has not been evaluated for other languages as well. It shows, though, that results for other languages are scattered across numerous publications and not readily available. Moreover, even results of the same target language are not easily comparable to each other because evaluations often use different data sets.

While most evaluations are confined to newspaper texts, there are some exceptions. Gadde et al. (2011) evaluate and adapt a POS tagger trained on standard English data (the WSJ corpus) to SMS data. Gimpel et al. (2011) and Owoputi et al. (2012) develop POS taggers for Twitter data, with accuracies of around 90% (best version: 92.80%). Choi et al. (2015) evaluate statistical dependency parsers on the English part of OntoNotes 5, with data from different genres, which they specify as broadcasting conversation, broadcasting news, news magazine, newswire, pivot text, telephone conversation, and web text. In their study, the best overall accuracies are reached by the ClearNLP and the Mate parser with about 90%, while they found the spaCy parser (version 1.x) to be the fastest parser annotating 755 sentences or 13,963 tokens per second.

For German, Giesbrecht and Evert (2009) evaluate POS taggers on web data and show that accuracy values drop significantly for non-standard data, especially for posts from online forums, which were tagged with accuracies $< 90\%$. Similarly, Neunerdt et al. (2013) evaluate and compare taggers trained on standard data with taggers retrained on comments in German from different online forums. Standard taggers achieve accuracies of $> 87\%$, retrained taggers of $> 93\%$. Beißwenger et al. (2016) show that these results hold true several years later, as they find similar results for the annotation of computer-mediated communication (CMC) and web data. While the best systems reach F_1-Scores $> 99\%$ for tokenization, the best tagger only achieved a tagging accuracy of 90%.

[1]`https://aclweb.org/aclwiki/State_of_the_art`

3 Data

As already mentioned, language varieties differ considerably on all linguistic levels, e.g. syntax or lexicon (Biber and Conrad, 2009), and pose different challenges to automatic annotation tools. Since most available models are trained on standard (newspaper) language, their accuracy might drop when they are applied to non-standard data like informal written communication. By using data from various domains, ranging from formal to informal contexts, we can evaluate whether the application of pre-trained models is limited to a restricted language domain or if they can be used for other language varieties as well.

For the evaluation, we used data from five different registers representing large resources of raw texts: encyclopedic text (*Wikipedia*)[2], literary text (*Novelette*)[3], Christian sermons (*Sermon*)[4], prepared but freely-performed talks (*TED*)[5] and movie subtitles (*Movie*)[6]. Table 1 gives an impression of the selected text types and their particularities.

For each register, a random sample of approximately 1,500 tokens was selected, resulting in a total of 559 sentences with 7,642 tokens (2,649 types). Table 2 gives an overview of the data.

Gold Standard For all annotations we manually created a gold standard. The annotation of sentence boundaries[7] and tokenization was carried out in text files, and all other annotations were done with WebAnno (de Castilho et al., 2016)[8].

We use the Stuttgart–Tübingen tagset (or STTS, Schiller et al. (1999))[9] for POS tagging, and the

[2]Sample `wpd15_sample.i5.xml` from the Wikipedia subcorpus of DeReKo (`http://corpora.ids-mannheim.de/pub/wikipedia-deutsch/2015/wpd15_sample.i5.xml.bz2`).

[3]Texts of the genre 'novelette' from GutenbergDE corpus, edition 14 (`http://gutenberg.spiegel.de/`), which were published after 1900.

[4]Automatically downloaded from the SermonOnline database (`http://www.sermon-online.de`)

[5]German translations of English talks, automatically downloaded from the official website `https://www.ted.com/talks?language=de`.

[6]German subtitles for movies tagged as "Action, Adventure, Drama" or "Comedy, Drama" from the OpenSubtitles corpus (`http://www.opensubtitles.org/`), downloaded at `http://opus.nlpl.eu/download.php?f=OpenSubtitles/v2018/raw/de.zip`

[7]For Wikipedia and Movie subtitles, sentence boundaries were already present in the data and only corrected as necessary.

[8]Version 3.4.6 (`https://webanno.github.io/webanno/`

[9]STTS is the de facto standard POS tagset for German

Subcorpus	Example Sentence
Wikipedia	Das 6. Kanadische Kabinett (engl. *6th Canadian Ministry*, franz. *6e conseil des ministres du Canada*) regierte Kanada vom 21. Dezember 1894 bis zum 27. April 1896.
Novelette	Zwischen dem roten Rausch der Pelargonien und seinem Damaskus lag eine Fülle engbeschriebener Tagebuchblätter, jedes mit der zierlich-säuberlichen Unterschrift »Erich Friese, Kandidat« versehen.
Sermon	Wenn ihr aber zu Christus gehört, seid ihr auch Abrahams Nachkommen und bekommt das Erbe, das Gott Abraham versprochen hat. (Gal 3,29)
TED	Im Himalaya, der drittgrössten Eismasse, sehen Sie oben neue Seen, die vor ein paar Jahren Gletscher waren.
Movie	- Dad, das reicht!

Table 1: Example sentences from each register.

Subcorpus	Register	#Tok	#Sent	#Doc
Wikipedia	written, encyclopedic	1,514	96	12
Novelette	written, prose	1,588	69	12
Sermon	spoken, planned	1,520	90	16
TED	spoken, planned	1,506	101	17
Movie	spoken	1,514	203	21
Total		7,642	559	78

Table 2: Overview of the data from the five subcorpora used in the study.

TIGER annotation scheme (Crysmann et al., 2005) for morphological annotations, which can be considered an extension and improvement upon the original 'large' STTS. We also performed lemmatization on the texts, also according to the TIGER annotation scheme, albeit with some modifications: nominalizations are treated like normal nouns, and personal and reflexive pronouns are annotated with their nominative form (e.g. *ich*, *du*, etc.) except where this is not possible, as is the case for *sich*. For tokens that are not annotated with a lemma in the TIGER scheme, such as interjections, foreign words, punctuation, we use the surface form as its lemma.

For dependencies, we annotated only arguments, which should be reliably comparable across annotation schemes, and not modifiers, which are handled quite differently in different schemes, e.g. with regard to attachment site or label name.[10] For this paper, we consider subjects, direct and indirect objects, clausal subjects and objects, predicatives, and expletives to be arguments. We excluded prepositional objects because it is often difficult to determine when they are acting as arguments or adjuncts. All texts were annotated independently by one to five student annotators and checked manually afterwards by one of the authors.

Annotation	#Tokens	#Types	#Ambig
POS	7,642	53	114
Lemma	7,642	2,077	27
Morphology	4,331	233	323

Table 3: Overview of the gold standard dataset.

Table 3 provides an overview of the gold data. The table shows the number of tokens with a particular type of annotation (which varies since words that cannot be inflected are not annotated for morphology), the number of annotated types, and the number of ambiguous word forms (e.g. the word form *gehört*, could either represent a present tense form of the lemma *gehören* 'belong' or a participle form of the lemma *hören* 'hear'). Figure 1 shows the distribution of dependency relations in the gold data.

4 NLP Tools

For the evaluation in this paper, we selected systems that are freely available, i.e. open source, and for which pre-trained models for German are provided. Of course, the selection of tools we test here is not comprehensive, but we will make our data and evaluation scripts publicly available so further systems can be added and compared with

data. Morphological tagsets and lemmatization is less well standardized for German but most corpora use schemes that are based on or similar to the TIGER schemes we use.

[10]In order to be able to compare diverging annotation formats, we also annotated auxiliaries and prepositions within predicatives (cf. Section 5).

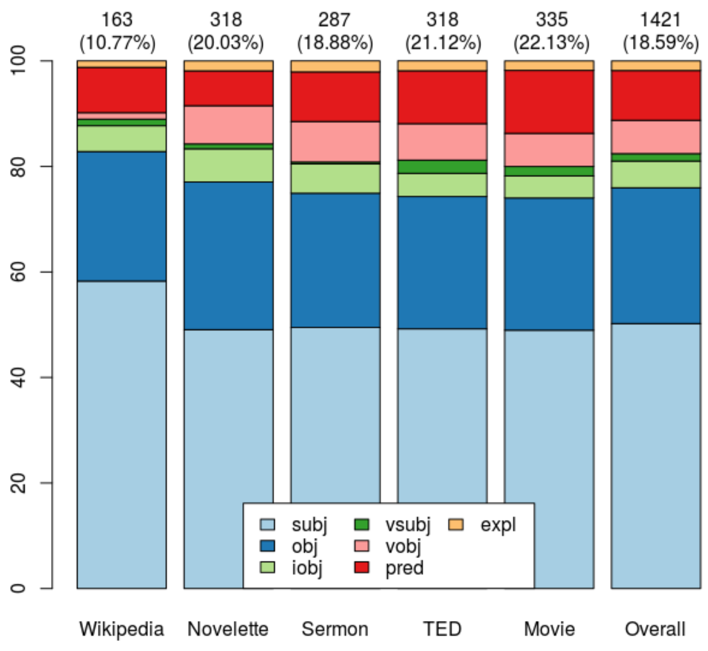

Figure 1: Distribution of dependency relations in the gold data. The numbers on top of the columns indicate the total frequencies of the seven relations considered in the evaluation and, in parentheses, the corresponding percentage (e.g. in the Wikipedia subcorpus, 163 tokens, which correspond to 10.77% of all tokens, have been annotated with one of the seven relations).

the systems described here.

Most prominently, there are those systems which provide a full range of annotations from tokenization to dependency parsing: CoreNLP (Manning et al., 2014), which combines rule-based tokenization with statistical tagging and dependency parsing, spaCy v2.1.3 (Honnibal and Montani, 2017),[11] and the Python-based StanfordNLP (Qi et al., 2018).

We also test a few dedicated tokenizers: NLTK (Bird et al., 2009), comprising a regular expression–based word tokenizer and the Punkt tokenizer (Kiss and Strunk, 2006) for sentence segmentation, SoMaJo (Proisl and Uhrig, 2016), and Syntok (Leitner, 2019). We evaluate both the rule-based sentencizer included in the spaCy pipeline as well as the sentence boundaries derived from spaCy's dependency analysis, which is the usual way of determining sentence boundaries with spaCy. These results are listed under 'spaCy parser' in Table 4.

Some tools only perform lemmatization, such as IWNLP (Liebeck and Conrad, 2015), which makes use of a word form to lemma mapping extracted from the German Wiktionary, and GermaLemma (Konrad, 2019) (with the Pattern extensions[12] enabled), based on the TIGER corpus.

Since GermaLemma only lemmatizes words from a restricted set of POS (N*, V*, ADJ* and ADV*), i.e. content words, we combined this system with another tool that also lemmatizes function words (spaCy) and added a few special rules for pronouns. This ensemble lemmatizer is included in the study as GermaLemma++.[13]

Many of the tools we test provide some combination of word-level annotations. SoMeWeTa (Proisl, 2018) produces STTS tags (small tagset) only, whereas RFTagger (Schmid and Laws, 2008)[14] and Clevertagger (Sennrich et al., 2013) use a modified version of the large STTS that includes morphological features. TreeTagger (Schmid, 1994; Schmid, 1995) provides just STTS and lemmas.

We also test ParZu (Sennrich et al., 2009), which is a parser that combines a hand-written grammar with statistical approaches.

Of the selected tools, RNNTagger (Schmid, 2019) and StanfordNLP (Qi et al., 2018) are both neural network–based systems implemented in the PyTorch framework. SpaCy uses convolutional

[11]`https://github.com/explosion/spaCy/releases/tag/v2.1.3`

[12]Smedt and Daelemans (2012), `https://www.clips.uantwerpen.be/pages/pattern`.

[13]`https://github.com/rubcompling/germalemmaplusplus`

[14]Specifically the Java interface RFTJ (Ziai and Ott, 2014), which offers easier production of STTS tags, improved lemmatization, as well as finer control over model loading.

neural network models for tagging and parsing, but does not need a GPU or significant resources for annotation or training.

5 Evaluation

In this study we evaluate the NLP tools with respect to both accuracy and speed. Section 5.1 describes the performance evaluation, i.e. annotation speed of the selected systems. In Section 5.2, the comparison of the different annotations, as produced by the systems, with the gold standard annotation is explained.

5.1 Computational Efficiency

We are interested not only in the annotations that the systems we test produce, but also in the computational resources they require to produce them. This is particularly a concern in the context of practical applications where timely results are critical.

We wanted to measure the resources each system requires to produce each level of annotations, but the production of each type of annotation is not always separable from the others, e.g. morphology and/or lemma annotations are often produced together with POS tags (and though there are sometimes options regarding whether or not they are output, this doesn't necessarily mean that they aren't computed). Also, sentence segmentation and word tokenization are sometimes dependent on one another, but in different directions. CoreNLP requires sentence boundaries before tokenization, and others, like Syntok, do this the other way round.

For each annotation step, we separated model loading from actual annotation time required. We measure the time the various systems require to complete three roughly-comparable annotation steps: (1) tokenization (sentence segmentation, word tokenization), (2) word-level annotations (POS, morphology, lemmas), and (3) dependencies. Systems that only perform lemmatization are run separately, since they also require POS annotations as an input, all other systems produce lemmas and POS annotation simultaneously.

The annotation time for each step was measured as CPU time across five trials for each of the subcorpora, using a measure of seconds per thousand tokens, which represents the computational resources required by a particular system for a particular task. All trials were performed on a Linux workstation equipped with an Intel Core i7-5820K processor, 15 GB of RAM, and an Nvidia GeForce GTX 980 graphics card with 4 GB of memory.

5.2 Accuracy

For the accuracy evaluation, the systems were provided with gold annotations from the previous annotation steps, as detailed here:

	Input (Gold)		**Output**
Tokenization	Plain text	→	Sentences, Tokens
Word-level	Sentences, Tokens	→	POS, Morph, Lemmas
Lemmatization	POS	→	Lemmas
Dependencies	POS	→	Dependencies

Of course this approach is not a realistic scenario, as manually created, i.e. completely correct, annotations are normally not available, so these results are to be interpreted as an upper bound.

Tokenization In the Universal Dependencies Treebank for German,[15] on which the StanfordNLP model was trained, multi-word tokens (APPRART in the STTS) are split into separate words. To enable the comparison of the system's output with the gold standard, we suppressed these splits during tokenization.

POS The deviating tokenization of multi-word tokens also affects the other annotation levels, as the corresponding models expect the tokens to be split for them to be annotated correctly. We therefore accepted split multi-word tokens for the concerned systems (StanfordNLP) for all word-level annotations and used rules to check if the system's annotation (APPR + ART) matches the gold-standard annotation (APPRART).

Morphology The morphology annotations of the systems follow different naming conventions for the morphological features (e.g. `Sg` vs. `Sing` for 'singular'), so, in order to compare them, we mapped all annotations to the TIGER tagset (Crysmann et al., 2005). Some systems also annotate further morphological features, which we ignored either because they are not present in the TIGER tagset (features like definiteness) or because they are already included in the POS tags (e.g. finiteness).

[15] UD German GSD (`https://universaldependencies.org/treebanks/de_gsd/index.html`)

Morphology tags are composed of multiple features, e.g. `3.Pl.Pres.Ind`, which we evaluate individually. Features that are only present in the system output are ignored. The overall accuracy is calculated token-wise as the average percentage of morphological features in the gold standard that were present and correct in the system output.

Lemmas The NLP tools included in this study follow different guidelines for the annotation of lemmas, so we allow for a degree of variation in this regard, so long as it is clear that the analysis a given lemma variant suggests is correct. For instance, most prominently, some use the masculine form of the definite article as its lemma, *dem* $\rightarrow$ *der*, whereas others might use the feminine form *die*. For words tagged as PPER, the surface form is accepted as its lemma. Reflexive pronouns may be lemmatized as *sich*, and words tagged as ART, PDS or PRELS may be lemmatized as *eine* as well as *ein* or *die* as well as *der*. We also consider <ß> and <ss> to be equivalent throughout.

Furthermore, we accept some special lemmas used by certain systems. For example, since RFTagger lemmatizes all numbers to a common symbol, either `<card>` or `<ord>`, we consider these symbols equivalent to the token's surface form, since this is how we lemmatized numbers in our gold-standard dataset. Similarly, TreeTagger lemmatizes multi-word tokens tagged as APPRART with the lemmas of the constituent prepositions and articles, as in `zu+die` for `zur` instead of `zu`, as in our dataset. We also accept these forms as correct.

In general, all comparisons of lemmas are case-insensitive and wherever a tool does not produce a lemma for a given token, we take the surface form of that token as its lemma. If a tool outputs more than one alternative lemma for a word, as is the case for StanfordNLP, IWNLP, RFTagger and TreeTagger, we evaluate the first one.

Dependencies For the dependency evaluation, we only consider the relevant argument relations (subj, obj, iobj, vsubj, vobj, pred, expl). The four parsers compared in the study follow three different guidelines for dependency annotation and each of them uses a different scheme. In order to account for differing naming conventions, we accept a number of alternative labels for each relation, which we consider to be equivalent. To capture the structural differences between guidelines we use a set of rules to allow for certain mismatches between the system outputs and the gold standard. In particular, the head of a relation may be the main verb, as in the gold standard, or an accompanying finite auxiliary as annotated by the parsers. Similarly, the head of a predicative phrase may be the noun, as is the case in the gold standard, or the preposition (for ParZu and spaCy). We also cover the copula analysis of CoreNLP and StanfordNLP, which according to the Universal Dependencies guidelines,[16] analyse the predicative phrase as the head of a copula construction. Finally, there are certain constructions in German for which the subject is difficult to determine (also theoretically). These constructions involve certain predicatives and expletives (e.g. as in *Das ist es* 'that's it'). To address this, we consider it correct if the system switches the subject and the predicative phrase, and similarly, if the system analyses expletives as subjects (or objects, in rare cases) and vice versa. The StanfordNLP parser performs much better when Universal POS tags are present, so we add these to the data provided to this parser using the mapping provided by the Universal Dependencies project.[17]

6 Results and Discussion

In this section, we examine the most important results of our experiments. All of the scripts we used and detailed results (including per-register results) can be found in this paper's repository at `https://github.com/rubcompling/konvens2019`.

6.1 Computational Efficiency

Figure 2 shows the results of run time evaluation. Overall, the annotation speed of most systems does not differ substantially. The word-level StanfordNLP component, encompassing POS tagging, morphological analysis, and lemmatization, is an outlier, with much longer run times, on average, than the other systems. The RNNTagger, another neural network–based system, is also on the upper end of the scale among the systems we tested. This is indicative of the large computational resources such systems require.

Though the CoreNLP dependency parser is the slowest of those we tested, CoreNLP is the fastest system for word-level annotations (which is probably related to the fact that it only produces POS

[16]`https://universaldependencies.org/guidelines.html`

[17]`https://universaldependencies.org/tagset-conversion/de-stts-uposf.html`

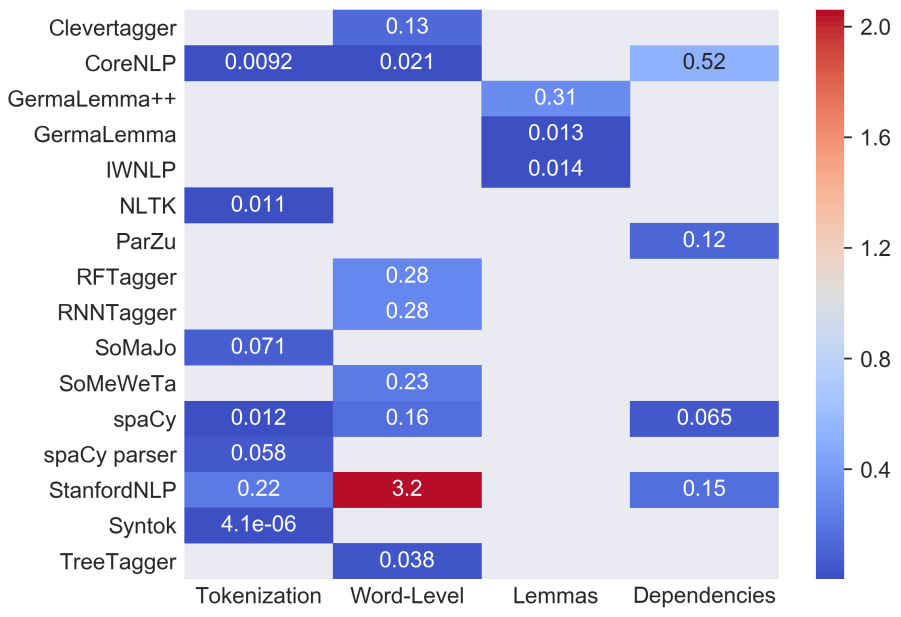

Figure 2: Process time required by the systems according to task, measured in seconds per thousand tokens.

annotations in this step), though this appears to come at the cost of accuracy. As was the case in Choi et al. (2015), spaCy is capable of the fastest dependency parsing.

The fastest system for tokenization is Syntok (another outlier) requiring only 0.000004 seconds per 1,000 tokens. It doesn't seem to use any linguistic models as such – just a few well-selected regex-based heuristics. Despite the minimal computational resources required, Syntok is still the second most accurate sentence segmenter we tested.

6.2 Accuracy

Table 4 shows the results of the accuracy evaluation for all systems and annotations.[18]

Tokenization For sentence segmentation and tokenization, we calculated F_1-scores for all systems. The results show that sentence segmentation is more challenging than word tokenization which can be considered a solved task.

A closer look at the system output shows that many of the systems have difficulty recognizing abbreviations, ordinal numbers (e.g. in dates), ellipsis dots, and dashes that are used to indicate speaker changes in movie dialogues. Further problems are caused by direct speech, especially as regards the use of inward-pointing angle quotes (as in »example text«) which are typical for German literary texts but appear to be unexpected for some of the systems. Also the data contains some sentences which are not marked with punctuation marks and can only be recognized as sentences based on the content.

POS The results for POS tagging are similar to the findings of Giesbrecht and Evert (2009) and Beißwenger et al. (2016) with tagging accuracies ranging from 88.2% to 94.3%. However, there are only slight differences between the selected registers. Most taggers perform best on the TED talk transcripts while the lowest accuracy values can be observed for movie subtitles.

The most frequent errors for all taggers include the common confusions of nouns and proper names, adverbs and adverbial adjectives, and of different verb forms. It is striking that all of the tools also tag sentence internal punctuation incorrectly, e.g. as numbers, names, adjectives, etc., which is the most frequent error for half of the taggers reducing their accuracy by up to 1.6 percentage points.

Morphology The accuracy values for morphology annotation differ substantially between systems with results varying by 10 percentage points.

Depending on their POS, words have different morphological features. Following the TIGER

[18]As the formulation of sensible baselines for several of the annotation steps, e.g. morphology and dependencies, can be quite complex, we don't include them in our analysis here.

	Tokens	**Sents**	**POS**	**Morph**	**Lemmas**	**Deps**
Clevertagger	–	–	0.8818	0.8338	–	–
CoreNLP	**0.9965**	0.8909	0.9074	–	–	0.5316
GermaLemma++	–	–	–	–	0.9541	–
GermaLemma	–	–	–	–	0.8786	–
IWNLP	–	–	–	–	0.8650	–
NLTK	0.9961	**0.9177**	–	–	–	–
ParZu	–	–	–	0.8359	0.9431	**0.7744**
RFTagger	–	–	0.9310	0.8903	0.9377	–
RNNTagger	–	–	0.9346	**0.9248**	**0.9751**	–
SoMaJo	0.9964	0.8480	–	–	–	–
SoMeWeTa	–	–	0.9403	–	–	–
spaCy	0.9955	0.8410	0.9250	–	0.8913	0.6687
spaCy parser	–	0.8940	–	–	–	–
StanfordNLP	0.9920	0.8989	**0.9427**	0.8235	0.9415	0.7217
Syntok	0.9912	0.9125	–	–	–	–
TreeTagger	–	–	0.9210	–	0.9605	–

Table 4: Overall F_1-scores (for tokens and sentences) or accuracy (for POS, morphology, lemmas and dependencies) for all systems at all annotation levels.

scheme (Crysmann et al., 2005), we evaluate seven features: gender, case, person, number, tense, mood, and degree. Table 5 provides detailed results for all features.

As morphology annotation depends on correct POS tags, it is possible that errors on this annotation level are the result of error propagation from the POS level, e.g. adjectives that are tagged as adverbs do not receive a degree annotation, finite verbs that are recognized as infinitives are not annotated at all. Sometimes the systems also indicate that they cannot assign a unique value to an attribute although this is possible for human annotators based on the given context.

While no clear differences between registers can be observed, there are some morphological features that generally seem to be harder than others. Overall, the most difficult features are gender and case with error rates of 10% or more for all systems. The annotation of gender is most difficult for proper names and pronouns, while error rates are low for nouns and articles.

It should also be noted that StanfordNLP does not annotate the degree feature for adjectives, which occurs with 10% of the annotated tokens and makes up 3% of the annotated features overall. This explains, in part, the system's poor result in this annotation task.

Lemmas Most of the lemmatizers achieve accuracy values well above 90%. Always choosing a given word form as its lemma results in an accuracy of 70.7%. In general, there are no significant differences in lemmatization between registers. However, all of the systems do perform slightly worse on the Novelette subcorpus than on the other subcorpora.

We observe frequent deviations from the gold standard for demonstrative and indefinite pronouns, which can likely be attributed to further differences between lemmatization guidelines. Some systems also produce stems instead of lemmas for some words, e.g. StanfordNLP and TreeTagger annotate the stem *jen* instead of possible lemmas *jener* or *jene*. Furthermore, some systems do not lemmatize certain words or word classes at all, for instance content words in the case of GermaLemma, which results in a lower accuracy as well.

Dependencies The evaluation shows that dependency annotation is clearly the most difficult of the six annotation tasks. All four parsers we tested achieve accuracies for the annotation of arguments below 78%. A closer look at the results shows that the most difficult relations seem to be clausal subjects, clausal objects and expletives.

The spaCy parser produces the largest number of false positives, i.e. argument relations that are not present in the gold standard (mostly object clauses).

	Case	Degree	Gender	Mood	Number	Person	Tense
Clevertagger	0.8385	0.9217	0.6537	0.9009	0.9261	0.9108	0.8803
ParZu	0.8104	0.8733	0.7913	0.6744	0.8870	0.9388	0.9743
RFTagger	0.8858	0.9447	0.7734	0.9537	0.9522	0.9492	0.9550
RNNTagger	0.9094	0.9401	0.8770	0.9730	0.9547	0.9713	0.9717
StanfordNLP	0.8563	–	0.7634	0.9189	0.9527	0.8954	0.9408

Table 5: Overall accuracy per morphological feature.

This may in part result from differences in handling coordination: While spaCy tends to annotate the affected relation once per coordinated element, other guidelines use a special coordination relation for this.

CoreNLP has the second highest number of false positives. It also makes the most mistakes of the four parsers and has the highest error rates for almost all relations, resulting in an accuracy value only just above 50%.

StanfordNLP and ParZu both reach accuracies above 70%. ParZu achieves the best result of all parsers overall and produces much fewer false positives.

However, it often annotates multiple roots per sentence and/or annotates subjects, objects or clausal objects as root of the sentence. It also seems to have a slightly different definition of predicatives and annotates expletives worse than the other systems (although no system annotates expletives particularly well).

7 Conclusions and Future Work

Though we would like to directly relate accuracies to run times and provide some concrete measure of efficiency, this is difficult in practice, since run times pertain to multiple accuracy dimensions: the run time of the POS annotation step sometimes covers just POS, morphological annotations, and/or lemmatization, depending on what a given system provides, which makes it difficult to say what the relationship is between run time and accuracy exactly. Nevertheless, we will attempt to take the systems' run times into account when comparing the accuracies achieved.

Though the F_1-scores above 0.99 for all tested tokenizers suggest that tokenization can be considered a solved task, there is some degree of difference in how the tools handle sentence segmentation. Overall, NLTK provides some of the best token and sentence boundaries, while being one of the fastest systems we tested. Syntok is even faster and only slightly less accurate.

RNNTagger offers high accuracy across all word-level annotations and is relatively fast, especially as compared with taggers offering similar levels of accuracy. Though the StanfordNLP tagger seems to be a bit more accurate at POS tagging, it does so at the cost of requiring much greater computational resources. If you have POS annotations and just need lemmas, then GermaLemma++ provides the best accuracy with only a modest increase in the computing resources required, in comparison to standard GermaLemma.

SpaCy offers relatively good performance, but requires especially little computational resources and can annotate large volumes of data quickly. However, where accuracy is most important, ParZu might provide better results while still being relatively fast.

Future work could include retraining the systems for each domain and observing the degree to which these domain-specific models improve the accuracy of the systems' output. Furthermore, one could use these results to construct a pipeline from the appropriate systems for a particular situation and have an easy-to-use and effective NLP pipeline for the standard linguistic annotations. It would also be interesting to evaluate in future work how well such a pipeline (in which each stage depends on system output, as opposed to gold-standard annotations) would compare to the upper bound results described here.

Acknowledgments

We would like to thank our student annotators Doreen Scholz, Larissa Weber, Jennifer Wodrich, Helena Wedig, Sara Klein, Anna Ehlert and Katharina Streit for the annotations. This work was supported by the German Research Foundation (DFG), SFB/CRC 1102 "Information density and linguistic encoding" (Project C6).

References

Michael Beißwenger, Sabine Bartsch, Stefan Evert, and Kay-Michael Würzner. 2016. EmpiriST 2015: A shared task on the automatic linguistic annotation of computer-mediated communication and web corpora. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task*, pages 44–56.

Douglas Biber and Susan Conrad. 2009. *Register, Genre, and Style*. Cambridge University Press.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Jinho D. Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 387–396.

Berthold Crysmann, Silvia Hansen-Schirra, George Smith, and Dorothea Ziegler-Eisele. 2005. *TIGER Morphologie-Annotationsschema*. Retrieved from `https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/TIGERCorpus/annotation/tiger_scheme-morph.pdf`.

Richard Eckart de Castilho, Éva Mújdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. 2016. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH) at COLING 2016*, pages 76–84.

Phani Gadde, L.V. Subramaniam, and Tanveer 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*.

Eugenie Giesbrecht and Stefan Evert. 2009. Part-of-speech tagging – a solved task? An evaluation of POS taggers for the web as corpus. In *Proceedings of the 5th Web as Corpus Workshop (WAC5)*, pages 27–35.

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: short papers*, pages 42–47.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Markus Konrad. 2019. GermaLemma: A lemmatizer for German language text. `https://github.com/WZBSocialScienceCenter/germalemma`.

Florian Leitner. 2019. Syntok: Text tokenization and sentence segmentation (segtok v2). `https://github.com/fnl/syntok`.

Matthias Liebeck and Stefan Conrad. 2015. IWNLP: Inverse Wiktionary for natural language processing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 414–418, Beijing, China.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Melanie Neunerdt, Michael Reyer, and Rudolf Mathar. 2013. A POS tagger for social media texts trained on web comments. *Polibits*, 48:61–68.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, and Nathan Schneider. 2012. Part-of-speech tagging for Twitter: Word clusters and other advances. Technical report, School of Computer Science, Carnegie Mellon University.

Thomas Proisl and Peter Uhrig. 2016. SoMaJo: State-of-the-art tokenization for German web and social media texts. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task*, pages 57–62, Berlin, Germany. Association for Computational Linguistics (ACL).

Thomas Proisl. 2018. SoMeWeTa: A part-of-speech tagger for German social media and web texts. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 665–670, Miyazaki, Japan. European Language Resources Association ELRA.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium, October. Association for Computational Linguistics.

Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. *Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset)*. Retrieved from `http://www.sfs.uni-tuebingen.de/resources/stts-1999.pdf`.

Helmut Schmid and Florian Laws. 2008. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 777–784, Manchester, UK.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *Proceedings of the ACL SIGDAT Workshop*, Dublin, Ireland.

Helmut Schmid. 2019. RNNTagger – a neural part-of-speech tagger. `https://www.cis.uni-muenchen.de/~schmid/tools/RNNTagger/`.

Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin. 2009. A new hybrid dependency parser for German. In *Proceedings of the GSCL Conference*, Potsdam, Germany.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting synergies between open resources for German dependency parsing, POS-tagging, and morphological analysis. In *Proceedings of Recent Advances in Natural Language Processing*, pages 601–609, Hissar, Bulgaria.

Tom De Smedt and Walter Daelemans. 2012. Pattern for Python. *Journal of Machine Learning Research*, 13:2063–2067.

Ramon Ziai and Niels Ott. 2014. RFTagger Java interface. `http://sifnos.sfs.uni-tuebingen.de/resource/A4/rftj/`.