A German POS Tagger for NLTK

Darius Weber
RWTH Aachen
darius.weber@rwth-aachen.de

Thomas Maximilian Schöller
Ulm University
thomas.schoeller@uni-ulm.de

Abstract

We show that the recommended part-of-speech (POS) tagger of the NLTK Python library achieves state-of-the-art results for German POS tagging. After training the tagger on 90% of the German TIGER corpus, the tagger achieves 97% accuracy. We also evaluated the performance on five corpora from different domains, showing robustness on out-of-domain texts.

1 Introduction

Part-of-speech (POS) tagging is a foundational task in natural language processing, enabling higher-level linguistic analyses such as syntactic parsing and machine translation. It helps in identifying sentence structure and meaning. The Python Library NLTK recommends the use of its averaged perceptron tagger for this task. It has shown state-of-the-art performance for English texts (Honnibal). However, the project only provides weights for Russian and English. This paper addresses this gap by examining the performance of the averaged perceptron POS tagger on German texts. We find accuracy levels of 96-97% when the tagger is trained on 90% of the German corpus TIGER. A baseline comparison to a tagger based on Naïve Bayes shows the competitive performance of our tagger. Furthermore, we show robustness on out-of-domain texts. On corpora from five different domains we report a mean drop in accuracy of around 6.5%. Our error analysis shows that infrequent tags contribute significantly to misclassification errors.

2 Related Work

Wu and Dredze demonstrated the effectiveness of transformer-based models for POS tagging, achieving accuracies of 92.1% to 99% for 15 languages using a BERT model. They achieved 95.2% for German POS tagging.

The accuracy of POS tagging on various German corpora from different domains has been investigated by Ortmann et al.. They reported accuracies between 88.2% and 94.3% for several free and open-source POS taggers. Their findings highlight the variability in performance when moving beyond the domain of training corpora.

Nolte worked on German POS tagging using NLTK and extended the `ClassifierBasedTagger`. This classifier is based on Naïve Bayes. They reported an accuracy of 96% after training the tagger on 90% of the TIGER corpus.

3 Methods

3.1 POS Tagging in NLTK

POS tagging is a sequence-labeling task. Its goal is to assign a POS tag to each word in a text. For this task, the NLTK project currently recommends the use of its implementation of a POS tagger based on the averaged perceptron[1]. Collins introduced the usage of the averaged perceptron approach to POS tagging based on the work of Freund and Schapire.

The NLTK averaged perceptron POS tagger has built-in support for English and Russian[2]. The following sections describe the algorithm used with a particular focus on the specifics of the NLTK implementation.

3.1.1 Normalization

As a pre-processing step before training or predicting, the input tokens are normalized. All words are lower cased, numbers composed of four digits are replaced with a special `!YEAR` token, and longer numbers are replaced with a `!DIGITS` token.

[1] As of October 2024, https://github.com/nltk/nltk/blob/9a5622f8a5b228df9499cd03181d9f8491e39f17/nltk/tag/__init__.py#L145

[2] See: https://github.com/nltk/nltk/blob/9a5622f8a5b228df9499cd03181d9f8491e39f17/nltk/tag/__init__.py#L115

Feature Key in Dictionary	Description
`bias`	Bias
`i suffix aft`	Last three characters of the token
`i pref1 M`	First character of the token
`i-1 tag ART`	Tag of the previous token
`i-2 tag -START-`	Tag from two tokens prior
`i tag+i-2 tag ART -START-`	Tags of the preceding two tokens
`i word mitgliedschaft`	Current token
`i-1 tag+i word ART mitgliedschaft`	Tag of the preceding token & current token
`i-1 word die`	Preceding token
`i-1 suffix die`	Three last letters of the preceding token
`i-2 word -START2-`	Token two positions to the left
`i+1 word erfolgt`	Successor word
`i+1 suffix lgt`	Suffix of successor word
`i+2 word für`	Token two possions to the right

Table 1: Features extracted for the word "Mitgliedschaft" in the sentence "Die Mitgliedschaft erfolgt für eine der Unterabteilungen".

3.1.2 Features

According to the author of the NLTK implementation (Honnibal), the feature list was derived from Brown clusters created by Turian et al. The feature list was selected by experimentation to reduce the amount of required memory. The context window encompasses the preceding two tokens and tags and the following two words. Table 1 lists all the features for an example token from a training sentence.

3.1.3 Predicting

To create the predicted POS tags for a given untagged sentence, the code loops over the individual tokens of the sentence. If the use of the tag dictionary is set and the token appears in the tag dictionary, the corresponding tag is returned. Otherwise, the prediction is created by passing the token features to the perceptron. The perceptron determines the most likely tag as the tag with the highest score. The score is calculated as a sum of the weights of each feature.

Prediction Algorithm

- **Input:** Feature vector $\mathbf{x}$ with $x_f = 1$ for each feature f.
- **Initialize** the score vector $\mathbf{S}$ where $S_t = 0$ for each known tag t.
- **For each tag** t:
  - Retrieve the weight vector $\mathbf{w}_t$ for tag t containing a weight for each feature f from the weights dictionary $\mathbf{w}$.
  - Calculate the score S_t as the dot product of the feature vector $\mathbf{x}$ and the weight vector $\mathbf{w}_t$:

$$S_t = \mathbf{x} \cdot \mathbf{w}_t$$

- **Select the tag** t^* with the highest score:

$$t^* = \max S_t$$

3.1.4 Training

To handle unambiguous cases, words that appear frequently in training data together with the same POS tag are added to a lookup dictionary. Two hard-coded parameters decide whether a word is added: The word must appear at least 20 times in the training data and in at least 97% of the cases, the word must have the same associated POS tag. The use of the tag dictionary can be deactivated when predicting POS tags.

The subsequent nested training loops iterate over all sentences and their individual tokens. The features are extracted and a prediction is created using the perceptron algorithm (see 3.1.3). If the prediction is incorrect, the model weights are updated accordingly. Multiple training iterations are possible. Each iteration consists of one pass of all training sentences and the sentences are shuffled before each pass.

The implementation of the averaging algorithm (originally introduced by Freund and Schapire)

works by summing up the value of each individual weight for each training example (and hence weight update) in a vector. The final weights are calculated by dividing the vector of summed-up weights through the number of updates. However, there is one trick used to optimize for time complexity: The sum is not actually updated for each individual weight for each training example. Instead, an additional vector is used to record the count of training examples seen whenever the corresponding weight is updated. Using this timestamp the sum value for the weight needs to be updated only when the individual weight changes saving on arithmetic operations. Using the averaged perceptron over the original perceptron algorithm introduced by Rosenblatt has the advantage of attaining a more stable learning curve helping with faster convergence. It also ensures that the resulting weights have a large margin which in turn improves accuracy.

3.2 Training Corpus

For the training corpus, we chose the TIGER corpus (Brants et al.). The TIGER corpus is one of the three major treebanks available for German (Rehbein and van Genabith). We chose this corpus because of its public availability without the requirement to sign an individual license agreement. There is a prospect that the corpus might be opened for commercial use in the future[3]. However (as of November 2024), the use is restricted to academic use only, preventing the integration of our derived works into the NLTK repository for now.

The TIGER corpus contains articles from the German Newspaper *Frankfurter Rundschau* published between November 1995 and March 1997. It is tagged using the Stuttgart-Tübingen TagSet (STTS). We used version 2.2 which was released in July 2012 [4]. This release contains 50,472 sentences and 888,238 tokens in total. The tokens are composed of 89,383 individual types. 85,255 of these types appear unambiguously (95.3%), i.e. these types do only have one associated POS tag in the corpus.

[3]As of November 2024, the use for commercial purposes is under review. See: https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/

[4]Available at: https://www.ims.uni-stuttgart.de/documents/ressourcen/korpora/tiger-corpus/download/start.html

4 Results

All of the scripts we used can be found in the project repository: https://github.com/MaxAtoms/NLTK-German-POS-Tagger. Our results can be automatically reproduced via a continuous integration pipeline and are available here: https://github.com/MaxAtoms/NLTK-German-POS-Tagger/releases.

For the empirical evaluation we considered several approaches. First, we experimented with a 90/10 train/test split on the German TIGER corpus to examine if the average perceptron POS tagger could reproduce state-of-the-art results. We distinguished between using the lookup dictionary (with tagdict) and not using it (without tagdict), in order to examine the difference between the performance of the averaged perceptron algorithm itself and the combined method of the perceptron algorithm and lookup dictionary. In the spirit of Ortmann et al., we also tested the generalization ability of the POS tagger. To do this, we trained the POS tagger with the lookup dictionary entirely on the TIGER corpus, using five different subcorpora as the test set. These subcorpora include Wikipedia (wikipedia.conll)[5], Novelette (novelette.conll)[6], Sermon (sermononline.conll)[7], TED (ted.conll)[8] and Movie (opensubtitles.conll)[9]. These are directly downloadable from https://github.com/rubcompling/konvens2019/tree/master/data/gold/balanced/annotations. See table 2 for details. Since the training time depends heavily on the number of iterations of the training algorithm, as we saw in section 3.1.4, we also tested the ability of the POS tagger over 1, 5, 10 and 15 iterations of training to test for possible convergence after a certain number of iterations. We report all results in terms of accuracy across all tag predictions in the corresponding test set.

[5]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).

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

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

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

[9]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

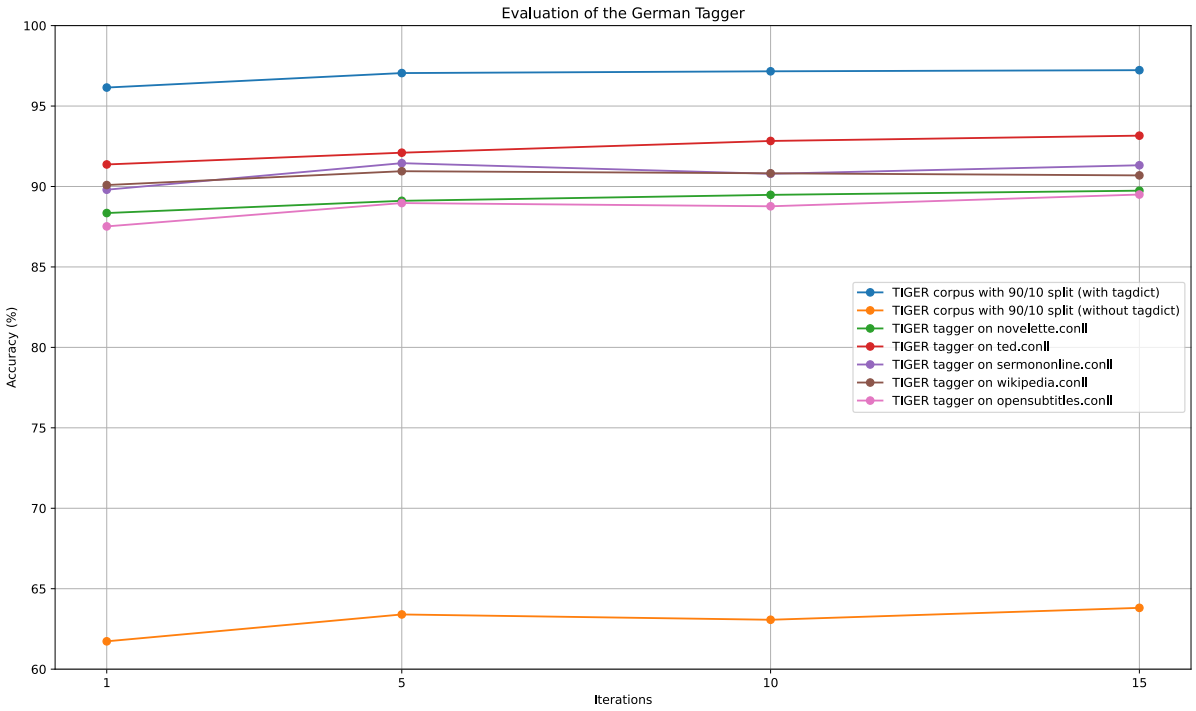

Figure 1: Results of the evaluation. Starting at 60% accuracy.

Subcorpus	#Tokens	#Sentences	#Docs
Wikipedia	1,514	96	12
Novelette	1,588	69	12
Sermon	1,520	90	16
TED	1,506	101	17
Movie	1,514	203	21

Table 2: Overview of the five subcorpora (Ortmann et al.).

The results are shown in Figure 1. The best accuracy is obtained using the TIGER 90/10 split with lookup dictionary with about 96-97% accuracy. Here, the tag dictionary comprises 2,870 types, a significantly lower number than the absolute number of unambiguous types in the overall training corpus (cf. 3.2). This reduction is caused by the decision parameters introduced in 3.1.4. The 96-97% accuracy is comparable to the performance of Nolte using Naïve Bayes on the TIGER corpus and also comparable to the performance of state-of-the-art English POS tagging results (see Manning). Leaving out the tag dictionary results in a 35% decrease in accuracy. This shows that the use of the tag dictionary is crucial to achieve high tagging accuracy, which is why the experiments on the subcorpora were all performed using the tag dictionary approach. When focusing on the results here, we can observe that the tagger archives highest results on the TED subcorpus with around 92-93% accuracy and lowest accuracy on the Movie subcorpus with around 87-89% accuracy. In summary, we can say that the tagger reaches accuracies between 87-93% on out-of-domain corpora, which positively confirms the generalization ability of the average perceptron POS tagger. Looking at all the results, we can see that using more iterations in training results in a small increase in performance, most noticeable from one to five iterations. Surprisingly, we can also get a decrease in performance when looking at the TIGER 90/10 split without lookup dictionary or the tagger on the Sermon, Wikipedia and Movie subcorpus from 5 to 10 iterations. Possible explanations are overfitting, where higher iterations tend to memorize the training data rather than learning general patterns, or overspecialization. Overspecialization occurs when the model becomes too finely tuned to the specific characteristics of the training data, resulting in a decrease of performance when evaluating on other corpora from different domains. The various domains are illustrated in Table 3.

4.1 Error Analysis of the POS Tagger

Figure 1 leaves open the question of which tags the average perceptron POS tagger misses most often, contributing to the lack of accuracy between 3-13%. To investigate this we have to consider the

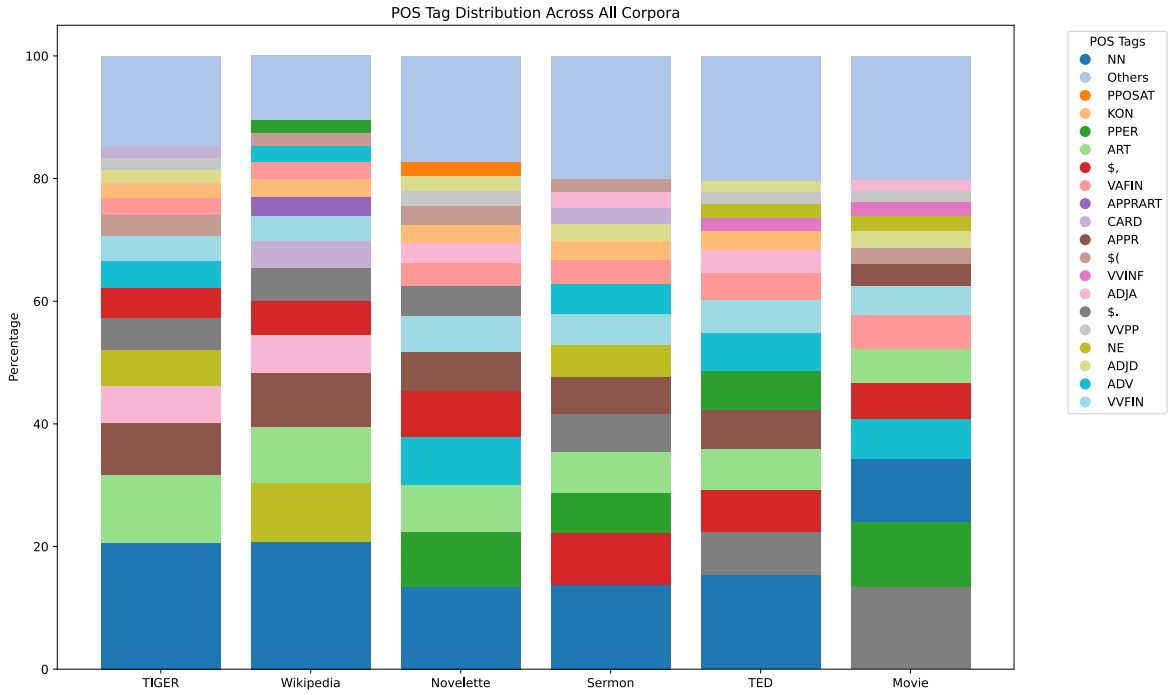

Figure 2: Distribution of POS tags for each corpus. Different colors indicate different tags.

Corpus	Domain
TIGER	written, journalistic
Wikipedia	written, encyclopedic
Novelette	written, prose
Sermon	spoken, planned
TED	spoken, planned
Movie	spoken

Table 3: Summary of the six corpus domains. Inspired by (Ortmann et al.).

distribution of the tags for each corpus as shown in Figure 2. For each corpus we displayed the top 15 most common tags such as nouns, articles or punctuation marks. See appendix A for an explanation of the tags. Mostly all tags that are classified under "Others" do not reach more than 2% of the total tags. With this in mind, we can observe if the tagger fails on edge cases, i.e. tags that do not occur very often, or if it generally fails to distinguish a certain class of tags. Figure 3 illustrates the prediction errors across the different test sets using the average perceptron POS tagger after five iterations of training as in Figure 1. We colored the tags that appear in "Others" in Figure 2 in blue and otherwise in red, so that we can observe the edge cases. Immediately we see that the most common wrong predictions are mostly coloured red.

These include, quite frequently, named entities (*NE*), adverbs (*ADV*), singular nouns (*NN*) and finite verbs (*VVFIN*). This makes sense, as these categories are often challenging due to ambiguity, inflection and flexible word order in German. However, especially in the subcorpora, we can also observe a failure in edge cases marked in blue. In the TED corpus, for example, the tagger frequently fails on sentence-internal punctuation (e.g., *$(*) and foreign language material (*FM*) in Wikipedia, as well as on substituting indefinite pronouns (*PIS*) in TIGER and Novelette. In general, we observe that while infrequent tags constitute far less than 33% of each corpus, they account for approximately 33% of misclassification errors. Special cases therefore play an important role in POS tagging.

4.2 Baseline Comparison

We used the extended NLTK `ClassifierBasedTagger` from Nolte as a baseline for comparison. Table 4 presents the accuracy differences between the average perceptron POS tagger (after five iterations) and the baseline model across various corpora. We trained the baseline model on exactly the same conditions with a 90/10 split for the TIGER evaluation, and for the evaluation on the five subcorpora, we used the entire TIGER corpus as the train set. The results indicate that the average perceptron POS

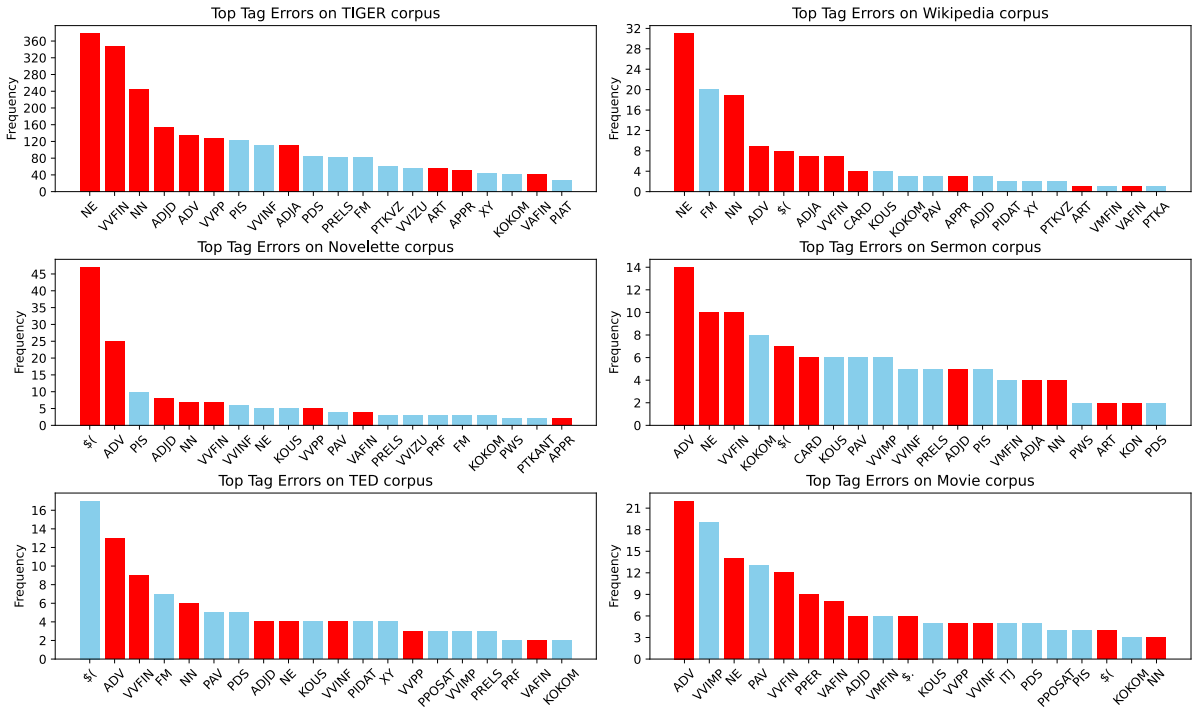

Figure 3: Most frequent prediction errors of the average perceptron POS tagger. Red color indicates that this tag occurs frequently in the corpus, blue indicates that it does not.

Corpus	Avg. Perc. (%)	Baseline(%)
TIGER 90/10	97.05	94.14
Wikipedia	90.95	86.72
Novelette	89.11	88.16
Sermon	91.45	87.76
TED	92.10	88.98
Movie	88.97	85.40

Table 4: Comparison of average perceptron POS tagger after five iterations and baseline model accuracy across corpora.

tagger consistently outperforms the baseline model by 1% to 4%, with the highest difference on the Wikipedia corpus and the smallest on the Novelette corpus.

5 Conclusion

We have shown that the averaged perceptron POS tagger provided by NLTK can achieve state-of-the-art accuracy for German text processing when trained on the TIGER corpus, attaining 97% accuracy. We have also shown a fairly strong generalization ability of 87-93% across five different out-of-domain subcorpora such as Wikipedia and TED. However, while the use of the TIGER corpus enables high POS tag performance, its licensing restrictions limit the immediate integration of our work into the NLTK repository for broader adoption.

Future work could explore the incorporation of additional features tailored to German, investigation of the decision parameters for the tag dictionary and domain adaption as shown by Daumé III. In addition, it would be interesting to measure the performance of the averaged perceptron POS tagger in terms of sentence accuracy across the corpora, as it was done by Manning.

References

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

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics.

Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263. Association for Computational Linguistics.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. 37(3):277–296.

Matthew Honnibal. A good POS tagger in about 200 lines of python.

Christopher D. Manning. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189. Springer.

Philipp Nolte. NLTK-contributions.

Katrin Ortmann, Adam Roussel, and Stefanie Dipper. Evaluating off-the-shelf NLP tools for german. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers*, pages 212–222. German Society for Computational Linguistics & Language Technology.

Ines Rehbein and Josef van Genabith. Why is it so difficult to compare treebanks? TIGER and TüBa-d/z revisited.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. 65(6):386–408.

O. Tange. GNU parallel - the command-line power tool. 36(1):42–47. Place: Frederiksberg, Denmark.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844. Association for Computational Linguistics.

A Appendix

A.1 List of tags from the Stuttgart-Tübingen-TagSet

Tag	Description	Example
ADJA	adjective, attributive	[das] große [Haus]
ADJD	adjective, adverbial or predicative	[er fährt] schnell, [er ist] schnell
ADV	adverb	schon, bald, doch
APPR	preposition; circumposition left	in [der Stadt], ohne [mich]
APPRART	preposition with article	im [Haus], zur [Sache]
APPO	postposition	[ihm] zufolge, [der Sache] wegen
APZR	circumposition right	[von jetzt] an
ART	definite or indefinite article	der, die, das, ein, eine, ...
CARD	cardinal number	zwei [Männer], [im Jahre] 1994
FM	foreign language material	[Er hat das mit “] A big fish [” übersetzt]
ITJ	interjection	mhm, ach, tja
KOUI	subordinate conjunction with zu and infinitive	um[zu leben], anstatt [zu fragen]
KOUS	subordinate conjunction with sentence	weil, daß, damit, wenn, ob
KON	coordinate conjunction	und, oder, aber
KOKOM	comparative conjunction	als, wie
NN	common noun	Tisch, Herr, [das] Reisen
NE	proper noun	Hans, Hamburg, HSV
PDS	substituting demonstrative pronoun	dieser, jener
PDAT	attributive demonstrative pronoun	jener [Mensch]
PIS	substituting indefinite pronoun	keiner, viele, man, niemand
PIAT	attributive indefinite pronoun without determiner	kein [Mensch], irgendein [Glas]
PIDAT	attributive indefinite pronoun with determiner	[ein] wenig [Wasser], [die] beiden [Brüder]
PPER	non-reflexive personal pronoun	ich, er, ihm, mich, dir
PPOSS	substituting possessive pronoun	meins, deiner
PPOSAT	attributive possessive pronoun	mein [Buch], deine [Mutter]
PRELS	substituting relative pronoun	[der Hund,] der
PRELAT	attributive relative pronoun	[der Mann,] dessen [Hund]
PRF	reflexive personal pronoun	sich, dich, mir
PWS	substituting interrogative pronoun	wer, was
PWAT	attributive interrogative pronoun	welche [Farbe], wessen [Hut]
PWAV	adverbial interrogative or relative pronoun	warum, wo, wann, worüber, wobei
PAV	pronominal adverb	dafür, dabei, deswegen, trotzdem
PTKZU	zu before infinitive	zu [gehen]
PTKNEG	negative particle	nicht
PTKVZ	separable verbal particle	[er kommt] an, [er fährt] rad
PTKANT	answer particle	ja, nein, danke, bitte
PTKA	particle with adjective or adverb	am [schönsten], zu [schnell]
SGML	SGML markup	turnid=n022k TS2004
SPELL	letter sequence	S-C-H-W-E-I-K-L
TRUNC	word remnant	An- [und Abreise]
VVFIN	finite verb, full	[du] gehst, [wir] kommen [an]
VVIMP	imperative, full	komm [!]
VVINF	infinitive, full	gehen, ankommen
VVIZU	infinitive with zu, full	anzukommen, loszulassen
VVPP	perfect participle, full	gegangen, angekommen
VAFIN	finite verb, auxiliary	[du] bist, [wir] werden
VAIMP	imperative, auxiliary	sei [ruhig!]

Table Continued

Tag	Description	Example
VAINF	infinitive, auxiliary	werden, sein
VAPP	perfect participle, auxiliary	gewesen
VMFIN	finite verb, modal	dürfen
VMINF	infinitive, modal	wollen
VMPP	perfect participle, modal	gekonnt, [er hat gehen] können
XY	non-word containing non-letter	3:7, H2O, D2XW3
$,	comma	,
$.	sentence-final punctuation mark	. ? !
$(	other sentence-internal punctuation mark	; :- [,](