The COW14 Tool Chain for German

Roland Schäfer
Linguistic Web Characterization (DFG)
Freie Universität Berlin
roland.schaefer@fu-berlin.de

Felix Bildhauer
Grammar Department
Institut für Deutsche Sprache Mannheim
bildhauer@ids-mannheim.de

Abstract

In this paper, we describe the COW tokenization and part-of-speech tagging pipeline with which we participated in the EmpiriST 2015 shared task. We briely discuss the original design goals for the tool chain and the minimal changes we made for the shared task. It should be noticed that we did not expect our system to perform competitively, especially not on CMC data and in the POS tagging track, and that we consequently viewed our system as an improved baseline system.

1 Original Design Goals

This section describes the original design goals of our tool chain.[1] Since we participtated with a production system only minimally adapted for the EmpiriST 2015 shared task, it is important to keep in mind the intended use and users of the system when evaluating the performance of and the errors made by the system.

The system with which we participated is the production system implemented for the construction of the COW and CommonCOW corpora that are described in Schäfer and Bildhauer (2012), Schäfer (2015), Schäfer (2016 to appear), but also (in comparison to other initiatives) in Biemann et al. (2013) and (implicitly) in textbook form in Schäfer and Bildhauer (2013).[2] Instead of developing custom tools for linguistic annotation—as we did for the non-linguistic preprocessing in the form of the *texrex* web page processor (Schäfer and Bildhauer, 2012; Schäfer, 2015; Schäfer, submitted)—we wrapped available tools and models (tokenizers, POS taggers and lemmatizers, morphological analyzers, named-entity recognizers, and dependency parsers).

Since the COW tool chain is a production system, we consider it vital to understand who our primary customers are, and what the resulting design goals were. We have a background in theoretical linguistics, especially morpho-syntax and semantics. For empirical work in these fields, corpora have become a major source of data. Since the focus in theoretical linguistics has shifted to rare phenomena, large web corpora containing a fair share of non-standard language are sometimes the only available source of data. However, our intended primary users (including ourselves) are not interested in the specifics of computer-mediated communication but rather in a broad view of variation and alternations occurring in present-day language. This means that linguistic annotations like POS tags can only be utilized by this type of user if they are near-perfect, and corpus queries consequently have a near-perfect recall. This is reflected in the behavior of our users.

From log analyses of the queries made by our users at `https://webcorpora.org`, we know that (as of 15 May 2016) POS tag specifications are used in only 14.8% (DECOW-only 13.7%), lemma information in 13.39% (DECOW-only 14.13%), and simple token specifications in 89.2% (DECOW-only 89.2%) of all queries. In other words, most linguists who use our web corpora do not use POS tagging—let alone other types of annotation—either because they are unaware of what such annotations can do for them, or because they cannot risk publishing corpus studies where material was not taken into account because the tagger made crucial errors, and queries specifying POS annotations did not return all the relevant targets. While low precision for corpus queries just means that corpus linguists have to filter their concordances by hand more thoroughly,

[1] The tools have been freely available since 2014. They are distributed under a permissive 2-clasue BSD license on GitHub (`https://github.com/rsling/cow`).

[2] `http://corporafromtheweb.org`

below 100% recall runs the danger of invalidating corpus studies.

As an example, the named entity annotation in DECOW14 was added because some productive DECOW users work on the morpho-syntax of German person names. After the real-life accuracy of the Stanford Named Entity Recognizer with the models from Faruqui and Padó (2010) was evaluated as unacceptable (Helmers, 2013), carefully designed heuristics and lists of names created by hand were used instead of the automatically generated annotation (Ackermann, to appear). The entire automatic annotation was essentially useless, at least for the intended use.

That said, improving POS tagging has never been our primary focus, simply because we are aware that near-perfect results cannot be achieved anyway. Since corpus query engines always use tokens as their basic unit for indexing, near-perfect tokenization is a de facto-requirement, however. This is why we invested considerable effort into tokenization but not into POS tagging. We describe the COW14 tool chain for German in Section 2, report the results on the EmpiriST 2015 data sets in Section 3 before summarizing the paper in Section 4.

2 Implementation

Before we go into the technical details of the COW14 tool chain, we would like to mention our main woe (as corpus creators) is that, even in 2016, most NLP tools generally do not come with the ability to process text contained in XML files, which—in the simplest and totally sufficient case—means skipping over anything in <> and treating the five canonical XML entities as their literal counterparts. The problem exists across the board with Ucto and TreeTagger discussed below, but also with all other tools we use, such as FreeLing (Carreras et al., 2004), the Stanford Named Entity Recognizer (Faruqui and Padó, 2010), mate-tools (Bohnet and Nivre, 2012), and Marmot (Mueller et al., 2013). Such problems are not usually tackled in shared tasks where accuracy is, of course, the only metric of interest and usability in large production systems is less relevant. That said, we now proceed to the technical details.

2.1 Tokenizer

The COW14 tool chain is merely a series of script wrappers around existing tools. Since we use our university's SLURM-based high-performance cluster for gigatoken corpus creation and SLURM is best controlled using Bash scripts, all tools are wrapped in Bash scripts.

We currently use the rule-based Ucto tokenizer for tokenization and heuristic sentence splitting (van Gompel et al., 2012). It is wrapped in a script that also performs some pre- and post-processing. Unfortunately, much of this processing goes into keeping Ucto from separating material that should not be separated. In general, we find it highly difficult to write clean rule sets for Ucto without triggering completely unpredictable side-effects.[3] Also, Ucto's added functionality of being able to discern different types of tokens (numbers, dates, etc.), while an interesting (yet little documented) feature, makes writing rule sets complicated and unpredictable. As a consequence, we are currently designing our own rule-based tokenizer and are planning to move our entire rule set from Ucto to the new system.

Pre-processing includes:

- converting XML entities to literals because Ucto cannot deal with XML entities
- marking certain kinds of strings in a way that they are not broken up by Ucto, for example double names written as *Kay-M.*, file names, DOIs, ISBNs, content-type declarations, dates and numbers with periods (otherwise often detected as sentence ends)
- pre-processing quotes to make sure they are always treated as separate tokens
- pre-processing obfuscated email addresses with *[at]* instead of @.

In the Ucto rule set, we have rules (some copied from the generic Ucto profile for German) which recognize, for example,

- email addresses and URLs
- dates and numbers
- various special abbreviation-like tokens such as *H&M* and *C++*
- number-letter combinations that should not be split such as *90-fach* (*90-fold*)
- over 250 custom abbreviations (plus some regexes which detect abbreviations heuristically)

[3] Most side effects, we think, are due to the fact that Ucto compiles the rules into complex regular expressions for the ICU library. For example, we observed cases where the scopes of matched groups and replacement operators were obviously mangled, leading to unsolicited replacements.

The post-processing mainly deals with restoring material that was protected from separation in preprocessing.

Virtually all of this was there before the EmpiriST 2015 shared task. We merely changed the way some tokens are treated. For example, we previously kept single-word strings with asterisks like *freu* as one token but split up similar multiword strings such *total␣freu* as *␣total␣freu␣* (4 tokens). However, we did not even attempt to achieve full compatibility with the EmpiriST guidelines.

2.2 POS Tagger

For POS tagging, we use the TreeTagger (Schmid, 1994) with the standard models. The Bash wrapper's main function is to set up a correct piping between different TreeTagger instances for tagging and chunking with other scripts in between. The only improvements we implemented in the wrapper concern regular expression-based recognition of smileys and other emoticons as well as some *blank*-tokens inserted by our web page processing system *texrex*. This applies after TreeTagger. It cannot be implemented by pre-tagging, simply because introducing new POS tags for smileys would require an extended tag set and consequently a retrained tagger model.

Other than that we only improved lemmatization and POS tagging by amending the tagger lexicon. We sorted the frequency list of tokens lemmatized as *unknown* in a large subset of DECOW14A and manually created a 3,800 entries long lexicon addition for TreeTagger with POS and lemmas in order to take care of the most frequent unknown words.[4] While this necessarily also improves the quality of the POS tagging in our full corpus, it most likely did not improve the results in the EmpiriST 2015 shared task, simply because the sample is so small that the added words do not occur in it with high enough frequencies.

3 Results

Table 1 summarizes the COW14 results on the EmpiriST 2015 web and CMC tokenization gold standard data sets according to the official evaluation script. The near-perfect performance on web data is not surprising because this is the type of data for which we optimized our tool chain. The remaining errors are analyzed in Table 2. Except for the serious error labeled *concatenation* and the split-up emoticon, all the other errors are irrlevant for our primary target users, as explained in Section 1.[5] In other words, for our purposes, tokenization is a solved problem.

Dataset	Prec	Rec	f_1
Web	99.84	99.71	99.77
CMC	98.31	98.07	98.18

Table 1: COW14 results on EmpiriST 2015 tokenization data sets

We do not discuss the tokenization errors for the CMC data set in detail because we did not optimize out tokenizer for such data. There are some errors like the *concatenation* error from Table 2, many emoticon which were broken up, and incorrect handling of run-together sentences. Since we normally fix run-together sentences before tokenization using a tool included in the *texrex* suite, the error rate can be expected to be lower on data that has run through our complete pipeline.

For completeness, we provide the tagging results are given in Table 3. We do not discuss the errors in detail because our contribution was non-competitive. Also, we did not implement the extensions to STTS introduced for EmpiriST (Beißwenger et al., 2016), and the evaluation against the extended tag set is virtually meaningless.[6]

4 Summary and Outlook

For our purposes, word-level tokenization of web texts is a solved task. Interestingly, rule-based approaches such as ours apparently achieve highly

[4] We stopped at the point where the list contained less than one fixable *unknown* token in a window of thirty tokens. The remaning unknowns are productively formed compounds and noise.

[5] We were surprised to see the emoticon being incorrectly tokenized because there *is* a rule in our Ucto rule set which should detect emoticons like :) We suspect that this is yet another Ucto side-effect, maybe introduced by the tweaks we made in order to conform to the EmpiriST format.

[6] We would like to point out that in a post-hoc evaluation on 18,993 tokens from DECOW14A that we did ourselves, we found that TreeTagger achieves an accuracy of 96.93% (standard STTS). However, this evaluation was confined to tokens occurring within regions labelled as sentences by our tool chain, and we apply a large number of heuristics in order to make sure that only very clean regions are labelled as sentences. Users are then advised to rely on annotation only in regions labeled as sentences. Thus, we provide the best possible quality of annotation for the subset of the corpus where high quality can be reached, and we leave all other (potentially noisy) material in the corpus for completeness (see also Section 1).

Count	Type	Example Gold	Example COW14
16	hyphenization	Herford␣–␣Altenbeken	Herford–Altenbeken
2	mixed alphanumeric tokens	R1	R␣1
1	truncated compounds	Viren-␣/␣Spywarescanner	Viren␣-␣/␣Spywarescanner
1	concatenation	eBook␣Das	eBookDas
1	emoticon	:)	:␣)
5	brackets	Geschichte[␣Bearbeiten␣]	Geschichte[Bearbeiten]
2	years with periods	1814.	1814␣.
4	punctuation clusters	:!:	:␣!␣:
2	dates	2015␣/11␣/22	2025␣/␣11␣/␣22

Table 2: Breakdown of tokenizer errors by types for the web data

Dataset	Accuracy STTS	Accuracy Extended
Web	92.96	91.82
CMC	81.49	77.89

Table 3: COW14 results on EmpiriST 2015 tagging data sets

competitive accuracy. In our own work, we will therefore rather focus on sentence segmentation, which appears to us to be less of a solved task.

As for POS tagging, the situation is far less satisfying, especially considering that even the EmpiriST 2015 winner in the web tagging track (UdS-distributional) achieved only 94.62% accuracy on the web data (plain STTS) according to the preliminary results released by the EmpiriST organizers. This is no more than 1.66% better compared to our baseline system. From a linguistic point of view, we have doubts that the kind of creative and non-standard language found on the web (and even more so in CMC) can be dealt with by extending tag sets and improving guidelines. Standardized categorization of non-standard data is—at least partly—a contradiction in terms.

Importantly, we suggest that discussions of changes or extensions to tag sets should involve the largest possible group of users because we see linguistic annotation purely as a service that we provide to make our corpora more usable for our users. Most of our users are not specifically interested in CMC, but rather in large corpora which contain a certain amount of non-standard grammar. Therefore, we are planning to conduct a survey among the users of our corpora and ask them how they use the linguistic annotation provided by us, and in which directions they would like to see them improve or change.

Acknowledgments

Roland Schäfer's contribution to this paper was funded by the German Research Council (Deutsche Forschungsgemeinschaft, DFG) through the personal grant SCHA1916/1-1 *Linguistic Web Characterization*.

We would like to thank the high performance computing (HPC) unit of ZEDAT data center at Freie Universität Berlin for the provided computing time.

References

Tanja Ackermann. to appear. From genitive inflective to possessive marker? – the development of german possessive -s with personal names. In Tanja Ackermann, Horst Simon, and Christian Zimmer, editors, *Germanic Genitives*.

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, social media and web corpora. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X)*, Berlin.

Chris Biemann, Felix Bildhauer, Stefan Evert, Dirk Goldhahn, Uwe Quasthoff, Roland Schäfer, Johannes Simon, Leonard Swiezinski, and Torsten Zesch. 2013. Scalable construction of high-quality web corpora. *Journal for Language Technology and Computational Linguistics*, 28(2):23–60.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea, July. Association for Computational Linguistics.

Xavier Carreras, Isaac Chao, Lluís Padró, and Muntsa Padró. 2004. Freeling: An open-source suite of language analyzers. In *Proceedings of the Fourth In-*

ternational Conference on Language Resources and Evaluation (LREC '04), pages 239–242.

Manaal Faruqui and Sebastian Padó. 2010. Training and evaluating a german named entity recognizer with semantic generalization. In *Proceedings of KONVENS 2010*, Saarbrücken, Germany.

Lea Arianna Helmers. 2013. Eigennamenerkenung in Web-Korpora des Deutschen - eine Herausforderung für die (Computer)linguistik. BA Thesis, Humboldt Universität, Berlin.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October. Association for Computational Linguistics.

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

Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 486–493, Istanbul. ELRA.

Roland Schäfer and Felix Bildhauer. 2013. *Web Corpus Construction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, San Francisco.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In Piotr Bański, Hanno Biber, Evelyn Breiteneder, Marc Kupietz, Harald Lüngen, and Andreas Witt, editors, *Proceedings of Challenges in the Management of Large Corpora 3 (CMLC-3)*, Lancaster. UCREL.

Roland Schäfer. 2016, to appear. CommonCOW: massively huge web corpora from CommonCrawl data and a method to distribute them freely under restrictive EU copyright laws. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC '16)*.

Roland Schäfer. submitted. Accurate and efficient general-purpose boilerplate detection for crawled web corpora. Accepted with revisions by Language Resources and Evaluation Journal.

Maarten van Gompel, Ko van der Sloot, and Antal van den Bosch. 2012. Ucto: Unicode tokeniser. version 0.5.3. reference guide. ILK Technical Report ILK 12-05, Induction of Linguistic Knowledge Research Group, Tilburg Centre for Cognition and Communication, Tilburg University, Tilburg, November.