sticker2: a Neural Syntax Annotator for Dutch and German

Daniël de Kok and **Neele Falk** and **Tobias Pütz**
Seminar für Sprachwissenschaft
University of Tübingen, Germany
{daniel.de-kok, neele.falk, tobias.puetz}@uni-tuebingen.de

Abstract

In this work, we introduce sticker2, a production-quality, neural syntax annotator for Dutch and German based on deep transformer networks. sticker2 uses multi-task learning to support simultaneous annotation of several syntactic layers (e.g. part-of-speech, morphology, lemmas, and syntactic dependencies). Moreover, sticker2 can finetune pretrained models such as XLM-RoBERTa (Conneau et al., 2019) for state-of-the-art accuracies.

To make use of the deep syntax models tractable for execution environments such as WebLicht (Hinrichs et al., 2010), we apply model distillation (Hinton et al., 2015) to reduce the model's size. Distillation results in models that are roughly 5.2-8.5 times smaller and 2.5-4.4 times faster, with only a small loss of accuracy.

sticker2 is widely available through its integration in WebLicht. Nix derivations and Docker images are made available for advanced users that want to use the models outside WebLicht.

1 Introduction

WebLicht (Hinrichs et al., 2010) is an environment for building and executing natural language processing chains. The primary goal of WebLicht is to provide easy access to a wide range of text processing tools to researchers in the humanities and social sciences. WebLicht has changed and grown considerably since its introduction 10 years ago. Its data exchange format, Text Corpus Format (Heid et al., 2010), has been updated to accommodate additional annotation layers, such as chunking and topological fields. The visualization of annotation layers in WebLicht has been improved considerably, and fine-grained search of annotations is now possible (Chernov et al., 2017). The number of annotation services, along with the types of annotations they provide and the languages that they support, has also grown steadily. Currently, WebLicht can apply syntactic analysis such as part-of-speech tagging (17 tools), lemmatization (11 tools) and dependency parsing (11 tools) for 46 languages. WebLicht also provides 6 different named entity recognition systems that work on four different languages.

A recent focus in WebLicht has been the addition of annotation tools based on neural network models. Neural network models provide state-of-the-art results for most natural language processing tasks, outperforming prior non-neural models. In this work, we introduce one such tool, sticker2. sticker2 is a high-accuracy, production-focused neural syntax annotator for Dutch and German, providing part-of-speech tag, lemma, morphology, dependency, and topological field (German) annotations. Since sticker2 combines existing ideas from the literature, this paper will focus on the design choices that were made to make sticker2 ready for production use. In Section 2 we will discuss the architecture of sticker2. The models for Dutch and German are discussed in Section 3. Finally, we will discuss the integration of sticker2 into WebLicht in Section 4.

2 sticker2

2.1 Architecture

At its core, sticker2 is a *sequence labeler* – it assigns, per task, a label to each token in a sentence. In order to do so, sticker2 first splits tokens into smaller pieces (Wu et al., 2016). For instance, the Dutch words *handelsorganisatie* 'trade organization' and *klopt* 'is correct/pulsates' are split into *handel-s-organisatie* and *klopt-t* respectively. This splitting of compounds and/or morphemes enable the use of relatively small vocabularies. Each such piece is represented as a mathematical vector that captures similarities between pieces. These vectors are fed to a transformer network (Vaswani et al., 2017). A tranformer network consists of multiple layers, where in each layer the representation of a piece is updated by attending to the representations of a sentence's pieces in the previous layer. This results in contextualized representations of each piece. For example, when *klop-t* co-occurs with *hart* 'heart', its representation could be specialized for its 'pulsation' sense. These contextualized representations are then weighted differently for each specific task (Peters et al., 2018; Kondratyuk and Straka, 2019) to extract the information that pertains to that task. Finally, distributions over possible labels are obtained by applying a classification layer to the task-specific representations. The model is trained end-to-end and simultaneously on all tasks (*multi-task training*). This allows the model to learn joint contextual representations for all tasks in the transformer.

Since such transformer networks typically require a larger amount of data to train than what is available in supervised training sets, sticker2 supports finetuning of pretrained models (Devlin et al., 2019; Conneau et al., 2019). These models are usually trained on a general task for which no supervised data is required, such as predicting randomly-masked words.

Besides basic sequence labeling, sticker2 also supports two forms of structural predictions: lemmatization and dependency parsing. In lemmatization, sticker2 predicts edit trees (Chrupała, 2008), which are applied to tokens after prediction to infer their lemmas. For dependency parsing, we use the dependency encoding scheme proposed by Spoustová and Spousta (2010). In this scheme, every token is annotated with a triple that contains: (1) the dependency relation of the token to its head, (2) the part-of-speech tag of the head; and (3) the relative position of the head in terms of its part-of-speech tags. For instance, the tag `nmod/noun/-2` means that a token should be attached with the *nmod* relation to the second preceding noun. After tagging, such tags are decoded to construct the dependency structure of a sentence.

2.2 Amenities for production use

In order to support sticker2 in production setups, we have focused on several aspects: speed, memory use, parallelization, and deployability. Prediction speed and memory use are, by and large, dominated by the size of the transformer network. In particular, prediction speed is dominated by the number of hidden layers and the hidden layer size, whereas memory use is dominated by the hidden layer size and the number of word or sentence piece embeddings.

To reduce the number of hidden layers and the hidden layer size, sticker2 supports model distillation (Hinton et al., 2015). Model distillation trains a new, smaller student model on the predictions of a larger teacher model on a large, unsupervised training set. After distillation, the student model is fine-tuned on the supervised training set. As we will show in Section 3, model distillation will result in drastically smaller and faster models at a marginal loss in accuracy. In addition to performing model distillation, we reduce the size of the models further by using smaller vocabularies in the student models.

We have implemented sticker2 in the Rust programming language.[1] The transformer models are also implemented in Rust using the linear algebra and back-propagation primitives that are made available through `libtorch` (Paszke et al., 2019). Rust programs compile to a single binary that can be deployed without requiring an additional language runtime or runtime packages. Rust's strong safety guarantees allow us to run sticker2 virtually lock-free in a single process. This makes it possible to deploy a single server process that can serve a large number of clients concurrently, while sharing resources such as the model.

[1] `https://www.rust-lang.org/`

3 Dutch and German syntax models

In this section, we describe the Dutch and German syntax models that we make available for sticker2. For both languages, we provide a fine-tuned XLM-RoBERTa base (Conneau et al., 2019) model, as well as smaller distilled models.

Dutch The Dutch model was fine-tuned and evaluated on the *Universal Dependencies* (UD) (Nivre et al., 2016) conversion of the Lassy Small treebank (Van Noord et al., 2013; Bouma and van Noord, 2017). This treebank consists of 65,147 sentences and 1,095,087 tokens. We split a random shuffle of the treebank sentences in 70/10/20% portions to obtain training, development, and held-out sets. We fine-tune the XLM-RoBERTa base model on the *universal part-of-speech tag*, *lemma*, *morphology tag*, and *universal dependency* layers of the treebank.

As discussed in Section 2.2, we use model distillation to create smaller and faster models. Since XLM-RoBERTa uses a large (multi-lingual) vocabulary of 250,000 sentence pieces, we use a smaller vocabulary of 30,000 word pieces for the distilled models.[2] Given the XLM-RoBERTa model, which has 12 layers (*l=12*), 768 hidden units (*h=768*), and 12 attention heads (*hd=12*), we extract the following two models: (1) h = 368, l = 12, hd = 12; and (2) h = 368, l = 6, hd = 12. The model was first distilled on the Lassy Large corpus (Van Noord et al., 2013) minus the sentences of Lassy Small, leading to a corpus of 47.6M sentences and 700M tokens. After distillation, the model is fine-tuned on the training set.

German The German model was trained on the UD conversion of TüBa-D/Z (Telljohann et al., 2005; Çöltekin et al., 2017), which consists of 104,787 sentences and 1,959,474 tokens. We use the same 70/10/20% split as for Dutch. The model is fine-tuned on the same layers as those described for Dutch, with the addition of a topological field layer. We distill models of the same transformer network and vocabulary sizes as for Dutch. The models are distilled on a mixture of Taz newspaper and Wikipedia subsections of the TüBa-D/DP (de Kok and Pütz, 2019) minus the sentences of TüBa-D/Z, consisting of 33.8M sentences and 648.6M tokens.

Results The accuracies of the Dutch and German models can be found in Table 1 and Table 2 respectively. The accuracy of dependency parsing is reported in *labeled attachment score* (LAS), which is the percentage of tokens that have the correct head and dependency relation. The tagging speed was measured in sentences per second on the held-out data on a Core i5-8259U mobile CPU with 4 threads. These results show that we can reduce the size and improve the speed of the models drastically for production use, such as in WebLicht, with relatively small reductions in accuracy.

Model	POS	Lemma	Morph	LAS	Size (MB)	Sent/sec
XLM-RoBERTa	98.89	99.04	98.87	93.13	1003	44
l = 12, h = 384, hd = 12	98.81	99.05	98.82	93.35	194	112
l = 6, h = 384, hd = 12	98.80	99.01	98.78	93.09	127	194

Table 1: Performance of XLM-RoBERTa and distilled models on the Lassy Small held-out data set.

Model	POS	Lemma	Morph	TF	LAS	Size (MiB)	Sent/sec
XLM-RoBERTa	99.24	99.33	98.35	98.14	95.59	1107	35
l = 12, h = 384, hd = 12	99.20	99.31	98.33	98.14	95.77	199	88
l = 6, h = 384, hd = 12	99.18	99.28	98.27	98.03	95.33	131	147

Table 2: Performance of XLM-RoBERTa and distilled models on the TüBa-D/Z held-out set.

[2]The reduction from 250,000 to 30,000 word pieces reduces the size of the word piece embeddings from 366MiB to 44MiB in the distilled models.

4 Integration into WebLicht

As shown in the previous section, the distilled models are much smaller and faster and can therefore be easily put into production. The WebLicht web services hosted in Tübingen are tested and deployed in production as *Docker* images. sticker2, having been integrated into WebLicht, also runs within its own isolated environment provided by the Docker platform. Several Docker containers are working together to offer the sticker2 service: One for each sticker2 worker and another one for a central service that is responsible for data conversion and communication. The central service, acting as a front-end to client requests, connects to the sticker2 workers, retrieves the processed data and converts it into the Text Corpus Format. This way the users can call the sticker2 web service via the WebLicht user interface and retrieve the syntactic annotations for their own data in a simple way.

5 Conclusion

In this work, we have introduced the sticker2 syntax annotation tool, as well as models for Dutch and German. We have shown that by using techniques such as model distillation, models can be made small and fast enough for high-accuracy annotation of large text corpora. Finally, we have described how sticker2 has been integrated into WebLicht, making it available as part of the CLARIN infrastructure.

sticker2 is available in the WebLicht interface[3] as *Sticker2-Dutch* and *Sticker2-German* or as a predefined easy chain. Advanced users can also use Nix derivations[4] or Docker images published through Docker Hub.[5]

In the near future, we hope to add support for additional languages, as large UD treebanks for those languages become available.

6 Acknowledgements

We would like to thank Erhard Hinrichs and Patricia Fischer for their feedback on this work. The financial support of the research reported in this paper is provided by the German Federal Ministry of Education and Research (BMBF), the Ministry of Science, Research and Art of the Federal State of Baden-Württemberg (MWK) as part of CLARIN-D and by the German Research Foundation (DFG) as part of the Collaborative Research Center 'The Construction of Meaning' (SFB 833), project A3.

References

Gosse Bouma and Gertjan van Noord. 2017. Increasing return on annotation investment: the automatic construction of a Universal Dependency treebank for Dutch. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 19–26, 5.

Alexandr Chernov, Erhard W. Hinrichs, and Marie Hinrichs. 2017. Search your own treebank. In Markus Dickinson, Jan Hajic, Sandra Kübler, and Adam Przepiórkowski, editors, *Proceedings of the 15th International Workshop on Treebanks and Linguistic Theories (TLT15), Bloomington, IN, USA, January 20-21, 2017*, volume 1779 of *CEUR Workshop Proceedings*, pages 25–34. CEUR-WS.org.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University.

Çağrı Çöltekin, Ben Campbell, Erhard Hinrichs, and Heike Telljohann. 2017. Converting the TüBa-D/Z treebank of German to Universal Dependencies. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 27–37, Gothenburg, Sweden, May.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Daniël de Kok and Sebastian Pütz. 2019. *Stylebook for the Tübingen treebank of dependency-parsed German (TüBa-D/DP)*. Seminar fur Sprachwissenschaft, Universitat Tübingen, Tübingen, Germany.

[3] `https://weblicht.sfs.uni-tuebingen.de/weblicht/`
[4] `https://github.com/stickeritis/nix-packages`
[5] `https://hub.docker.com/repository/docker/danieldk/sticker2`

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June.

Ulrich Heid, Helmut Schmid, Kerstin Eckart, and Erhard Hinrichs. 2010. A corpus representation format for linguistic web services: The D-SPIN text corpus format and its relationship with ISO standards. In *Proceedings of LREC 2010*, Valletta, Malta, May.

Erhard Hinrichs, Marie Hinrichs, and Thomas Zastrow. 2010. WebLicht: Web-based LRT services for German. In *Proceedings of the ACL 2010 System Demonstrations*, pages 25–29, Uppsala, Sweden, July.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of EMNLP-IJCNLP 2019*, pages 2779–2795, Hong Kong, China, November.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT 2018, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June.

Drahomíra Spoustová and Miroslav Spousta. 2010. Dependency parsing as a sequence labeling task. *The Prague Bulletin of Mathematical Linguistics*, 94:7–14.

Heike Telljohann, Erhard W Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. 2005. Stylebook for the Tübingen treebank of written German (TüBa-D/Z). In *Seminar für Sprachwissenschaft, Universität Tübingen, Germany*.

Gertjan Van Noord, Gosse Bouma, Frank Van Eynde, Daniël de Kok, Jelmer Van der Linde, Ineke Schuurman, Erik Tjong Kim Sang, and Vincent Vandeghinste. 2013. Large scale syntactic annotation of written Dutch: Lassy. In *Essential speech and language technology for Dutch*, pages 147–164. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.