Fast and accurate dependency parsing for Dutch and German

Daniël de Kok ME@DANIELDK.EU
Patricia Beer* PATRICIA.BEER@UNI-TUEBINGEN.DE

Seminar für Sprachwissenschaft, University of Tübingen, Germany

1. Introduction

Statistical dependency parsers that use transformer-based (Vaswani et al. 2017) pre-trained language models such as BERT (Devlin et al. 2019) and XLM-R (Conneau et al. 2020) show large improvements (Wu and Dredze 2019, Kondratyuk and Straka 2019, He and Choi 2019) over prior models. However, such deep transformer models are large and relatively inefficient, making them unfit for constrained environments such as mobile devices and CPU prediction on cloud virtual machines. In this work, we explore distillation (Hinton et al. 2015) as a means to compress XLM-R-based syntax annotators for Dutch and German. We start with a finetuned XML-R model as the baseline, showing that models can be made up to 10 times smaller and 4 times faster with a loss of at most 1.1 in labeled attachment score (LAS).

2. Baseline models

For our Dutch and German baseline syntax annotation models, we finetune the XLM-R Base language model (Conneau et al. 2020). XLM-R is a set of multilingual language models, trained on more than 2 TiB of text from 100 languages using the masked language model objective (Devlin et al. 2019). The XLM-R Base model uses a transformer with 12 hidden layers, a hidden dimensionality of 768, 12 attention heads, pointwise feed-forward layers with an inner dimensionality of 3,072, and a vocabulary consisting of 250,000 sentence pieces (Kudo and Richardson 2018).

Since we want to annotate multiple syntax layers, we use multi-task learning. For each task, we train a separate scalar weighting (Peters et al. 2018, Kondratyuk and Straka 2019) of the transformer layers to obtain a task-specific representation. For sequence labeling tasks, we apply a single-layer feed-forward ReLU layer to the task-specific representations, followed by a softmax layer to predict label distributions. For dependency parsing, we use biaffine classifiers (Dozat and Manning 2016) to predict the dependency edges and labels. Both the sequence labeling tasks and the biaffine classifiers are optimized using cross-entropy loss. We use label smoothing (Szegedy et al. 2016) to regularize the softmax classifiers.

We train the following syntax layers: Universal Dependencies (UD) part-of-speech, UD morphology, lemmas, and UD dependencies (Nivre et al. 2016). UD part-of-speech and morphology are predicted as unstructured labels. For lemmatization, we predict edit trees (Chrupała 2008), which are applied to tokens to infer their lemmas. For prediction of the dependency edges, we find the minimum spanning tree using the Chu-Liu-Edmonds algorithm (Chu 1965, Edmonds 1967, McDonald et al. 2005) with the weights predicted by the biaffine edge classifier.

3. Distillation

We use model distillation (Hinton et al. 2015) to create smaller and faster models. Model distillation transfers knowledge from a teacher model to a student model by training the student on predictions

of the teacher. In our case, the teachers are the Dutch and German baseline models discussed in Section 2. The student models also use the transformer architecture, but with different sets of hyperparameters. We will first discuss these hyperparameters in Section 3.1. We will then discuss the objectives that we use for distillation in Section 3.2.

3.1 Student model hyperparameters

Several transformer hyperparameters influence the speed and/or size of a transformer model, namely: the hidden layer dimensionality h, the number of hidden layers l, the vocabulary size v, the number of attention heads a, and the inner layer dimensionality i in the pointwise feed-forward layers. We will now discuss our choices for these hyperparameters in the student models.

In our initial experiments, we have found that reducing the number of attention heads or the inner layer dimensionality has a large negative impact on annotation accuracy. Thus, we will focus our attention to the remaining hyperparameters. Table 1 summarizes the time and space complexity of different components of the transformer as a function of h, i, l, v, and n (the sentence length in pieces). We have also included the actual sizes of these components in the XLM-R Base model for reference.

Component	**Space complexity**	**Time complexity**	**Size (MiB)**
Piece embeddings	$\mathcal{O}(vh)$	$\mathcal{O}(n)$	732
Self-attention	$\mathcal{O}(lh^2)$	$\mathcal{O}(nh^2 + n^2h)$	108
Pointwise feed-forward	$\mathcal{O}(lhi)$	$\mathcal{O}(nhi)$	216

Table 1: Space complexity, time complexity, and sizes of the different components of the XLM RoBERTa Base encoder. v = vocabulary size, h = hidden layer size, l = number of hidden layers, i = inner size of pointwise feed-forward layers, n = sentence length in pieces.

To reduce the size of the model, the largest compression can be gained by reducing the size of the vocabulary. Following in the footsteps of BERT (Devlin et al. 2019), many monolingual pretrained language models use a 30,000 piece vocabulary. Compared to the 250,000 piece XLM-R vocabulary, this is a ˜8.3x reduction. Following this tradition, we use a 30,000 piece vocabulary in all of our distilled models.

To speed up distilled models, the hidden layer dimensionality (h) or the number of hidden layers (l) should be lowered. From the perspective of time complexity, both hyperparameters are equally attractive, since the running time scales linearly with them. Recent empirical work has shown that within a given parameter budget it is preferable to have more layers over having a larger hidden dimensionality (Turc et al. 2019). Another benefit of reducing the hidden layer dimensionality is that it leads to another large space saving in the piece embedding matrix, since v is still a large factor. Therefore, we choose to reduce the hidden layer dimensionality for all distilled models from $h = 768$ in XLM-R Base to $h = 384$.

Besides using $v = 30,000$ and $h = 384$ for all distilled models, we explore two other variations. The first variation reduces the number of hidden layers from $l = 12$ in XLM-R Base to $l = 6$. This should again speed up processing and reduce the number of parameters in the self-attention and pointwise feed-forward layers by a factor of two. The second variation uses the ALBERT model (Lan et al. 2019), which decouples the dimensionality of piece embeddings e from the dimensionality of the hidden layers h and permits parameter sharing between transformer layers. We use these two features to reduce the embedding dimensionality to $e = 128$ and set the number of layer groups to $g = 6$, which means that with $l = 12$, every two layers share parameters.

3.2 Distillation objectives

The goal of distillation is to transfer knowledge from the teacher model to the student model. More specifically, given the same inputs, the student model should make the same predictions as the teacher. Following (Hinton et al. 2015), we optimize the student model to predict the same label distribution as the teacher. So, for a particular type of sequence label (e.g. universal part-of-speech), with label inventory Q, the distillation objective is the cross-entropy of predicted distributions of the teacher and the student for all data points $\mathbf{x}^1..\mathbf{x}^N$:

$$-\sum_{n=1}^{N}\sum_{q\in Q} p_{\text{teacher}}(q|\mathbf{x}^n)\log p_{\text{student}}(q|\mathbf{x}^n) \tag{1}$$

We use a similar objective for dependency edges in biaffine parsing, but rather than predicting a label from a label inventory Q, the models predict the head given all pieces in the sentence.

Hinton et al. (2015) also propose to apply softmax temperature, in which logits are divided by a temperature before softmax normalization. This makes the teacher distributions more uniform, providing more information to the student. We found that this was not necessary for our models since we use label smoothing (Section 2). In fact, applying a softmax temperture $T > 1$ often resulted in the label distributions becoming nearly uniform.

In addition to learning to emulate the softmax distribution of the teacher, we found that learning to predict the embeddings and hidden layer outputs of each transformer layer in the student leads to quicker convergence. Following TinyBERT (Jiao et al. 2020), we add the mean squared error of the teacher's hidden layer outputs and the student's hidden layer outputs as an additional distillation objective. Since the student's hidden layer size is smaller than that of the teacher, we apply a (learned) linear mapping to project the student's hidden layer space to that of the teacher. For the student models that only have 6 layers, the student learns to predict the outputs of the embedding layer of the teacher, as well as the output of every second hidden layer of the teacher.

4. Evaluation methodology

We will now describe the training and evaluation methodology for the Dutch and German models.

Data For both languages, we use training, development, and held-out data sets. The development set is used to tune additional model hyperparameters (such as the learning rate) and to determine when training has converged. The held-out data set is only used to report the results in Section 5. For Dutch, the data sets are based on the UD conversion of the Lassy Small treebank (Van Noord et al. 2013, Bouma and van Noord 2017), consisting of 65,147 sentences and 1,095,087 tokens. For German, we use the UD conversion of TüBa-D/Z release 11 (Telljohann et al. 2005, Çöltekin et al. 2017), consisting of 104,787 sentences and 1,959,474 tokens. We split a random shuffle of each treebank in 70%/10%/20% portions for the training, development, and held-out data sets.

Distillation is carried out on a large, unannotated corpus. For Dutch, we use sentences from the Lassy Large Treebank (Van Noord et al. 2013), with the sentences of Lassy Small removed (47.6M sentences, 700M tokens). For German, we use the Taz newspaper and Wikipedia subsections of the TüBa-D/DP (de Kok and Pütz 2019) minus the sentences of TüBa-D/Z (33.8M sentences, 648.6M tokens).

Training The Dutch and German baseline models are created by finetuning XLM-R Base on the Dutch and German training sets until the models converge. We then distill the models described in Section 3.1 from the baseline models using the unannotated corpus of the corresponding language. Finally, we also finetune the distilled models on the training sets until they converge.

Evaluation Biaffine dependency parsing is evaluated using the labeled attachment score (Kübler et al. 2009), excluding punctuation (LAS). For all other annotation layers, we report their accuracy.

5. Results

The annotation speeds and sizes of the Dutch and German models are shown in Table 2. We can see that the smaller, distilled models are indeed much faster than the baseline model. For example, the reduction of the hidden layer size from $h = 768$ to $h = 384$ alone results in a 2.2x speedup in CPU prediction (61 sent/s to 135 sent/s) for Dutch. Reducing the number of hidden layers from $l = 12$ to $l = 6$ provides another 1.8x speedup, resulting in an annotation speed of 240 sent/s. Similar speedups can be observed for German. Use of the ALBERT architecture ($l = 12$, $g = 6$) does not improve annotation speed, but does leads to a smaller model of only 104MiB.

Model	**Size (MiB)**	**Sents/s CPU**	**Sents/s GPU**	**Pieces/s CPU**	**Pieces/s GPU**
Dutch					
Finetuned XLM-R base	1,087	61	755	1,644	20,389
h = 384, l = 12, g = 12, e = 384	200	135	1,450	3,166	33,947
h = 384, l = 12, g = 6 e = 128	104	137	1,463	3,218	34,250
h = 384, l = 6, g = 6, e = 384	133	240	2,359	5,626	55,249
German					
Finetuned XLM-R base	1,104	50	614	1,560	19,085
h = 384, l = 12, g = 12, e = 384	208	105	1,131	2,929	31,489
h = 384, l = 12, g = 6, e = 128	111	105	1,131	2,931	31,482
h = 384, l = 6, g = 6, e = 384	140	180	1,748	5,024	48,688

Table 2: Size and speed of the Dutch and German models. CPU speed is measured on an AMD Ryzen 3700X with 4 threads, GPU speed on an NVIDIA RTX 2060 Super.

The distilled models post lower accuracies, as shown in Table 3. This can be expected, since the distilled models have a much smaller number of parameters. However, we believe these trade-offs to be acceptable in many applications. For example a Dutch syntax annotation model that is 2.2x faster and 5.4x smaller model can be had at a loss of only 0.46 LAS.

Model	**UD POS**	**Morph**	**Lemma**	**LAS**
Dutch				
Finetuned XLM-R base	98.90	98.87	99.03	94.37
h = 384, l = 12, g = 12, e = 384	98.83	98.80	99.03	93.91
h = 384, l = 12, g = 6, e = 128	98.81	98.76	99.00	93.79
h = 384, l = 6, g = 6, e = 384	98.80	98.79	99.05	93.42
German				
Finetuned XLM-R base	99.54	98.38	99.34	96.59
h = 384, l = 12, g = 12, e = 384	99.50	98.31	99.31	96.17
h = 384, l = 12, g = 6, e = 128	99.46	98.23	99.27	95.85
h = 384, l = 6, g = 6, e = 384	99.46	98.20	99.29	95.48

Table 3: Accuracies of the Dutch and German models.

6. Conclusion

In this abstract, we have explored model distillation as a means to produce smaller and faster transformer models from a finetuned, pretrained language model. The results show that models can be speeded up and compressed considerably with a loss in accuracy that is acceptable in many applications.

References

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

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

Chu, Yoeng-Jin (1965), On the shortest arborescence of a directed graph, *Scientia Sinica* **14**, pp. 1396–1400.

Çöltekin, Çağrı, Ben Campbell, Erhard Hinrichs, and Heike Telljohann (2017), Converting the TüBa-D/Z treebank of German to Universal Dependencies, *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, Gothenburg, Sweden, pp. 27–37. https://www.aclweb.org/anthology/W17-0404.

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov (2020), Unsupervised cross-lingual representation learning at scale, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, pp. 8440–8451. https://www.aclweb.org/anthology/2020.acl-main.747.

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

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019), BERT: Pre-training of deep bidirectional transformers for language understanding, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, Minneapolis, Minnesota, pp. 4171–4186. https://www.aclweb.org/anthology/N19-1423.

Dozat, Timothy and Christopher D Manning (2016), Deep biaffine attention for neural dependency parsing, *arXiv preprint arXiv:1611.01734*.

Edmonds, Jack (1967), Optimum branchings, *Journal of Research of the national Bureau of Standards B* **71** (4), pp. 233–240.

He, Han and Jinho D Choi (2019), Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with bert, *arXiv preprint arXiv:1908.04943*.

Hinton, Geoffrey, Oriol Vinyals, and Jeffrey Dean (2015), Distilling the knowledge in a neural network, *NIPS Deep Learning and Representation Learning Workshop*. http://arxiv.org/abs/1503.02531.

Jiao, Xiaoqi, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu (2020), TinyBERT: Distilling BERT for natural language understanding, *Findings of the Association for Computational Linguistics: EMNLP 2020*, Association for Computational Linguistics, Online, pp. 4163–4174. https://www.aclweb.org/anthology/2020.findings-emnlp.372.

Kondratyuk, Dan and Milan Straka (2019), 75 languages, 1 model: Parsing Universal Dependencies universally, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Hong Kong, China, pp. 2779–2795. https://www.aclweb.org/anthology/D19-1279.

Kübler, Sandra, Ryan McDonald, and Joakim Nivre (2009), Dependency parsing, *Synthesis lectures on human language technologies* **1** (1), pp. 1–127, Morgan & Claypool Publishers.

Kudo, Taku and John Richardson (2018), SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, Brussels, Belgium, pp. 66–71. https://www.aclweb.org/anthology/D18-2012.

Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut (2019), Albert: A lite bert for self-supervised learning of language representations, *arXiv preprint arXiv:1909.11942*.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajic (2005), Non-projective dependency parsing using spanning tree algorithms, *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pp. 523–530.

Nivre, Joakim, 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, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1659–1666.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018), Deep contextualized word representations, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Association for Computational Linguistics, New Orleans, Louisiana, pp. 2227–2237. https://www.aclweb.org/anthology/N18-1202.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna (2016), Rethinking the inception architecture for computer vision, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

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

Turc, Iulia, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019), Well-read students learn better: On the importance of pre-training compact models, *arXiv preprint arXiv:1908.08962v2*.

Van Noord, Gertjan, 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, *Essential speech and language technology for Dutch*, Springer, pp. 147–164.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin (2017), Attention is all you need, *in* Guyon, I., U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, Vol. 30, Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wu, Shijie and Mark Dredze (2019), Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*

(EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, pp. 833–844. https://www.aclweb.org/anthology/D19-1077.