Computational Linguistics in the Netherlands Journal 10 (2020) 91–107 Submitted 10/2020; Published 12/2020

Self-distillation for German and Dutch dependency parsing

Daniël de Kok* ME@DANIELDK.EU
Tobias Pütz* TOBIAS.PUETZ@UNI-TUEBINGEN.DE

*Seminar für Sprachwissenschaft, University of Tübingen, Germany

Abstract

In this paper, we explore self-distillation as a means to improve statistical dependency parsing models for Dutch and German over purely supervised training. Self-distillation (Furlanello et al. 2018) trains a new student model on the output of an existing (weaker) teacher model. In contrast to most previous work on self-distillation, we perform distillation using a large, unannotated corpus.

We show that in dependency parsing as sequence labeling (Spoustová and Spousta 2010, Strzyz et al. 2019), self-distillation plus finetuning provides large improvements over models that use supervised training. We carry out experiments on the German TüBa-D/Z universal dependency (UD) treebank (Çöltekin et al. 2017) and the UD conversion of the Dutch Lassy Small treebank (Bouma and van Noord 2017). We find that self-distillation improves German parsing accuracy of a bidirectional LSTM parser from 92.23 to 94.33 Labeled Attachment Score (LAS). Similarly, on Dutch we see improvement from 89.89 to 91.84 LAS.

1. Introduction

While modern statistical (Chen and Manning 2014, Kiperwasser and Goldberg 2016, Dozat and Manning 2017) and SAVG-based[1] parsers (Van Noord et al. 2006) have significantly pushed the state-of-the-art in parsing, there is a certain subset of dependencies that have proven to be particularly hard to parse. Table 1 shows the ten most frequent errors by UD relation (Nivre et al. 2020) of baseline BiLSTM baseline parsers (Section 5) for Dutch and German. Further analysis reveals that for Dutch and German approximately 21% and 25% of the attachment errors concern prepositional phrase attachment errors, where the noun of the prepositional phrase is either attached to a verb with the *obl* relation or to a noun with the *nmod* relation. Approximately 10% (Dutch) to over 12% (German) concern subject-direct object ambiguities, which manifest in incorrect attachment of noun phrases with the *nsubj* or *obj* labels. Resolving PP-attachment and subject-object ambiguities are thus still the largest challenges when parsing Dutch and German.

These relations are particularly difficult to parse, because the correct attachment is not only determined by syntactic constraints, but also by semantic preferences, or so-called selectional preferences (Katz and Fodor 1963, Wilks 1975, Boguraev 1979). As shown by Van Noord (2007), hand-corrected treebanks are generally too small to contain enough examples to model such selectional preferences. However, Van Noord (2007) also demonstrates that when a parser analyzes the large majority of attachments correctly, that selectional preferences can be modelled using the output of the parser. The reason that a parser can predict most attachments correctly is that it can rely on canonical word orders. For instance, the large majority of subjects and objects are attached correctly because parsers can rely on the general preference in Dutch and German to order the subject before the direct object.

In many prior works, machine-parsed data was used to extract PP attachment preferences (Hindle and Rooth 1993, Pantel and Lin 2000, Mirroshandel et al. 2012), whereas other work used the same principle to model attachment preferences for each dependency label in a label inventory (Johnson

1. Stochastic attribute-value grammar

Relation	% of errors
obl	11.14
nmod	9.97
fixed	7.79
parataxis	6.92
nsubj	5.69
amod	5.22
obj	4.72
conj	4.56
root	4.27
advmod	4.22

(a) Dutch

Relation	% of errors
obl	14.73
nmod	10.19
conj	8.45
parataxis	6.70
nsubj	6.40
amod	6.40
obj	6.32
appos	4.98
root	4.73
advmod	4.64

(b) German

Table 1

and Riezler 2000, Van Noord 2007, Fischer et al. 2019). These works use relatively simple word-based association strength metrics, such as point-wise mutual information or attachment probability. However, these metrics suffer from the curse of dimensionality (Bengio et al. 2001), since there is a large number of valid associations between heads, dependents, and dependency relations, which are unlikely to be all seen sufficiently often in training corpora. Given this problem, it is likely that we could improve on these approaches by modeling preferences based on word embeddings rather than concrete words. For example, we would like a model to learn that the verb *eat* takes objects from the subspace of edible things, rather than specific pairs such as *eat rice* or *eat cauliflower*.

There have been various attempts to adress the shortcomings of these simpler word-based association scores by modeling attachment preferences using word embeddings. Skipgram-based embedding models (Mikolov et al. 2013) use a probability function $S(w, c) = \sigma(W_w, C_c)$ where W and C are word and context embedding matrices. W and C are optimized such that $S(w, c) = 1$ if w and c co-occur or $S(w, c) = 0$ otherwise. In most applications, C is discarded and W is used as a word embedding matrix. However, the function S can be used as an indicator of association strength. Kiperwasser and Goldberg (2015) use such a function in a graph-based dependency parser. They find that the embedding-based scoring function improves over the baseline parsing model, but does not outperform pointwise mutual information as a metric for association strength. Fischer et al. (2019) reproduce this result in a transition-based dependency parser.

Van Noord, 2016 (personal communication) proposes another method for using word embeddings to compute attachment preferences. For each head h governing over a dependent with the relation r, $h \xrightarrow{r} *$, a slot embedding is computed by taking the weighted average of the embeddings of the words occuring in the slot $*$ in a large corpus. During parsing, the association scores of a head h and candidate c with the relation r is computed as the cosine similarity between the slot embedding of $h \xrightarrow{r} *$ and the embedding of c. This approach is evaluated by using the scores in Alpino's maximum entropy disambiguation model (Van Noord et al. 2006). The results are similar to what was found in other approaches that incorporated embeddings – the use of the embedding-based metric improves over the baseline (90.73 concept accuracy over 90.44 CA), but use of the embedding-based metric does not outperform the use of pointwise mutual information (90.84 CA). Combining both embedding-based association scores with pointwise mutual information does lead to a small improvement (90.87 CA).

While the use of embeddings for modelling selection preferences has, so far, not been more succesful than the use of simpler word-based association scores, we believe that these approaches have two important shortcomings. First, they use rely on functions that result in scalar values, discarding the high-dimensional space that embeddings provide. Second, attachment preferences

are learned as a separate step, rather than being fully integrated in the training of the parsing model. The approach that we take in this paper to learn attachment preferences is very simple – we train a model on the output of a baseline parser while it parses a large, unannotated, corpus. In this process of so-called *self-distillation* the newly trained *student model* sees a wider variety of language than the baseline *teacher* model that was trained on a small hand-corrected treebank. The hope is that since the teacher predicts the large majority of dependency relations correctly, the student model will learn regularities with regards to the attachment of previously unseen tokens. However, in contrast to the prior approaches, the attachment preferences are not condensed into scalar values and learning attachment preferences is fully integrated in model training.

The remainder of the paper is structured as follows. In Section 2 we will discuss prior work on self-distillation in more detail. Section 3 describes methodology for encoding dependency relations that we use in the experiments. Section 4 describes the network architectures that we will test self-distillation with. Section 5 describes the experimental setup. Finally, the results of these experiments are discussed in 6.

2. Self-distillation

Methods of training a model based on the imperfect output of another model have been around for more than a decade. McClosky et al. (2006) and Huang and Harper (2009) found improvements in constituency parsing by using model predictions on unlabelled data as additional training data. More recently, van Noord et al. (2018) showed that semantic parsing can be improved by adding the output of a symbolic semantic parser to the training data of a sequence to sequence neural network model. Pütz and Glocker (2019) found that adding the output of a different parser to the training data improves a transition-based semantic parser, especially in a low-resource setting.

Using model outputs as training targets for neural networks became known as *knowledge distillation* (Hinton et al. 2015). It was originally proposed as a model compression technique with the aim of making large models more economical for inference or to enable their use on memory- or compute-constrained devices (Tsai et al. 2019, Sanh et al. 2019, Anderson and Gómez-Rodríguez 2020). Knowledge distillation commonly trains a large-scale teacher model and then distills this teacher model into a smaller student model. Kuncoro et al. (2016) train an ensemble of 20 greedy stack LSTM parsers and distill it into a single graph-based parser using a novel cost formulation.

In parallel, another strain of work explored knowledge distillation as a means to improve model accuracy by distilling the teacher model into a student model of an equal or larger size (Furlanello et al. 2018). Clark et al. (2019) extends this method to a multi-tasking setup, where they distill multiple teacher models into a single student model. These born-again or self-distilled networks often surpass their teacher models in accuracy without access to new information. A theoretical explanation of this method is still lacking, although some advances have been made (Mobahi et al. 2020). In most prior work on self-distillation, the student model is trained on the same training data that is used to train the teacher model. In contrast to such approaches, we explore self-distillation on unlabeled data.

3. Parsing as sequence labeling

3.1 Introduction

In our experiments, we use the *dependency parsing as sequence labeling* scheme proposed by Spoustová and Spousta (2010) and Strzyz et al. (2019). In this scheme, each token is annotated with a complex tag that encodes the position of the token's head and its relation to the head. The position of the head is encoded as the part-of-speech of the head and its relative offset in terms of that part of speech. For example, a token annotated with the tag `nsubj/verb/2` is attached to the second

succeeding *verb* with the *nsubj* relation. Figure 1 shows an example of a dependency tree and how it would be encoded in this tagging scheme.

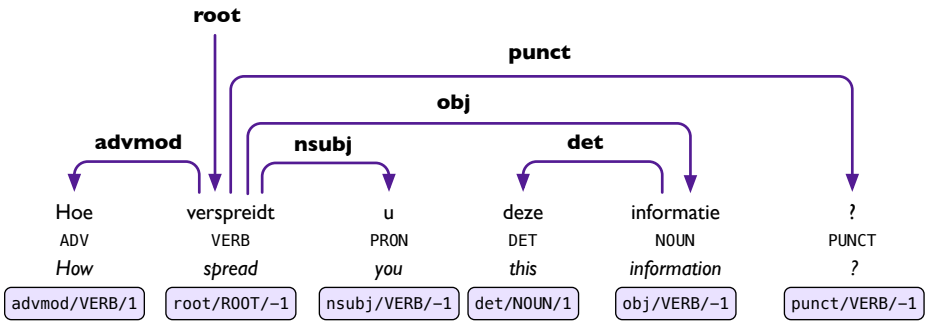

Figure 1: A dependency graph with its encoding as complex tags. Each tag encodes: the relation to the head; the part-of-speech of the head; the relative index of the head in part-of-speech tags.

This dependency labeling scheme has three benefits: (1) it can be used with a standard sequence labeler and does not require a parsing architecture; (2) tagging and decoding of dependency tags can be performed in $\mathcal{O}(n)$ time where n is the sentence length; and (3) the scheme can produce non-projective dependencies. However, the scheme can only encode dependency graphs that obey the single-headedness property.

In the remainder of this section, we discuss issues that arise during decoding. We also discuss the part-of-speech inventory that we use in the dependency labels that were discussed in this section.

3.2 Issues in decoding

During training, an encoder is used to convert gold-standard dependency trees to dependency labels. A standard sequence labeling architecture can then be trained on the pairings of inputs (such as words and part-of-speech tags) and dependency labels. During prediction, the sequence labeler predicts the top-n most probable labels for each token and a decoder constructs dependency edges from these labels.

Decoding can fail in three different ways, which we will address now in more detail:

1. **Failure to decode a tag.** The head position in the assigned dependency label may not be present in the sentence. For example, the label `nsubj/verb/2` requires that there is a second succeeding token tagged as *verb*. However, if no such token exists, no edge can be constructed from this label. In such a case, we attempt to decode the next label from the best-n ($n = 3$) labels. If none of the best-n labels could be decoded, we attach the token to root token of the dependency graph.

2. **Unrooted graph.** If no token acts as the root of the dependency graph, we follow the strategy proposed by Strzyz et al. (2019) and reattach the token with the highest probability to have a root attachment in its best-n labels.[2] If no such token exists, the leftmost token will be chosen to act as the root of the graph.

3. **Cyclic graph.** It is possible that the decoded graph contains a cycle. In this case, we again follow the strategy proposed by Strzyz et al. (2019) and reattach the left-most token in the cycle to the the root token of the dependency graph.

After applying these post-processing steps, parsing will always result in a rooted, fully-spanning tree.

2. Such a token would have the label `root/root/-1`.

3.3 Part-of-speech inventory

Since part-of-speech tags are used as a part of the sequence labels, the granularity of the part-of-speech inventory influences the performance of the model. A large part-of-speech inventory may lead to a combinatory explosion of part-of-speech tags, dependency relations, and relative offsets. Given the relatively modest size of hand-corrected treebanks, this can lead to data sparseness.

A small part-of-speech inventory, on the other hand, is less informative to the parser. For example, if the part-of-speech tag set distinguishes finite verbs, infinitives, participles, and past participles, it may help the parser to disambiguate verbal attachments such as clausal complements.

This trade-off between the granularity of tag sets and parser performance has been observed in prior literature. For instance, Maier et al. (2014) compare three tag sets for German with different grannularity. They observe that using too coarse-grained *or* too fine-grained tag inventories decreases parsing performance. Our preliminary experiments with various tagsets have confirmed these observations.

We will now describe the tag sets that we use for German an Dutch in more detail.

German For German, we use the UD conversion (Çöltekin et al. 2017) of the TüBa-D/Z treebank that is described in more detail in Section 5. This treebank annotates each token with a universal part-of-speech tag (Nivre et al. 2020), an STTS[3] tag (Schiller et al. 1999), and morphological features (Telljohann et al. 2005). Table 2 lists the number of distinct dependency labels that are generated when using universal part-of-speech tags, STTS tags, or concatenations of STTS tags with morphological features. For each tag set, the number of dependency labels with ten or fewer instances is also shown. When moving from universal part-of-speech tags to STTS tags, the number of dependency labels and the number of sparse dependency labels grows relatively modestly compared to the number of additional part-of-speech tags. However, adding morphological features to the STTS tags does increase both the number of part-of-speech tags and the number of dependency labels drastically. In further experiments, we will use the STTS tag set for German, since it provides a good trade-off between grannularity and data availability.

Tag set	**POS tags**	**<= 10**	**Dependency labels**	**<= 10**
Universal Tags	16	0	2,255	1,234
STTS	54	0	3,103	1,785
STTS + morph	562	122	9,688	6,231

Table 2: The number of part-of-speech tags and resulting dependency labels for each part-of-speech inventory of the TüBa-D/Z UD treebank. Also listed is the number of part-of-speech and dependency labels that occur fewer than 10 times in the treebank.

Dutch We conducted our experiments for Dutch on a UD conversion of the Lassy Small treebank (Bouma and van Noord 2017). The treebank is described in more detail in Section 5. The conversion provides universal and D-COI (Van Eynde 2005) part-of-speech tags. Both tag sets are on the extreme ends when it comes to their inventory sizes. We have also generated a custom tag set that was inspired by the STTS tag set by appending certain features of the D-COI tags to the universal tags. The influence of the part-of-speech tag set on the set of dependency labels is shown in Table 3. As for German, we pick the middle ground that the extended universal tag set provides.

4. Sequence labeling architectures

In our experiments, we compare two commonly-used sequence labeling architectures for neural parsing, namely bidirectional LSTMs (Hochreiter and Schmidhuber 1997, Graves and Schmidhuber 2005)

3. Stuttgart-Tübingen Tag Set

Tag set	POS tags	<= 10	Dependency labels	<= 10
Universal Tags	16	0	2,096	1,203
Universal Tags extended	43	1	2,636	1,569
D-COI	214	40	4,650	2,950

Table 3: The number of part-of-speech tags and resulting dependency labels for each part-of-speech inventory of the Lassy Small UD treebank. Also listed is the number of part-of-speech and dependency labels that occur fewer than 10 times in the treebank.

and transformers (Vaswani et al. 2017). Doing so allows us to gauge whether self-distillation works with different architectures. We will first discuss the input and output of the models, before describing the model architectures in more detail.

Input We represent each token as a concatenation of its word embedding and its part-of-speech tag embedding. The word embeddings are trained with the structured skipgram model (Ling et al. 2015). In contrast to the well-known skipgram model (Mikolov et al. 2013), the structured skipgram model takes word order into account while training embeddings. This results in embeddings that are more strongly tailored towards syntax tasks than embeddings trained using the skipgram model. We also use subword embeddings (Bojanowski et al. 2017). Subword embeddings allow the model to learn certain regularities for stems and affixes, as well as making it possible to compute embeddings for unknown words after training.[4] The part-of-speech embeddings are trained using the structured skip-gram model without subword embeddings. The training data and hyperparameters for the embeddings are described in Section 5.

Output Each model predicts dependency labels (Section 3.1) using a softmax classification layer. We also experimented with a conditional random field (CRF) classification layer (Collobert et al. 2011). However, preliminary experiments did not show any improvements over a softmax classification layer.

Bidirectional LSTM The first type of model architecture that we evaluate is Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997). This is a type of recurrent neural network that creates a contextual representation of a token based on the embedding of the token and a hidden state that represents the contextualized representation of the preceding tokens. We use two commonly-used extensions to LSTM. First, we concatenate the word representations from LSTMs that are applied in sentence order and reverse sentence order (Graves and Schmidhuber 2005). As a result of applying such a bidirectional LSTM (BiLSTM), a token's representation contains information from its left and right contexts. Second, we stack several bidirectional LSTMs (Graves and Schmidhuber 2005). This allows the network to make increasingly powerful representations of the input.

For models that use supervised training, we use 2 BiLSTM layers with 200-dimensional representations. For the self-distillation models, we use 3 BiLSTM layers with 400-dimensional representations. We use more shallow networks for purely supervised training, because we found that increasing the number of layers decreased performance, possibly due to difficulties in backpropagation of weight updates. Layer normalization (Ba et al. 2016) is applied after the BiLSTM.

Transformer The second type of architecture that we evaluate is the transformer network (Vaswani et al. 2017). Recurrent neural networks, such as the LSTM, propagate longer-distance information from token to token. In transformers, the representation of a token is directly based on all other tokens using the self-attention mechanism. This mechanism computes per token an attention score for all tokens in a sentence. The token's representation is then based on the representations of all words in the sentence, weighted by the attention scores.

4. The embedings are trained using finalfrontier: `https://github.com/finalfusion/finalfrontier`

We use a standard transformer network as proposed by Vaswani et al. (2017) with 6 layers, 8 attentions heads, and 256 hidden units. We use an inner-layer dimensionality of 3072 in the pointwise feed-forward neural network. Positions are encoded using sinusoidal position encoding.

5. Experimental setup

The main goal of our evaluation is to measure to which degree self-distillation improves parsing accuracy. In order to do so, we need strong baseline models for each neural architecuture (Section 4) to act as a teacher in self-distillation. The training procedure of these baseline models is described in Section 5.1. We will then describe how we perform self-distillation in Section 5.2. Each distilled model is finetuned on the supervised training set, as described in Section 5.3. Finally, we describe the data sets in Section 5.4.

5.1 Baseline model

The transformer and BiLSTM baseline models for German and Dutch are trained on the training partition of a treebank with gold-standard dependency relations and non-gold part-of-speech tags (Section 5.4). A small development set is used to monitor training and refine the hyperparameters.

Both the BiLSTM and the transformer models were optimized using the Adam optimizer (Kingma and Ba 2015). For both types of models we also found it to be highly beneficial to use linear warmup of the learning rate during the initial stages of training to dampen the impact of large gradients on updates (Ma and Yarats 2019). After warmup, we use a learning rate that is scaled if the accuracy does not improve on the development set for a certain number of epochs. Training is ended using early stopping. The other hyperparameters that are used during training of the baseline models can be found in Appendix B.

5.2 Self-distillation

We apply a very simple form of self-distillation, where the student model is trained to predict the highest-probability label assigned to a token by the teacher model. Thus, for each token in the data that is used for self-distillation, we find the label l for token representation x_n,

$$\arg\max_{l} P_{\text{teacher}}(l|x_{...n-1}, x_n, x_{n+1...}) \tag{1}$$

where $x_{...n-1}$ is the left context of x_n and $x_{n+1...}$ the right context of x_n. We then train the student to predict the probability distribution $P_{\text{student}}(l|x_{...n-1}, x_n, x_{n+1...}) = 1$ using the negative log-likelihood as the objective function to be minimized.

We perform self-distillation during two iterations over the unannotated distillation data, after which the network typically converged. As with the baseline models, we use linear warmup of the learning rate. Then the learning rate decays linearly to zero during self-distillation. The hyperparameters used for distillation can be found in Appendix C.

5.3 Finetuning

After self-distillation, we finetune the model on the supervised training set. Finetuning continues training using the final self-distillation model parameters, but then using gold-standard labels rather than the output of the teacher model. We start finetuning of the BiLSTMs at a much lower learning rate than for training the baseline models, since the parameters of the model are already in a good starting state. Besides the lower starting learning rate, optimization is similarly to the baseline models.

We have found that performing another round of self-distillation and finetuning can improve performance of the model even further. From here on, we will refer to the model trained through

one round of self-distillation plus finetuning as a *first generation* model and a model trained through two rounds as a *second generation* model.

5.4 Data sets

German We use the universal dependencies (Çöltekin et al. 2017) conversion of the German TüBa-D/Z treebank (Telljohann et al. 2005) release 11 for training the base models and finetuning the distilled models. This treebank consists of 104,787 sentences and 1,959,474 tokens of newspaper text. In order to train and evaluate the parser with non-gold part-of-speech tags, we obtain non-gold tags through ten-fold jackknifing using a BiLSTM part-of-speech tagger. The data was shuffled for the experiments and then split such that 70% of sentences are used for training, 10% for development, and 20% as held-out data.

Self-distillation is performed on the the tokenized text of the Taz subcorpus of the Tübingen German Dependency Treebank TüBa-D/DP treebank (TüBa-D/DP) (de Kok and Pütz 2019), which consists of 29.9M sentences and 393.7M tokens. We have removed the sentences that are also part of the TüBa-D/Z treebank. The sentences were shuffled before self-distillation.

The word and tag embeddings were trained using `finalfrontier`[5] on the Taz, Wikipedia, and Europarl subcorpora of the TüBa-D/DP (the hyperparameters are summarized in Appendix A).

Dutch The Dutch models were trained and finetuned on the Lassy Small treebank (Van Noord et al. 2013). We converted Lassy Small to Universal Dependencies using the software provided with Bouma and van Noord (2017). After the conversion, the treebank consists of 65,147 sentences and 1,095,087 tokens.[6] We then transform the part-of-speech tags as discussed in Section 3.3. Finally, we perform ten-fold jackknifing to generate non-gold part-of-speech tags and shuffle and split the sentences following the same ratios as German (70%/10%/20% for train/dev/held-out).

Self-distillation is performed on the shuffled sentences of Lassy Large (Van Noord et al. 2013) minus the sentences in Lassy Small, resulting in a corpus which consists of 47.6M sentences and 700M tokens. The word and tag embeddings were also trained on Lassy Large.

6. Results

German Table 4 summarizes the results for German dependency parsing using the LSTM models. For each model, we report the Labeled Attachment Store (LAS) and the Unlabeled Attachment Score (UAS). The UAS reports the percentage of tokens that were annotated with the correct head, while the LAS reports the percentage of tokens that were annotated with the correct head *and* dependency relation.

As discussed in the previous section, the baseline model is trained on the gold-standard data set and acts as a teacher for the first round of self-distillation. Even without finetuning, the first generation distilled model outperforms the baseline model with an improvement of 0.54 in Labeled Attachment Score (LAS). The difference between the baseline model and the distilled model becomes even more profound after finetuning, which increases the improvement over the baseline model by 1.07 LAS. Performing another round of distillation and finetuning using the first-generation distilled model improves the accuracy by another 0.49 in LAS.

The German transformer model (Table 5) performs almost identically to the BiLSTM, showing similar improvements after self-distillation.

Dutch The performance of the Dutch BiLSTM models is shown in Table 6. The accuracy of the Dutch models is quite a bit lower than the German models. We believe that the difference can be attributed both to the smaller size of the treebank (65,147 sentences in Lassy Small, compared to

5. `https://github.com/finalfusion/finalfrontier/`
6. A small number sentences were removed during conversion due to conversion errors.

Model	Generation	LAS	UAS
baseline	0	92.96	94.50
self-distillation	1	93.50	94.92
finetuned	1	94.03	94.92
self-distillation	2	94.29	95.56
finetuned	2	**94.52**	**95.74**

Table 4: Accuracy of the German BiLSTM models. Each round of self-distillation adds marked improvements over the previous model.

Model	Generation	LAS	UAS
baseline	0	92.91	94.41
self-distillation	1	93.62	94.96
finetuned	1	94.06	95.36
self-distillation	2	94.36	95.59
finetuned	2	**94.50**	**95.73**

Table 5: Accuracy of the German transformer models. Each round of self-distillation improves over the previous model.

104,787 sentences in the TüBa-D/Z) and the fact that TüBa-D/Z contains text from a single domain (newspaper text), while Lassy Small contains a mixture of different domains.

Model	Generation	LAS	UAS
baseline	0	90.34	92.85
self-distillation	1	90.75	93.17
finetuned	1	91.43	93.71
self-distillation	2	91.75	93.93
finetuned	2	**91.91**	**94.06**

Table 6: Accuracy of the Dutch BiLSTM models. Each round of self-distillation improves over the previous model.

Despite the lower performance of the Dutch models, the same trends can be observed as in the German experiments. The first generation distilled and finetuned model improves over the baseline with 1.09 LAS and the second generation model improves 0.48 LAS over the first generation model. The improvements are remarkably similar to the improvements provided by self-distillation for German, showing that self-distillation improves parsing performance in a consistent manner between Dutch and German.

Finally, the Dutch transformer models, shown in Table 7, perform substantially worse than the BiLSTM models. For instance, the baseline BiLSTM has an LAS that is 0.49 higher than that of the baseline transformer. Since we did not observe such a difference between the German BiLSTM and transformer models, it is likely that the smaller Dutch training set does not provide enough data to realize the full potential of the deeper transformer network.

Despite posting lower scores, the improvements that self-distillation of the Dutch transformer model provides are similar to all prior models – the first generation of self-distillation and finetuning improves 1.07 in LAS over the baseline and the second generation adds another 0.73 in LAS. Furthermore, the gap between the best BiLSTM model and the best transformer model has narrowed to 0.26 LAS.

Model	Generation	LAS	UAS
baseline	0	89.85	92.41
self-distillation	1	90.53	92.93
finetuned	1	90.92	93.30
self-distillation	2	91.42	93.68
finetuned	2	**91.65**	**93.87**

Table 7: Accuracy of the Dutch transformer models. The transformer models are outperformed by their BiLSTM counterparts. However, each round of self-distillation still improves over the previous model.

Improvements in specific relations As discussed in Section 1, our use of self-distillation is motivated by dependency relations that require selectional preferences to resolve. Since such preferences cannot be acquired broadly from small hand-annotated treebanks, our hope was that distilling a new model on a large, unannotated corpus would improve correct prediction of such relations. Table 8 shows the LAS deltas between the baseline Dutch BiLSTM parser and the distilled BiLSTM parser for the ten dependency relations with the most frequent errors in the baseline parser. There are sizable improvements for subject (1.71 LAS) and direct object attachments (1.43 LAS). Similarly large improvements can be seen for prepositional phrase attachment, which covers a large part of words attached with an *nmod* (1.44 LAS) or *obl* (2.22 LAS) relation. Distillation brings comparable improvements to the German BiLSTM, as shown in Table 9.

Besides the specific cases of subject, object, and prepositional phrase attachment, we see large improvements in accuracy all across the board. In fact, even larger improvements are observed for the *parataxis*, *fixed*, and *conj* relations. From this it seems that the model benefits more generally from seeing a larger variety of words in a larger variety of syntactic constellations.

Furthermore, the improvements are much larger than those reported by Van Noord (2007) and Fischer et al. (2019), which use word-based lexical association scores. This shows that selectional preferences can be captured better by models that use rich, contextualized word representations.

Relation	Baseline LAS	Self-distillation LAS	Δ
parataxis	56.73	61.60	4.87
fixed	74.94	78.50	3.56
conj	77.99	81.53	3.54
obl	84.74	86.96	2.22
nsubj	92.41	94.12	1.71
nmod	84.32	85.76	1.44
obj	89.04	90.47	1.43
advmod	88.81	89.93	1.12
amod	93.49	93.99	0.50
root	95.97	96.07	0.10

Table 8: Improvements in LAS of the second generation distilled and finetuned Dutch BiLSTM model for the relations with the most frequent errors (Table 1a)

.

7. Conclusion

Historically, parsers have only been trained on small hand-annotated data sets. This makes it impossible for models to develop strong attachment preferences for combinations of words that do not occur in the training data. In this paper, we have explored self-distillation on unannotated

Relation	Baseline LAS	Self-distillation LAS	Δ
parataxis	68.93	76.40	7.47
conj	81.46	86.18	4.72
nmod	81.67	84.33	2.66
appos	85.66	88.21	2.55
obl	87.23	89.71	2.48
obj	93.46	95.26	1.80
nsubj	94.56	96.18	1.62
amod	94.99	96.05	1.06
advmod	91.52	92.55	1.03
root	96.75	97.46	0.71

Table 9: Improvements in LAS of the second generation distilled and finetuned German BiLSTM model for the relations with the most frequent errors (Table 1b)

.

data as a means for the model to learn attachment preferences on a wider variety of data. Our experiments show that self-distillation can provide large improvements over strong baseline models (1.57 LAS and 1.59 LAS for Dutch and German respectively).

An open question remains what improvement additional rounds of self-distillation will bring. We have also not explored yet what effect the amount, genre, or quality of unannotated data has on self-distillation.

Another interesting question is whether self-distillation also improves parsing models that start from a pretrained model such as BERT (Devlin et al. 2019) or XLM-RoBERTa (Conneau et al. 2019). We have only conducted some initial experiments to answer this question. These experiments also show marked improvements after self-distillation of XLM-RoBERTa-based models, even though the improvements are smaller than those described in this work.

The software used in the experiments and various distilled models for Dutch and German are available through the `stickeritis` GitHub organization: `https://github.com/stickeritis/`

8. Acknowledgments

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

Anderson, Mark and Carlos Gómez-Rodríguez (2020), Distilling neural networks for greener and faster dependency parsing, *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, Association for Computational Linguistics, Online, pp. 2–13. https://www.aclweb.org/anthology/2020.iwpt-1.2.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016), Layer normalization, *arXiv preprint arXiv:1607.06450*.

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent (2001), A neural probabilistic language model, *in* Leen, T. K., T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information*

Processing Systems 13, MIT Press, pp. 932–938. http://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf.

Boguraev, Branimir Konstatinov (1979), Automatic resolution of linguistic ambiguities, *Technical report*, University of Cambridge, Computer Laboratory.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov (2017), Enriching word vectors with subword information, *Transactions of the Association for Computational Linguistics* **5**, pp. 135–146. https://www.aclweb.org/anthology/Q17-1010.

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)*, Association for Computational Linguistics, Gothenburg, Sweden, pp. 19–26. https://www.aclweb.org/anthology/W17-0403.

Chen, Danqi and Christopher D Manning (2014), A fast and accurate dependency parser using neural networks, *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 740–750.

Clark, Kevin, Minh-Thang Luong, Urvashi Khandelwal, Christopher D Manning, and Quoc Le (2019), BAM! Born-again multi-task networks for natural language understanding, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5931–5937.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa (2011), Natural language processing (almost) from scratch, *Journal of machine learning research* **12** (Aug), pp. 2493–2537.

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)*, Association for Computational Linguistics, 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 (2019), Unsupervised cross-lingual representation learning at scale.

de Kok, Daniël and Sebastian Pütz (2019), TüBa D/DP Stylebook.

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 (2017), Deep biaffine attention for neural dependency parsing, *Proceedings of the 5th International Conference on Learning Representations*, ICLR'17. https://arxiv.org/abs/1611.01734.

Fischer, Patricia, Sebastian Pütz, and Daniël de Kok (2019), Association metrics in neural transition-based dependency parsing, *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, Association for Computational Linguistics, Paris, France, pp. 181–189. https://www.aclweb.org/anthology/W19-7722.

Furlanello, Tommaso, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar (2018), Born again neural networks, *arXiv preprint arXiv:1805.04770*.

Graves, Alex and Jürgen Schmidhuber (2005), Framewise phoneme classification with bidirectional LSTM and other neural network architectures, *Neural networks* **18** (5-6), pp. 602–610, Elsevier.

Hindle, Donald and Mats Rooth (1993), Structural ambiguity and lexical relations, *Computational Linguistics* **19** (1), pp. 103–120, MIT Press, Cambridge, MA, USA. http://dl.acm.org/citation.cfm?id=972450.972456.

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.

Hochreiter, Sepp and Jürgen Schmidhuber (1997), Long short-term memory, *Neural computation* **9** (8), pp. 1735–1780, MIT Press.

Huang, Zhongqiang and Mary Harper (2009), Self-training PCFG grammars with latent annotations across languages, *Proceedings of the 2009 conference on empirical methods in natural language processing: Volume 2-Volume 2*, Association for Computational Linguistics, pp. 832–841.

Johnson, Mark and Stefan Riezler (2000), Exploiting auxiliary distributions in stochastic unification-based grammars, *Proceedings of the 1st North American chapter of the Association for Computational Linguistics Conference*, Association for Computational Linguistics, pp. 154–161.

Katz, Jerrold J and Jerry A Fodor (1963), The structure of a semantic theory, *language* **39** (2), pp. 170–210, JSTOR.

Kingma, Diederik P. and Jimmy Ba (2015), Adam: A method for stochastic optimization, *in* Bengio, Yoshua and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980.

Kiperwasser, Eliyahu and Yoav Goldberg (2015), Semi-supervised dependency parsing using bilexical contextual features from auto-parsed data, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1348–1353.

Kiperwasser, Eliyahu and Yoav Goldberg (2016), Simple and accurate dependency parsing using bidirectional LSTM feature representations, *Transactions of the Association for Computational Linguistics* **4**, pp. 313–327, MIT Press.

Kuncoro, Adhiguna, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith (2016), Distilling an ensemble of greedy dependency parsers into one MST parser, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Austin, Texas, pp. 1744–1753. https://www.aclweb.org/anthology/D16-1180.

Ling, Wang, Chris Dyer, Alan W. Black, and Isabel Trancoso (2015), Two/too simple adaptations of Word2Vec for syntax problems, *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Denver, Colorado, pp. 1299–1304. https://www.aclweb.org/anthology/N15-1142.

Ma, Jerry and Denis Yarats (2019), On the adequacy of untuned warmup for adaptive optimization, *arXiv preprint arXiv:1910.04209*.

Maier, Wolfgang, Sandra Kübler, Daniel Dakota, and Daniel Whyatt (2014), Parsing German: How much morphology do we need?, *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, Dublin City University, Dublin, Ireland, pp. 1–14. https://www.aclweb.org/anthology/W14-6101.

McClosky, David, Eugene Charniak, and Mark Johnson (2006), Effective self-training for parsing, *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Ling uistics*, Association for Computational Linguistics, pp. 152–159.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013), Distributed representations of words and phrases and their compositionality, *Advances in neural information processing systems*, pp. 3111–3119.

Mirroshandel, Seyed Abolghasem, Alexis Nasr, and Joseph Le Roux (2012), Semi-supervised dependency parsing using lexical affinities, *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Jeju Island, Korea, pp. 777–785. http://www.aclweb.org/anthology/P12-1082.

Mobahi, Hossein, Mehrdad Farajtabar, and Peter L Bartlett (2020), Self-distillation amplifies regularization in Hilbert space, *arXiv preprint arXiv:2002.05715*.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman (2020), Universal dependencies v2: An evergrowing multilingual treebank collection, *arXiv preprint arXiv:2004.10643*.

Pantel, Patrick and Dekang Lin (2000), An unsupervised approach to prepositional phrase attachment using contextually similar words, *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Hong Kong, pp. 101–108. http://www.aclweb.org/anthology/P00-1014.

Pütz, Tobias and Kevin Glocker (2019), Tüpa at semeval-2019 Task1: (almost) feature-free semantic parsing, *Proceedings of the 13th International Workshop on Semantic Evaluation*, pp. 113–118.

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf (2019), DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, *arXiv preprint arXiv:1910.01108*.

Schiller, Anne, Simone Teufel, Christine Thielen, and Christine Stöckert (1999), Guidelines für das Tagging deutscher Textcorpora mit STTS, *Technical report*, Technical Report. Institut für maschinelle Sprachverarbeitung, Stuttgart.

Spoustová, Drahomíra and Miroslav Spousta (2010), Dependency parsing as a sequence labeling task, *The Prague Bulletin of Mathematical Linguistics* **94** (1), pp. 7–14, Sciendo.

Strzyz, Michalina, David Vilares, and Carlos Gómez-Rodríguez (2019), Viable dependency parsing as sequence labeling, *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. 717–723. https://www.aclweb.org/anthology/N19-1077.

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*.

Tsai, Henry, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer (2019), Small and practical BERT models for sequence labeling, *arXiv preprint arXiv:1909.00100*.

Van Eynde, Frank (2005), Part of speech tagging en lemmatisering van het D-COI corpus, *Intermediate, projectinternal version*.

Van Noord, Gertjan (2007), Using self-trained bilexical preferences to improve disambiguation accuracy, *Proceedings of the Tenth International Conference on Parsing Technologies*, pp. 1–10.

Van Noord, Gertjan et al. (2006), At last parsing is now operational, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, pp. 20–42.

Van Noord, Gertjan, Gosse Bouma, Frank Van Eynde, Daniel 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.

van Noord, Rik, Lasha Abzianidze, Antonio Toral, and Johan Bos (2018), Exploring neural methods for parsing discourse representation structures, *Transactions of the Association for Computational Linguistics* **6**, pp. 619–633.

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 30*, Curran Associates, Inc., pp. 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Wilks, Yorick (1975), An intelligent analyzer and understander of English, *Communications of the ACM* **18** (5), pp. 264–274, ACM New York, NY, USA.

Appendix A. Embeddings hyperparameters

Hyper parameter	**Value**
Model	Structured skipgram
Minimum character n-gram length	3
Maximum character n-gram length	3
N-gram buckets	2^{21}
Context size	10
Dimensions	300
Minimum token count	15 (Dutch), 30 (German)
Epochs	15

Appendix B. Baseline model hyperparameters

B.1 BiLSTM

Hyper parameter	**Value**
Layers	2
Hidden units	200
Input dropout	0.2
LSTM cell dropout	0.4
Initial learning rate	0.01
Learning rate patience	4
Learning rate scale	0.5
Warmup steps	2000

B.2 Transformer

Hyper parameter	Value
Layers	6
Hidden units	256
Attention heads	8
Pointwise feed-forward units	3072
Input dropout	0.1
Attention dropout	0.2
Hidden dropout	0.2
Pointwise feed-forward dropout	0.3
Activation	ReLU
Position embeddings	sinusoidal
Initial learning rate	$3e^{-4}$
Learning rate patience	4
Learning rate scale	0.5
Warmup steps	200

Appendix C. Self-distillation model hyperparameters

C.1 BiLSTM

Hyper parameter	Value
Layers	3
Hidden units	400
Input dropout	0.2
LSTM cell dropout	0.4
Initial learning rate	$1e^{-3}$
Finetuning learning rate	$1e^{-4}$
Learning rate patience	4
Learning rate scale	0.5
Warmup steps	2000

C.2 Transformer

Hyper parameter	Value
Layers	6
Hidden units	256
Attention heads	8
Pointwise feed-forward units	3072
Input dropout	0.1
Attention dropout	0.2
Hidden dropout	0.2
Pointwise feed-forward dropout	0.3
Activation	ReLU
Position embeddings	sinusoidal
Initial learning rate	$4e^{-3}$
Finetuning learning rate	$5e^{-5}$
Learning rate patience	1
Learning rate scale	0.95
Warmup steps	1000 (distillation), 2000 (finetuning)