Deep Contextualized Self-training for Low Resource Dependency Parsing

Guy Rotman and **Roi Reichart**
Faculty of Industrial Engineering and Management, Technion, IIT
grotman@campus.technion.ac.il
roiri@technion.ac.il

Abstract

Neural dependency parsing has proven very effective, achieving state-of-the-art results on numerous domains and languages. Unfortunately, it requires large amounts of labeled data, that is costly and laborious to create. In this paper we propose a self-training algorithm that alleviates this annotation bottleneck by training a parser on its own output. Our *Deep Contextualized Self-training (DCST)* algorithm utilizes representation models trained on sequence labeling tasks that are derived from the parser's output when applied to unlabeled data, and integrates these models with the base parser through a gating mechanism. We conduct experiments across multiple languages, both in low resource in-domain and in cross-domain setups, and demonstrate that DCST substantially outperforms traditional self-training as well as recent semi-supervised training methods. [1]

1 Introduction

Deep Neural Networks (DNNs) have improved the state-of-the-art in a variety of NLP tasks. These include dependency parsing (Dozat and Manning, 2017), semantic parsing (Hershcovich et al., 2017), named entity recognition (Yadav and Bethard, 2018), POS tagging (Plank and Agić, 2018), and machine translation (Vaswani et al., 2017), among others.

Unfortunately, DNNs rely on in-domain labeled training data, which is costly and laborious to achieve. This annotation bottleneck limits the applicability of NLP technology to a small number of languages and domains. It is hence not a surprise that substantial recent research efforts have been devoted to DNN training based on both labeled and unlabeled data, which is typically widely available (§ 2).

A prominent technique for training machine learning models on labeled and unlabeled data is self-training (Yarowsky, 1995; Abney, 2004). In this technique, after the model is trained on a labeled example set it is applied to another set of unlabeled examples, and the automatically and manually labeled sets are then combined in order to re-train the model – a process that is sometimes performed iteratively. While self-training has shown useful for a variety of NLP tasks, its success for deep learning models has been quite limited (§ 2).

Our goal is to develop a self-training algorithm that can substantially enhance DNN models in cases where labeled training data is scarce. Particularly, we are focusing (§ 5) on the lightly supervised setup where only a small in-domain labeled dataset is available, and on the domain adaptation setup where the labeled dataset may be large but it comes from a different domain than the one to which the model is meant to be applied. Our focus task is dependency parsing, which is essential for many NLP tasks (Levy and Goldberg, 2014; Angeli et al., 2015; Toutanova et al., 2016; Hadiwinoto and Ng, 2017; Marcheggiani et al., 2017), but where self-training has typically failed (§ 2). Moreover, neural dependency parsers (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017) substantially outperform their linear predecessors, which makes the development of self-training methods that can enhance these parsers in low-resource setups a crucial challenge.

We present a novel self-training method, suitable for neural dependency parsing. Our algorithm (§ 4) follows recent work that has demonstrated the power of pre-training for improving DNN models in NLP (Peters et al., 2018; Devlin et al., 2019) and particularly for domain adaptation (Ziser and Reichart, 2018). However, while in previous work a representation model, also known as

[1]Our code is publicly available at https://github.com/rotmanguy/DCST.

a contextualized embedding model, is trained on a language modeling related task, our algorithm utilizes a representation model that is trained on sequence prediction tasks derived from the parser's output. Our representation model and the base parser are integrated into a new model through a gating mechanism, and the resulting parser is then trained on the manually labeled data.

We experiment (§ 6,7) with a large variety of lightly-supervised and domain adaptation dependency parsing setups. For the lightly-supervised case we consider 17 setups: 7 in different English domains and 10 in other languages. For the domain adaptation case we consider 16 setups: 6 in different English domains and 10 in 5 other languages. Our Deep Contextualized Self-training (DCST) algorithm demonstrates substantial performance gains over a variety of baselines, including traditional self-training and the recent cross-view training approach (CVT) (Clark et al., 2018) that was designed for semi-supervised learning with DNNs.

2 Previous Work

Self-training in NLP Self-training has shown useful for various NLP tasks, including word sense disambiguation (Mihalcea, 2004; Yarowsky, 1995), bilingual lexicon induction (Artetxe et al., 2018), neural machine translation (Imamura and Sumita, 2018), semantic parsing (Goldwasser et al., 2011) and sentiment analysis (He and Zhou, 2011). For constituency parsing, self-training has shown to improve linear parsers both when large training data is available (McClosky et al., 2006a,b), and in the lightly supervised and the cross-domain setups (Reichart and Rappoport, 2007). While several authors failed to demonstrate the efficacy of self-training for dependency parsing (e.g. (Rush et al., 2012)), recently it was found useful for neural dependency parsing in fully supervised multilingual settings (Rybak and Wróblewska, 2018).

The impact of self-training on DNNs is less researched compared to the extensive investigation with linear models. Recently, Ruder and Plank (2018) evaluated the impact of self-training and the closely related tri-training method (Zhou and Li, 2005; Søgaard, 2010) on DNNs for part-of-speech (POS) tagging and sentiment analysis. They found self-training to be effective for the sentiment classification task, but it failed to improve their BiLSTM POS tagging architecture. Tri-training has shown effective for both the classification and the sequence tagging task, and in Vinyals et al. (2015) it has shown useful for neural constituency parsing . This is in-line with Steedman et al. (2003) that demonstrated the effectiveness of the closely related co-training method (Blum and Mitchell, 1998) for linear constituency parsers.

Lastly, Clark et al. (2018) presented the Cross-view Training (CVT) algorithm, a variant of self-training that employs unsupervised representation learning. CVT differs from classical self-training in the way it exploits the unlabeled data: it trains auxiliary models on restricted views of the input to match the predictions of the full model that observes the whole input.

We propose a self-training algorithm based on deep contextualized embeddings, where the embedding model is trained on sequence tagging tasks that are derived from the parser's output on unlabeled data. In extensive lightly supervised and cross-domain experiments with a neural dependency parser, we show that our DCST algorithm outperforms traditional self-training and CVT.

Pre-training and Deep Contextualized Embedding Our DCST algorithm is related to recent work on DNN pre-training. In this line, a DNN is first trained on large amounts of unlabeled data and is then used as the word embedding layer of a more complex model that is trained on labeled data to perform an NLP task. Typically, only the upper, task specific, layers of the final model are trained on the labeled data, while the parameters of the pre-trained embedding network are kept fixed.

The most common pre-training task is language modeling or a closely related variant (McCann et al., 2017; Peters et al., 2018; Devlin et al., 2019; Ziser and Reichart, 2018). The outputs of the pre-trained DNN are often referred to as contextualized word embeddings, as these DNNs typically generate a vector embedding for each input word, which takes its context into account. Pre-training has led to performance gains in many NLP tasks.

Recently, Che et al. (2018) incorporated ELMo embeddings (Peters et al., 2018) into a neural dependency parser and reported improvements over a range of Universal Dependency (UD) (McDonald et al., 2013; Nivre et al., 2016, 2018) languages in the fully supervised setup. In this paper we focus on the lightly supervised and domain adaptation setups, trying to compensate for the lack of labeled data by exploiting automatically labeled trees generated by the base parser for unlabeled sentences.

Our main experiments (§7) are with models that utilize non-contextualized word embeddings. We believe this is a more practical setup when considering multiple languages and domains. Indeed, Che et al. (2018), who trained their ELMo model on the unlabeled data of the CoNLL 2018 shared task, reported that "The training of ELMo on one language takes roughly 3 days on an NVIDIA P100 GPU.". However, we also demonstrate the power of our models when ELMo embeddings are available (§8), in order to establish the added impact of deep contextualized self-training on top of contextualized word embeddings.

Lightly Supervised Learning and Domain Adaptation for Dependency Parsing Finally, we briefly survey earlier attempts to learn parsers in setups where labeled data from the domain to which the parser is meant to be applied is scarce. We exclude from this brief survey literature that has already been mentioned above.

Some notable attempts are: exploiting short dependencies in the parser's output when applied to large target domain unlabeled data (Chen et al., 2008), adding inter-sentence consistency constraints at test time (Rush et al., 2012), selecting effective training domains (Plank and Van Noord, 2011), exploiting parsers trained on different domains through a mixture of experts (McClosky et al., 2010), embedding features in a vector space (Chen et al., 2014), and Bayesian averaging of a range of parser parameters (Shareghi et al., 2019).

Recently, Sato et al. (2017) presented an adversarial model for cross-domain dependency parsing in which the encoders of the source and the target domains are integrated through a gating mechanism. Their approach requires target domain labeled data for parser training and hence it cannot be applied in the unsupervised domain adaptation setup we explore (§ 5). We adopt their gating mechanism to our model and extend it to integrate more than two encoders into a final model.

3 Background: The BiAFFINE Parser

The parser we utilize in our experiments is the BiAFFINE parser (Dozat and Manning, 2017). Since the structure of the parser affects our DCST algorithm, we briefly describe it here.

A sketch of the parser architecture is provided in Figure 1. The input to the parser is a sentence $(x_1, x_2, \ldots, x_m)$ of length m. An embedding layer embeds the words into fixed-size vectors $(w_1, w_2, \ldots, w_m)$. Additionally, character-level embeddings c_t^k retrieved from a CNN (Zhang et al., 2015), and a POS embedding p_t, are concatenated to each word vector. At time t, the final input vector $f_t = [w_t; c_t; p_t]$ is then fed into a BiLSTM encoder E_{parser} which outputs a hidden representation h_t:

$$h_t = E_{parser}(f_t). \tag{1}$$

Given the hidden representations of the i'th word h_i and the j'th word h_j , the decoder outputs a score $s_{i,j}$, indicating the model belief that the latter should be the head of the former in the dependency tree. More formally,

$$s_{i,j} = r_i^T U r_j + w_j^T r_j, \tag{2}$$

where $r_i = MLP(h_i)$, and U and w_j are learned parameters (MLP is a multi-layered perceptron).

Similarly, a score $l_{i,j,k}$ is calculated for the k'th possible dependency label of the arc (i, j):

$$l_{i,j,k} = q_i^T U_k^{'} q_j + w_k^{'T}[q_i; q_j] + \mathrm{b}_k^{'}, \tag{3}$$

where $q_i = MLP^{'}(h_i)$, and $U_k^{'}$, $w_k^{'}$, and $\mathrm{b}_k^{'}$ are learned parameters. During training the model aims to maximize the probability of the gold tree:

$$\sum_{i=1}^{m} p(y_i|x_i, \theta) + p(y_i'|x_i, y_i, \theta), \tag{4}$$

where y_i is the head of x_i, y_i' is the label of the arc (x_i, y_i), θ represents the model's parameters, $p(y_i|x_i, \theta) \propto exp(s_{x_i,y_i})$, and $p(y_i'|x_i, y_i, \theta) \propto exp(l_{x_i,y_i,y_i'})$. At test time, the parser runs the MST algorithm (Edmonds, 1967) on the arc scores in order to generate a valid tree.

4 Deep Contextualized Self-training

In this section we present our DCST algorithm for dependency parsing (Algorithm 1). As a semi-supervised learning algorithm, DCST assumes a labeled dataset $\mathbf{L} = \{(x_i^l, y_i^l)\}_{i=1}^{|\mathbf{L}|}$, consisting of sentences and their gold dependency trees, and an unlabeled dataset $\mathbf{U} = \{x_i^u\}_{i=1}^{|\mathbf{U}|}$, consisting of sentences only.

We start (Algorithm 1, step 1) by training the base parser (the BiAFFINE parser in our case) on

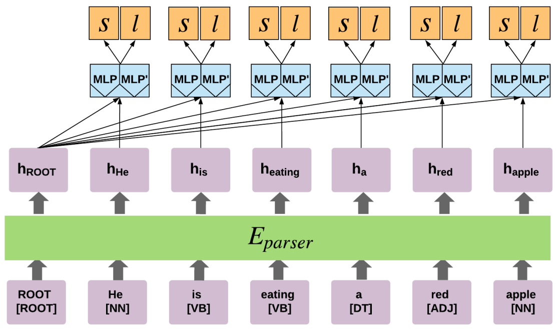

Figure 1: The BiAFFINE parser.

Algorithm 1 Deep Contextualized Self-training (DCST)

```
Input: Labeled data L, Unlabeled data U
Algorithm:
  1. Train the base parser on L (§ 3).
  2. Parse the sentences of U with the base parser.
  3. Transform the automatically parsed trees of U to one
     or more word-level tagging schemes (§ 4.1).
  4. Train (a) contextualized embedding model(s) to predict
     the word-level tagging(s) of U (§ 4.1).
  5. Integrate the representation model(s) of step 4 with the
     base parser, and train the resulting parser on L (§ 4.2).
```

the labeled dataset **L**. Once trained, the base parser can output a dependency tree for each of the unlabeled sentences in **U** (step 2). We then transform the automatic dependency trees generated for **U** into one or more word-level tagging schemes (step 3). In § 4.1 we elaborate on this step. Then, we train a BiLSTM sequence tagger to predict the word-level tags of **U** (step 4). If the automatic parse trees are transformed to more than one tagging scheme, we train multiple BiLTMs – one for each scheme. Finally, we construct a new parser by integrating the base parser with the representation BiLSTM(s), and train the final parser on the labeled dataset **L** (step 5). In this stage, the base parser parameters are randomly initialized, while the parameters of the representation BiLSTM(s) are initialized to those learned in step 4.

We next discuss the three word-level tagging schemes derived from the dependency trees (step 3), and then the gating mechanism employed in order to compose the hybrid parser (step 5).

4.1 Representation Learning (Steps 3 and 4)

In what follows we present the three word-level tagging schemes we consider at step 3 of the DCST algorithm. Transferring the parse trees into tagging schemes is the key for populating information from the original (base) parser on unlabeled data, in a way that can later be re-encoded to the parser through its word embedding layers. The key challenge we face when implementing this idea is the transformation of dependency trees into word level tags that preserve important aspects of the information encoded in the trees.

We consider tagging schemes that maintain various aspects of the structural information encoded in the tree. Particularly, we start from two tagging schemes that even if fully predicted still leave ambiguity about the actual parse tree: the number of direct dependants each word has and the distance of each word from the root of the tree. We then consider a tagging scheme, referred to as the Relative POS-based scheme, from which the dependency tree can be fully reconstructed. While other tagging schemes can definitely be proposed, we believe that the ones we consider here span a range of possibilities that allows us to explore the validity of our DCST framework.

More specifically, the tagging schemes we consider are defined as follows:

Number of Children Each word is tagged with the number of its children in the dependency tree. We consider only direct children, rather than other descendants, which is equivalent to counting the number of outgoing edges of the word in the tree.

Distance from the Root Each word is tagged with its minimal distance from the root of the tree. For example, if the arc $(ROOT, j)$ is included in the tree, the distance of the j'th word from the ROOT is 1. Likewise, if $(ROOT, j)$ is not included but $(ROOT,i)$ and (i,j) are, then j'th distance is 2.

Relative POS-based Encoding Each word is tagged with its head word according to the relative POS-based scheme (Spoustová and Spousta, 2010; Strzyz et al., 2019) The head of a word is encoded by a pair $(p, e) \in P \times [-m + 1, m - 1]$, where P is the set of all possible parts of speech and m is the sentence length. For a positive (negative) number e and a POS p, the pair indicates that the head of the represented word is the e'th word to its right (left) with the POS tag p. To avoid sparsity we coarsen the POS tags related to nouns, proper names, verbs, adjectives, punctuation-marks and brackets into one tag per category.

Although this word-level tagging scheme was introduced as means of formulating dependency

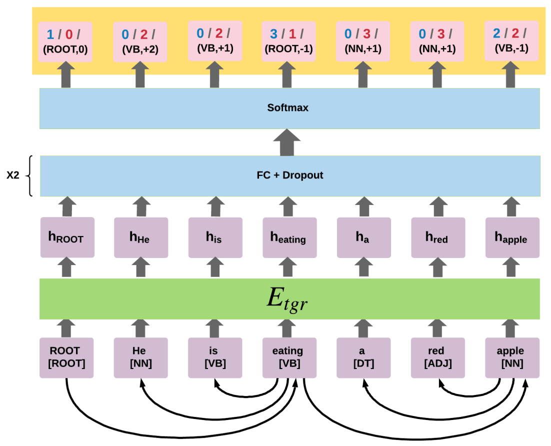

Figure 2: The sequence tagger applied to automatically parsed sentences in **U** (Algorithm 1, step 4). The tagger predicts for each word its label according to one of the three tagging schemes: Number of Children (blue), Distance from the Root (red), and Relative POS-based Encoding (black). The curved arrows sketch the gold dependency tree from which the word-level tags are derived.

parsing as a sequence tagging task, in practice sequence models trained on this scheme are not competitive with state-of-the-art parsers and often generate invalid tree structures (Strzyz et al., 2019). Here we investigate the power of this scheme as part of a self-training algorithm.

The Sequence Tagger Our goal is to encode the information in the automatically parsed trees into a model that can be integrated with the parser at later stages. This is why we choose to transform the parse trees into word-level tagging schemes that can be learned accurately and efficiently by a sequence tagger. Note that efficiency plays a key role in the lightly-supervised and domain adaptation setups we consider, as large amounts of unlabeled data should compensate for the lack of labeled training data from the target domain.

We hence choose a simple sequence tagging architecture, depicted in Figure 2. The encoder E_{tgr} is a BiLSTM, similarly to E_{parser} of the parser. The decoder is composed of two fully connected (FC) layers with dropout (Srivastava et al., 2014) and an exponential linear unit (ELU) activation function (Clevert et al., 2016), followed by a final softmax layer that outputs the tag probabilities.

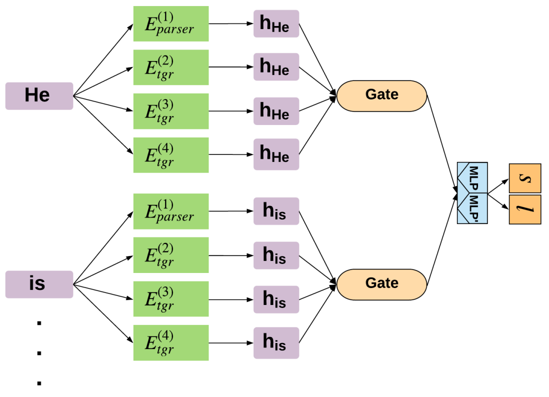

Figure 3: An illustration of the hybrid parser with three auxiliary sequence taggers. An input word vector is passed through the parser encoder ($E^{(1)}_{parser}$) and the three pre-trained tagger encoders ($E^{(2)}_{tgr} - E^{(4)}_{tgr}$). The gating mechanism (Gate) computes a weighted average of the hidden vectors. Finally, the output of the gating mechanism is passed to the BIAFFINE decoder to predict the arc and label scores for each word pair.

4.2 The Final Hybrid Parser (Step 5)

In step 5, the final step of Algorithm 1, we integrate the BiLSTM of the sequence tagger, which encodes the information in the automatically generated dependency trees, with the base parser. Importantly, when doing so we initialize the BiLSTM weights to those to which it converged at step 4. The parameters of the base (BIAFFINE) parser, in contrast, are randomly initialized. The resulting hybrid parser is then trained on the labeled data in **L**. This way, the final model integrates the information from both **L** and the automatic tagging of **U**, generated in step 2 and 3.

We next describe how the encoders of the sequence tagger and the BIAFFINE parser, E_{tgr} and E_{parser}, are integrated through a gating mechanism, similar to that of Sato et al. (2017).

The Gating Mechanism Given an input word vector f_t (§ 3), the gating mechanism learns to scale between the BiLSTM encoder of the parser to that of the sequence tagger (Figure 3):

$$a_t = \sigma(W_g[E_{parser}(f_t); E_{tgr}(f_t)] + b_g),$$
$$g_t = a_t \odot E_{parser}(f_t) + (1 - a_t) \odot E_{tgr}(f_t).$$

where $\odot$ is the element-wise product, σ is the sigmoid function, and W_g and b_g are the gating mechanism parameters. The combined vector g_t is then fed to the parser's decoder.

Extension to $n \geq 2$ Sequence Taggers We can naturally extend our hybrid parser to support n auxiliary taggers (see again Figure 3). Given n taggers trained on n different tagging schemes, the gating mechanism is modified to be:

$$s_t^{(i)} = W_g^{(i)}[E_{parser}^{(1)}(f_t); E_{tgr}^{(2)}(f_t); \ldots; E_{tgr}^{(n+1)}(f_t)] + b_g^{(i)},$$

$$a_t^{(i)} = \frac{exp(s_t^{(i)})}{\sum_{j=1}^{n+1} exp(s_t^{(j)})},$$

$$g_t = a_t^{(1)} \odot E_{parser}^{(1)}(f_t) + \sum_{i=2}^{n+1} a_t^{(i)} \odot E_{tgr}^{(i)}(f_t).$$

This extension provides a richer representation of the automatic tree structures, as every tagging scheme captures a different aspect of the trees. Indeed, in most of our experiments, when integrating the base parser with our three proposed schemes, the resulting model was superior to models that consider a single tagging scheme.

5 Evaluation Setups

This paper focuses on exploiting unlabeled data in order to improve the accuracy of a supervised parser. We expect this approach to be most useful when the parser does not have sufficient labeled data for training, or when the labeled training data do not come from the same distribution as the test data. We hence focus on two setups:

The Lightly Supervised In-domain Setup In this setup we are given a small labeled dataset $\mathbf{L} = \{(x_i^l, y_i^l)\}_{i=1}^{|\mathbf{L}|}$ of sentences and their gold dependency trees and a large unlabeled dataset $\mathbf{U} = \{(x_i^u)\}_{i=1}^{|\mathbf{U}|}$ of sentences coming from the same domain, where $|\mathbf{L}| \ll |\mathbf{U}|$. Our goal is to parse sentences from the domain of $\mathbf{L}$ and $\mathbf{U}$.

The Unsupervised Domain Adaptation Setup In this setup we are given a labeled source domain dataset $\mathbf{L} = \{(x_i^l, y_i^l)\}_{i=1}^{|\mathbf{L}|}$ of sentences and their gold dependency trees, and an unlabeled dataset $\mathbf{U} = \{(x_i^u)\}_{i=1}^{|\mathbf{U}|}$ of sentences from a different target domain. Unlike the lightly-supervised setup, here $\mathbf{L}$ may be large enough to train a high quality parser as long as the training and test sets come from the same domain. However, our goal here is to parse sentences from the target domain.

6 Experiments

We experiment with the task of dependency parsing, in two setups: (a) lightly supervised in-domain and (b) unsupervised domain adaptation.

Data We consider two datasets: **(a)** The English OntoNotes 5.0 (Hovy et al., 2006) corpus. This corpus consists of text from 7 domains: broadcast conversation (bc: 11877 training, 2115 development and 2209 test sentences), broadcast news (bn: 10681, 1293, 1355), magazine (mz: 6771, 640, 778), news (nw: 34967, 5894, 2325), bible (pt: 21518, 1778, 1867), telephone conversation (tc: 12889, 1632, 1364) and web (wb: 15639, 2264, 1683).[2] The corpus is annotated with constituency parse trees and POS tags, as well as other labels that we do not use in our experiments. The constituency trees were converted to dependency trees using the Elitcloud conversion tool.[3] In the lightly supervised setup we experiment with each domain separately. We further utilize this corpus in our domain adaptation experiments. **(b)** The Universal Dependencies (UD) dataset (McDonald et al., 2013; Nivre et al., 2016, 2018). This corpus contains more than 100 corpora of over 70 languages, annotated with dependency trees and universal POS tags. For the lightly supervised setup we chose 10 low-resource languages that have no more than 10K training sentences: Old Church Slavonic (cu), Danish (da), Persian (fa), Indonesian (id), Latvian (lv), Slovenian (sl), Swedish (sv), Turkish (tr), Urdu (ur) and Vietnamese (vi), and performed mono-lingual experiments with each.[4] For the domain adaptation setup we experiment with 5 languages, considering two corpora from different domains for each: Czech (cs_fictree: fiction, cs_pdt: news and science), Galician (gl_ctg: science and legal, gl_treegal: news), Italian (it_isdt: legal, news and wiki, it_postwita: social media), Romanian (ro_nonstandard: poetry and bible, ro_rrt: news, literature, science, legal and wiki) and Swedish (sv_lines: literature and politics, sv_talbanken: news and textbooks).

[2] We removed wb test set sentences where all words are POS tagged with "XX".

[3] `https://github.com/elitcloud/elit-java`.

[4] In case a language has multiple corpora, our training, development and test sets are concatenations of the corresponding sets in these corpora.

Training Setups For the lightly supervised setup we performed experiments with the 7 OntoNotes domains and the 10 UD corpora, for a total of 17 in-domain setups. For each setup we consider three settings that differ from each other in the size of the randomly selected labeled training and development sets: 100, 500 or 1000.[5] We use the original test sets for evaluation, and the remaining training and development sentences as unlabeled data.

For the English unsupervised domain adaptation setup, we consider the news (nw) section of OntoNotes 5.0 as the source domain, and the remaining sections as the target domains. The nw training and development sets are used for the training and development of the parser, while the unlabeled versions of the target domain training and development sets are used for training and development of the representation model. The final model is evaluated on the target domain test set.

Similarly, for unsupervised domain adaptation with the UD languages, we consider within each language one corpus as the source domain and the other as the target domain, and apply the same train/development/test splits as above. For each language we run two experiments, differing in which of the two corpora is considered the source and which is considered the target.

For all domain adaptation experiments, when training the final hybrid parser (Figure 3) we sometimes found it useful to keep the parameters of the BiLSTM tagger(s) fixed in order to avoid an overfitting of the final parser to the source domain. We treat the decision of whether or not to keep the parameters of the tagger(s) fixed as a hyper-parameter of the DCST models and tune it on the development data.

We measure parsing accuracy with the standard Unlabeled and Labeled Attachment Scores (UAS and LAS), and measure statistical significance with the t-test (following Dror et al. (2018)).

Models and Baselines We consider four variants of our DCST algorithm, differing on the word tagging scheme on which the BiLSTM of step 4 is trained (§ 4.1): **DCST-NC**: with the Number of Children scheme, **DCST-DR**: with the Distance from the Root scheme, **DCST-RPE**: with the Relative POS-based Encoding scheme and **DCST-ENS** where the parser is integrated with three BiLSTMs, one for each scheme (where ENS stands for ensemble) (§ 4.2).

To put the results of our DCST algorithm in context, we compare its performance to the following baselines. **Base**: the BiAFFINE parser (§ 3), trained on the labeled training data. **Base-FS**: the BiAFFINE parser (§ 3), trained on all the labeled data available in the full training set of the corpus. In the domain adaptation setups Base-FS is trained on the entire training set of the target domain. This baseline can be thought of as an upper bound on the results of a lightly-supervised learning or domain-adaptation method. **Base + Random Gating (RG)**: a randomly initialized BiLSTM is integrated to the BiAFFINE parser through the gating mechanism, and the resulting model is trained on the labeled training data. We compare to this baseline in order to quantify the effect of the added parameters of the BiLSTM and the gating mechanism, when this mechanism does not inject any information from unlabeled data. **Self-training**: the traditional self-training procedure. We first train the Base parser on the labeled training data, then use the trained parser to parse the unlabeled data, and finally re-train the Base parser on both the manual and automatic trees.

We would also like to test the value of training a representation model to predict the dependency labeling schemes of § 4.1, in comparison to the now standard pre-training with a language modeling objective. Hence, we experiment with a variant of DCST where the BiLSTM of step 4 is trained as a language model (**DCST-LM**). Finally, we compare to the cross-view training algorithm (**CVT**) (Clark et al., 2018), that was developed for semi-supervised learning with DNNs. [6]

Hyper-parameters We employ the BiAFFINE parser implementation of Ma et al. (2018). [7] We consider the following hyper-parameters for the parser and the sequence tagger: 100 epochs with an early stopping criterion according to the development set, the ADAM optimizer (Kingma and Ba, 2015), a batch size of 16, a learning rate of 0.002 and dropout probabilities of 0.33.

The 3-layer stacked BiLSTMs of the parser and the sequence tagger generate hidden representations of size 1024. The fully connected layers of

[5]In languages where the development set was smaller than 1000 sentences we used the entire development set.

[6]`https://github.com/tensorflow/models/tree/master/research/cvt_text`.

[7]`https://github.com/XuezheMax/NeuroNLP2`.

the tagger are of size 128 (first layer) and 64 (second layer). All other parser hyper-parameters are identical to those of the original implementation.

We employ 300-dimensional pre-trained word embeddings: GloVe (Pennington et al., 2014) [8] for English and FastText (Grave et al., 2018) [9] for the UD languages. Character and POS embeddings are 100-dimensional and are initialized to random normal vectors. **CVT** is run for 15K gradient update steps.

7 Results

Table 1 presents the lightly supervised OntoNotes results when training with 500 labeled sentences, while Table 2 presents the UD results in the same setup. Tables 3 and 4 report domain adaptation results for the 6 OntoNotes and 10 UD target domains, respectively. Underscored results are significant compared to the highest scoring baseline, based on t-test with $p < 0.05$.[10]

DCST with Syntactic Self-training DCST-ENS, our model that integrates all three syntactically self-trained BiLSTMs, is clearly the best model. In the lightly supervised setup, it performs best on 5 of 7 OntoNotes domains and on 8 of 10 UD corpora (with the UAS measure). In the cases where DCST-ENS is not the best performing model, it is the second or third best model. In the English and multilingual domain adaptation setups, DCST-ENS is clearly the best performing model, where in only 2 multilingual target domains it is second.

Moreover, DCST-NC, DCST-DR and DCST-RPE, that consider only one syntactic scheme, also excel in the lightly supervised setup. They outperform all the baselines (models presented above the top separating lines in the tables) in the UD experiments, and DCST-RPE and DCST-DR outperform all the baselines in 5 of 7 Ontonotes domains (with the LAS measure). In the domain adaptation setup, however, they are on par with the strongest baselines, which indicates the importance of exploiting the information in all three schemes in this setup (results are not shown in Tables 3 and 4 in order to save space).

Note, that with few exceptions, DCST-NC is the least effective method among the syntactically self-trained DCST alternatives. This indicates that encoding the number of children each word has in the dependency tree is not a sufficiently informative view of the tree.

Comparison to Baselines The CVT algorithm performs quite well in the English OntoNotes lightly supervised setup – it is the best performing model on two domains (nw and pt) and the best baseline for three other domains when considering the UAS measure (bc, bn and tc). However, its performance substantially degrades in domain adaptation. Particularly, in 5 out of 6 OntoNotes setups and in 9 out of 10 UD setups it is the worst performing model. Moreover, CVT is the worst performing model in the lightly supervised multilingual setup.

Overall, this recently proposed model that demonstrated strong results across several NLP tasks, does not rival our DCST models with syntactic self-training in our experimental tasks. Notice that Clark et al. (2018) did not experiment in domain adaptation setups and did not consider languages other than English. Our results suggest that in these cases DCST with syntactic self-training is a better alternative.

We next evaluate the impact of the different components of our model. First, comparison with DCST-LM – the version of our model where the syntactically self-trained BiLSTM is replaced with a BiLSTM trained on the same unlabeled data but with a language modeling objective, allows us to evaluate the importance of the self-generated syntactic signal. The results are conclusive: in all our four setups - English and multilingual lightly-supervised, and English and multilingual domain adaptation, DCST-LM is outperformed by DCST-ENS that considers all three self-trained BiLSTMs. DCST-LM is also consistently outperformed by DCST-RPE, DCST-DR and DCST-NC that consider only one syntactic annotation scheme, except from a few English lightly-supervised cases where it outperforms DCST-NC by a very small margin. Syntactic self-supervision hence provides better means of exploiting the unlabeled data, compered to the standard language modeling alternative.

Another question is whether the BiLSTM models should be trained at all. Indeed, in recent papers untrained LSTMs with random weights

[8] `http://nlp.stanford.edu/data/glove.840B.300d.zip`.

[9] `https://fasttext.cc/docs/en/crawl-vectors.html`.

[10] For this comparison, Base-FS is not considered a baseline, but an upper bound.

	bc		bn		mz		nw		pt		tc		wb	
Model	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Base	74.54	70.77	80.57	77.63	81.47	78.41	80.40	77.56	86.95	83.86	72.15	68.34	78.74	73.24
Base+RG	77.10	73.45	81.90	79.06	83.02	80.29	81.80	79.24	88.13	85.42	73.87	69.97	78.93	75.37
DCST-LM	75.94	72.33	80.01	76.96	82.50	79.53	80.33	77.57	87.53	84.56	72.16	68.30	77.09	73.49
Self-Training	74.64	71.18	82.35	79.75	83.44	80.86	81.93	79.43	87.50	84.52	69.70	66.62	79.18	75.86
CVT	78.47	73.54	82.76	78.19	82.90	78.56	**85.55**	**82.30**	**90.36**	**87.05**	75.36	69.96	78.03	73.10
DCST-NC	78.21	74.62	82.32	79.52	83.52	80.61	81.95	79.17	88.83	85.62	75.35	71.05	78.76	75.10
DCST-DR	78.61	74.80	83.32	80.26	84.27	81.15	82.67	79.74	88.90	85.66	75.05	70.82	79.80	76.12
DCST-RPE	78.70	75.11	83.07	80.41	84.16	81.62	83.02	80.45	88.95	85.96	75.35	71.06	80.25	76.91
DCST-ENS	**78.95**	**75.43**	**83.52**	**80.93**	**84.67**	**81.99**	82.89	80.41	89.38	86.47	**76.47**	**72.54**	**80.52**	**77.32**
Base-FS	86.23	84.49	89.41	88.17	89.19	87.80	89.29	88.01	94.08	92.83	77.12	75.36	87.23	85.56

Table 1: Lightly supervised OntoNotes results with 500 training sentences. Base-FS is an upper bound.

	cu		da		fa		id		lv		sl		sv		tr		ur		vi	
Model	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Base	75.87	67.25	78.13	74.16	82.54	78.59	72.57	57.25	72.81	65.66	76.00	69.28	78.58	72.78	56.07	39.37	84.49	78.10	67.18	62.51
Base+RG	77.98	69.01	80.21	76.11	84.74	80.83	73.18	57.56	74.51	67.60	78.18	71.27	79.90	73.70	58.42	40.32	86.18	79.65	68.75	64.64
DCST-LM	77.67	68.90	80.23	76.06	83.92	79.89	72.61	57.36	73.89	66.59	76.90	70.12	78.73	72.51	57.33	39.27	85.78	79.27	69.11	65.09
Self-Training	75.19	68.07	79.76	75.92	85.04	81.05	74.07	58.73	74.79	68.22	77.71	71.33	79.72	74.12	57.34	40.06	85.63	79.51	68.24	63.96
CVT	61.57	45.60	72.77	66.93	81.08	74.32	72.51	54.94	68.90	57.36	67.89	59.79	77.08	69.60	53.17	32.95	81.49	72.72	60.84	50.98
DCST-NC	78.85	69.75	81.23	76.70	85.94	81.85	74.18	58.63	76.19	68.73	79.26	72.72	81.05	75.09	58.17	39.95	86.17	79.91	69.93	65.91
DCST-DR	79.31	70.20	81.30	76.81	86.20	82.14	**74.56**	58.92	76.99	69.24	80.34	73.35	81.40	75.41	58.30	40.25	86.19	79.68	69.46	65.65
DCST-RPE	**80.57**	**71.83**	81.48	77.45	86.82	82.69	**74.56**	**59.19**	77.45	**70.38**	80.45	74.13	81.95	75.98	59.49	41.45	86.86	**80.92**	70.23	66.26
DCST-ENS	80.55	71.79	**82.07**	**78.04**	**87.02**	**83.13**	74.47	59.13	**77.63**	70.36	**80.68**	**74.32**	**82.40**	**76.61**	**59.60**	**41.72**	**86.96**	80.85	**70.37**	**66.88**
Base-FS	86.13	81.46	85.55	82.93	91.06	88.12	77.42	62.31	85.02	81.59	86.04	82.22	85.18	81.36	62.21	46.23	89.84	85.12	73.26	69.69

Table 2: Lightly supervised UD results with 500 training sentences. Base-FS is an upper bound.

	bc	bn	mz	pt	tc	wb
Model	LAS	LAS	LAS	LAS	LAS	LAS
Base	81.60	85.17	85.48	87.70	75.46	83.85
Base+RG	82.51	85.36	85.77	88.34	75.68	84.34
DCST-LM	82.48	85.77	86.28	89.28	75.72	84.34
Self-Training	80.61	84.52	85.38	87.69	73.62	82.82
CVT	74.81	84.90	84.49	85.71	72.10	82.31
DCST-ENS	**85.96**	**88.02**	**88.55**	**91.62**	**79.97**	**87.38**
Base-FS	84.49	88.17	87.80	92.83	75.36	85.56

Table 3: Unsupervised Domain adaptation OntoNotes results. Base-FS is an upper bound.

substantially enhanced model performance (Wang et al., 2019; Zhang and Bowman, 2018; Tenney et al., 2019; Wieting and Kiela, 2019).

Our results lead to two conclusions. Firstly, Base+RG, the model that is identical to the syntactically trained DCST except that the BiAFFINE parser is integrated with a randomly initialized BiLSTM through our gating mechanism, is consistently outperformed by all our syntactically self-trained DCST models, with very few exceptions. Secondly, in line with the conclusions of the aforementioned papers, Base+RG is one of the strongest baselines in our experiments. Perhaps most importantly, in most experiments this model outperforms the Base parser – indicating the positive impact of the randomly initialized representation models. Moreover, it is the strongest baseline in 2 English domain adaptation setups and in 5 of 10 languages in the lightly-supervised multilingual experiments (considering the UAS measure), and is the second-best baseline in 5 out of 7 English lightly-supervised setups (again considering the UAS measure). The growing evidence for the positive impact of such randomly initialized models should motivate further investigation of the mechanism that underlies their success.

Finally, our results demonstrate the limited power of traditional self-training: In English domain adaptation it harms or does not improve the Base parser; in multilingual domain adaptation it is the best model in 2 cases; and it is the best baseline in 2 of the 7 English lightly-supervised setups and in 3 of the 10 multilingual lightly-supervised setups. This supports our motivation to propose an improved self-training framework.

8 Ablation Analysis and Discussion

Impact of Training set Size Figure 4 presents the impact of the DCST-ENS method on the BiAFFINE parser, in the seven lightly-supervised English setups, as a function of the labeled training set size of the parser. Clearly, the positive impact is substantially stronger for smaller training sets. Particularly, when the parser is trained

	cs_fictree	cs_pdt	gl_ctg	gl_treegal	it_isdt	it_postwita	ro_nonstandard	ro_rrt	sv_lines	sv_talbanken
Model	LAS	LAS	LAS	LAS	LAS	LAS	LAS	LAS	LAS	LAS
Base	69.92	81.83	59.05	60.31	67.82	80.72	65.03	62.75	77.08	77.93
Base+RG	73.12	80.86	58.97	60.52	67.54	80.36	65.93	61.50	77.58	78.04
DCST-LM	73.59	83.33	59.41	60.54	67.52	80.95	65.19	62.46	77.40	77.62
Self-Training	69.50	81.53	59.67	**61.41**	68.02	82.01	66.47	**63.84**	77.60	77.64
CVT	59.77	81.53	51.12	50.31	58.60	70.07	50.82	45.15	45.25	62.87
DCST-ENS	**75.28**	**86.50**	**59.75**	60.98	**69.13**	**83.06**	**67.65**	63.46	**77.86**	**78.97**
Base-FS	84.46	83.70	84.44	78.09	90.02	81.22	81.71	84.99	82.43	86.67

Table 4: Unsupervised Domain adaptation UD results. Base-FS is an upper bound.

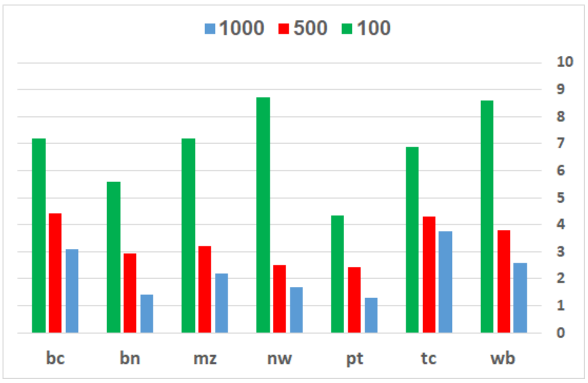

Figure 4: UAS gap between DCST-ENS and the Base parser, as a function of the training set size (100/500/1000), across OntoNotes domains.

with 100 sentences (the green bar) the improvement is higher than 5 UAS points in 6 of 7 cases, among which in 2 (nw and wb) it is higher than 8 UAS points. For 500 training sentences the performance gap drops to 2-4 UAS points, while for 1000 training sentences it is 1-3 points.

This pattern is in line with previous literature on the impact of training methods designed for the lightly-supervised setup, and particularly for self-training when applied to constituency parsing (Reichart and Rappoport, 2007). We note that many research papers failed to improve dependency parsing with traditional self-training even for very small training set sizes (Rush et al., 2012). We also note that syntactically self-trained DCST consistently improves the BiAFFINE parser in our domain adaptation experiments, although the entire training set of the news (nw) section of OntoNotes is used for training.

Impact of Self-training Quality We next aim to test the connection between the accuracy of the self-trained sequence taggers and the quality of the BiAFFINE parser when integrated with the BiLSTM encoders of these taggers. Ideally, we would expect that the higher the quality of the BiLSTM,

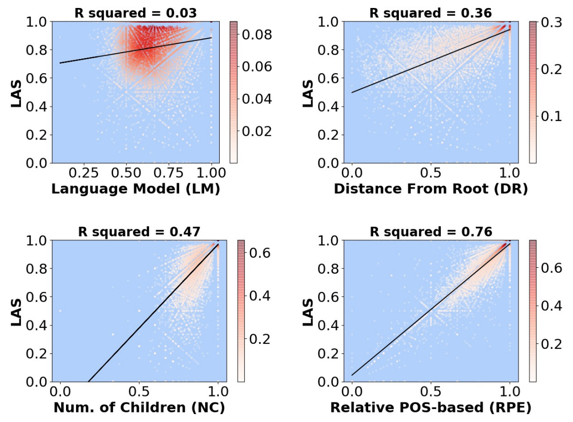

Figure 5: Auxiliary task accuracy scores of each BiLSTM tagger vs. the LAS score of the BiAFFINE parser when integrated with that BiLSTM. The BiLSTM scores are computed on the test sets and reflect the capability of the BiLSTM that was trained on unlabeled data with syntactic signal extracted from the base parser's trees (or as a language model for DCST-LM) to properly tag the test sentences. The points correspond to sentence scores across all OntoNotes 5.0 test sets, and the heat map presents the frequency of each point.

the more positive its impact on the parser. This would indicate that the improvement we see with the DCST models indeed results from the information encoded in the self-trained taggers.

To test this hypothesis, Figure 5 plots, for each of the BiLSTM taggers considered in this paper, the sentence-level accuracy scores of the tagger when applied to the OntoNotes test sets vs. the LAS scores of the BiAFFINE parser that was integrated with the corresponding BiLSTM, when that parser was applied to the same test sentences. In such a plot, if the regression line that fits the points has an R-squared (R^2) value of 1, this indicates a positive linear relation between the self-trained tagger and the parser quality.

The resulting R^2 values are well aligned with the relative quality of the DCST models. Particularly, DCST-LM, the least efficient method where the tagger is trained as a language model rather than on a syntactic signal, has an R^2 of 0.03. DCST-DR and DCST-NC, which are the next in terms of parsing quality (Table 1), have R^2 values of 0.36 and 0.47, respectively, although DCST-DR performs slightly better. Finally, DCST-RPE, the best performing model among the four in all cases but two, has an R^2 value of 0.76. These results provide a positive indication to the hypothesis that the improved parsing quality is caused by the representation model and is not a mere artifact.

Model	AD-NC	AD-DR	AD-PDH	POS Head Error
		OntoNotes		
Base	0.305	0.539	1.371	0.162
DCST-NC	0.274	0.510	1.196	0.146
DCST-DR	0.264	0.460	**1.099**	0.141
DCST-RPE	0.263	0.475	1.128	0.137
DCST-ENS	**0.257**	**0.458**	1.121	**0.135**
		UD		
Base	0.366	0.600	1.377	0.163
DCST-NC	0.327	0.551	1.168	0.148
DCST-DR	0.322	0.538	1.135	0.146
DCST-RPE	0.316	0.534	1.137	0.141
DCST-ENS	**0.312**	**0.524**	**1.128**	**0.139**

Table 5: Tagging scheme error analysis.

Tagging Scheme Quality Analysis We next aim to shed more light on the quality of the tagging schemes with which we train our BiLSTM taggers. We perform an error analysis on the parse trees produced by the final hybrid parser (Figure 3), when each of the schemes is employed in the BiLSTM tagger training step during the lightly-supervised setups. The metrics we compute correspond to the three tagging schemes, and our goal is to examine whether each of the self-trained representation models (BiLSTMs) improves the capability of the final parser to capture the information encoded in its tagging scheme.

Particularly, we consider four metrics: *Absolute Difference of Number of Children (AD-NC)*: The absolute difference between the number of children a word has in the gold tree and the corresponding number in the predicted tree; *Absolute Difference of Distance from the Root (AD-DR)*: The absolute difference between the distance of a word from the root in the gold tree and the corresponding distance in the predicted tree; *Absolute Difference of Positional Distance from the Head (AD-PDH)*: The absolute difference between the positional distance of a word from its head word according to the gold tree and the corresponding number according to the predicted tree (Kiperwasser and Ballesteros, 2018) (we count the words that separate the head from the modifier in the sentence, considering the distance negative if the word is to the right of its head); and *POS Head Error*: an indicator function which returns 0 if the POS tag of the head word of a given word according to the gold tree is identical to the corresponding POS tag in the predicted tree, and 1 otherwise.

For all the metrics we report the mean value across all words in our test sets. The values of AD-NC, AD-DR and AD-PDH are hence in the $[0, M]$ range, where M is the length of the longest sentence in the corpus. The values of the POS Head Error are in the $[0, 1]$ range. For all metrics lower values indicate that the relevant information has been better captured by the final hybrid parser.

Table 5 presents a comparison between the Base parser to our DCST algorithms. All in all, the DCST models outperform the Base parser across all comparisons, with DCST-ENS being the best model in all 8 cases except from one. The analysis indicates that in some cases a BiLSTM tagger with a given tagging scheme directly improves the capability of the final parser to capture the corresponding information. For example, DCST-DR, whose tagging scheme considers the distance of each word from the root of the tree, performs best (OntoNotes) or second best (UD) on the AD-DR metric compared to all other models except from the DCST-ENS model that contains the DCST-DR model as a component. Likewise, DCST-RPE, that encodes information about the POS tag of the head word for every word in the sentence, is the best performing model in terms of POS Head Error. In contrast to the relative success of DCST-RPE and DCST-DR in improving specific capabilities of the parser, DCST-NC, our weakest model across experimental setups, is also the weakest DCST model in this error analysis, even when considering the AD-NC metric that measures success in predicting the number of children a word has in the tree.

Sentence Length Adaptation We next aim to test whether DCST can enhance a parser trained on short sentences so that it can better parse long sentences. Dependency parsers perform better on short sentences, and we would expect self-training to bring in high quality syntactic information from automatically parsed long sentences.

	bc		bn		mz		nw		pt		tc		wb	
Model	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Base+ELMo	77.96	73.97	83.12	80.18	84.62	81.37	83.09	80.35	88.82	85.55	73.84	69.23	79.67	75.77
Base+ELMo+G	74.47	70.91	80.42	77.45	81.15	78.41	80.91	78.24	87.73	84.92	70.19	66.78	76.02	72.68
DCST-ENS+ELMo	**80.00**	**75.94**	**85.02**	**81.98**	**86.24**	**82.54**	**84.56**	**81.91**	**90.27**	**86.86**	**77.68**	**72.72**	**82.00**	**77.93**

Table 6: Lightly supervised OntoNotes results with ELMo embeddings.

	cu		da		fa		id		lv		sl		sv		tr		ur		vi	
Model	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Base+ELMo	72.35	61.43	80.32	76.86	85.84	81.71	73.68	58.01	79.93	73.91	76.40	67.52	81.51	76.10	53.36	34.67	86.11	79.91	71.28	67.04
Base+ELMo+G	**75.47**	**67.07**	79.12	75.05	83.09	79.43	73.00	57.69	72.86	67.13	74.99	69.75	79.66	74.29	53.87	**39.30**	84.83	78.53	66.57	61.56
DCST-ENS+ELMo	73.90	61.62	**82.29**	**78.49**	**87.87**	**83.25**	**74.95**	**58.55**	**82.47**	**76.41**	**79.69**	**70.36**	**83.93**	**78.27**	**59.35**	36.81	**87.51**	**81.53**	**72.76**	**68.48**

Table 7: Lightly supervised UD results with ELMo embeddings.

Model	UAS	LAS
Base	54.86	52.65
DCST-LM	55.26	52.63
Self-Training	54.22	52.16
CVT	50.61	46.13
DCST-ENS	**58.85**	**56.64**

Table 8: Sentence length adaptation results.

For this aim, we replicate the Onotnotes wb in-domain experiment, except that we train the parser on all training set sentences of up to 10 words, use the training set sentences with more than 10 words as unlabeled data for sequence tagger training (Algorithm 1, step 4), and test the final parser on all test sentences with more than 10 words.

Table 8 shows that DCST-ENS improves the Base parser in this setup by 3.99 UAS and LAS points. DCST-LM achieves only a marginal UAS improvement while CVT substantially harms the parser. This result further supports the value of our methods and encourages future research in various under-resourced setups.

ELMo Embeddings Finally, we turn to investigate the impact of deep contextualized word embeddings, such as ELMo (Peters et al., 2018), on the base parser and on the DCST-ENS model. To this end, we replace the Glove/FastText word embeddings from our original experiments with the multilingual ELMo word embeddings of Che et al. (2018). We follow Che et al. (2018) and define the ELMo word embedding for word i as: $w_i = W^{ELMo} \cdot \frac{1}{3} \sum_{j=0}^{2} h_{i,j}^{ELMo}$, where W^{ELMo} is a trainable parameter and $h_{i,j}^{ELMo}$ is the hidden representation for word i in the j'th BiLSTM layer of the ELMo model, which remains fixed throughout all experiments.

We experiment with three models: **Base + ELMo**: the BIAFFINE parser fed by the ELMo word embeddings and trained on the labeled training data; **Base + ELMo + Gating (G)**: The BIAFFINE parser fed by our original word embeddings, and ELMo word embeddings are integrated through our gating mechanism. Training is done on the labeled training data only; and **DCST-ENS + ELMo**: our ensemble parser where the BiLSTM taggers and the Base parser are fed by the ELMo word embeddings.

Tables 6 (OntoNotes) and 7 (UD) summarize the results in the lightly supervised setups with 500 training sentences. Like in previous experiments, DCST-ENS+ELMo is the best performing model in both setups. While Base+ELMo+G is superior in the *cu* and *tr* (LAS) setups, it is inferior in all OntoNotes domains. Note also that DCST-ENS+ELMo improves the UAS results of DCST-ENS from tables 1 and 2 on all OntoNotes domains and on 7 out of 10 UD languages.

9 Conclusions

We proposed a new self-training framework for dependency parsing. Our DCST approach is based on the integration of (a) contextualized embedding model(s) into a neural dependency parser, where the embedding models are trained on word tagging schemes extracted from the trees generated by the base parser on unlabeled data. In multilingual lightly-supervised and domain adaptation experiments, our models consistently outperform strong baselines and previous models.

In future work we intend to explore improved word tagging schemes, sequence tagging architectures and integration mechanisms. We shall also consider cross-language learning where the lexical gap between languages should be overcome.

Acknowledgements

We would like to thank the action editor and the reviewers, as well as the members of the IE@Technion NLP group for their valuable feedback and advice. This research was partially funded by an ISF personal grant No. 1625/18.

References

Steven Abney. 2004. Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3):365–395.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of ACL*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better ud parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64.

Wenliang Chen, Youzheng Wu, and Hitoshi Isahara. 2008. Learning reliable information for dependency parsing adaptation. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 113–120. Association for Computational Linguistics.

Wenliang Chen, Yue Zhang, and Min Zhang. 2014. Feature embedding for dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 816–826.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

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

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for*

Computational Linguistics: Human Language Technologies, pages 1486–1495.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Christian Hadiwinoto and Hwee Tou Ng. 2017. A dependency-based neural reordering model for statistical machine translation. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Yulan He and Deyu Zhou. 2011. Self-training from labeled features for sentiment analysis. *Information Processing & Management*, 47(4):606–616.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1127–1138.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*.

Kenji Imamura and Eiichiro Sumita. 2018. NICT self-training approach to neural machine translation at nmt-2018. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 110–115.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

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

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of CoNLL*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

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

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 337–344. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia

Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 92–97.

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, et al. 2018. Universal dependencies 2.2.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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

Barbara Plank and Željko Agić. 2018. Distant supervision from disparate sources for low-resource part-of-speech tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 614–620, Brussels, Belgium. Association for Computational Linguistics.

Barbara Plank and Gertjan Van Noord. 2011. Effective measures of domain similarity for parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1566–1576. Association for Computational Linguistics.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *The 56th Annual Meeting of the Association for Computational LinguisticsMeeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Alexander M. Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and POS tagging using inter-sentence consistency constraints. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1434–1444. Association for Computational Linguistics.

Piotr Rybak and Alina Wróblewska. 2018. Semi-supervised neural system for tagging, parsing and lematization. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 45–54.

Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 2017. Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79.

Ehsan Shareghi, Yingzhen Li, Yi Zhu, Roi Reichart, and Anna Korhonen. 2019. Bayesian learning for neural dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Tech-*

nologies, Volume 1 (Long and Short Papers), pages 3509–3519.

Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208. Association for Computational Linguistics.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 331–338. Association for Computational Linguistics.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodrıguez. 2019. Viable dependency parsing as sequence labeling. In *Proceedings of NAACL-HLT*, pages 717–723.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *Proceedings of ICLR*.

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1434–1444.

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 *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R. Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, Berlin Chen, Benjamin Van Durme, Edouard Grave, Ellie Pavlick, and Samuel R. Bowman. 2019. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4465–4476.

John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. In *Proceedings of ICLR*.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*.

Kelly W. Zhang and Samuel R. Bowman. 2018. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361, Brussels, Belgium. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge & Data Engineering*, (11):1529–1541.

Yftah Ziser and Roi Reichart. 2018. Pivot based language modeling for improved neural domain adaptation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1241–1251, New Orleans, Louisiana. Association for Computational Linguistics.