Better, Faster, Stronger Sequence Tagging Constituent Parsers

David Vilares
Universidade da Coruña, CITIC
Departamento de Computación
A Coruña, Spain
david.vilares@udc.es

Mostafa Abdou, Anders Søgaard
University of Copenhagen
Department of Computer Science
Copenhagen, Denmark
{abdou,soegaard}@di.ku.dk

Abstract

Sequence tagging models for constituent parsing are faster, but less accurate than other types of parsers. In this work, we address the following weaknesses of such constituent parsers: (a) high error rates around closing brackets of long constituents, (b) large label sets, leading to sparsity, and (c) error propagation arising from greedy decoding. To effectively close brackets, we train a model that learns to switch between tagging schemes. To reduce sparsity, we decompose the label set and use multi-task learning to jointly learn to predict sublabels. Finally, we mitigate issues from greedy decoding through auxiliary losses and sentence-level fine-tuning with policy gradient. Combining these techniques, we clearly surpass the performance of sequence tagging constituent parsers on the English and Chinese Penn Treebanks, and reduce their parsing time even further. On the SPMRL datasets, we observe even greater improvements across the board, including a new state of the art on Basque, Hebrew, Polish and Swedish.[1]

1 Introduction

Constituent parsing is a core task in natural language processing (NLP), with a wide set of applications. Most competitive parsers are slow, however, to the extent that it is prohibitive of downstream applications in large-scale environments (Kummerfeld et al., 2012). Previous efforts to obtain speed-ups have focused on creating more efficient versions of traditional shift-reduce (Sagae and Lavie, 2006; Zhang and Clark, 2009) or chart-based parsers (Collins, 1997; Charniak, 2000). Zhu et al. (2013), for example, presented a fast shift-reduce parser with transitions learned by a SVM classifier. Similarly, Hall et al. (2014) introduced a fast GPU implementation for Petrov and Klein (2007), and Shen et al. (2018) significantly improved the speed of the Stern et al. (2017) greedy top-down algorithm, by learning to predict a list of syntactic distances that determine the order in which the sentence should be split.

In an alternative line of work, some authors have proposed new parsing paradigms that aim to both reduce the complexity of existing parsers and improve their speed. Vinyals et al. (2015) proposed a machine translation-inspired sequence-to-sequence approach to constituent parsing, where the input is the raw sentence, and the 'translation' is a parenthesized version of its tree. Gómez-Rodríguez and Vilares (2018) reduced constituent parsing to sequence tagging, where only n tagging actions need to be made, and obtained one of the fastest parsers to date. However, the performance is well below the state of the art (Dyer et al., 2016; Stern et al., 2017; Kitaev and Klein, 2018a).

Contribution We first explore different factors that prevent sequence tagging constituent parsers from obtaining better results. These include: high error rates when long constituents need to be closed, label sparsity, and error propagation arising from greedy inference. We then present the technical contributions of the work. To effectively close brackets of long constituents, we combine the relative-scale tagging scheme used by Gómez-Rodríguez and Vilares (2018) with a secondary top-down absolute-scale scheme. This makes it possible to train a model that learns how to switch between two encodings, depending on which one is more suitable at each time step. To reduce label sparsity, we recast the constituent-parsing-as-sequence-tagging problem as multi-task learning (MTL) (Caruana, 1997), to decompose a large label

[1] After this paper was submitted, Kitaev and Klein (2018b) have improved our results using their previous self-attentive constituent parser (Kitaev and Klein, 2018a) and BERT representations (Devlin et al., 2018) as input to their system. We will acknowledge these results in the Experiments section.

Proceedings of NAACL-HLT 2019, pages 3372–3383
Minneapolis, Minnesota, June 2 - June 7, 2019.

space and also obtain speed ups. Finally, we mitigate error propagation using two strategies that come at no cost to inference efficiency: auxiliary tasks and policy gradient fine-tuning.

2 Preliminaries

We briefly introduce preliminaries that we will build upon in the rest of this paper: encoding functions for constituent trees, sequence tagging, multi-task learning, and reinforcement learning.

Notation We use w=$[w_0, w_1, ..., w_n]$ to refer to a raw input sentence and bold style lower-cased and math style upper-cased characters to refer to vectors and matrices, respectively (e.g. $\mathbf{x}$ and $\mathbf{W}$).

2.1 Constituent Parsing as Sequence Tagging

Gómez-Rodríguez and Vilares (2018) define a linearization function of the form $\Phi_{|w|} : T_{|w|} \rightarrow L^{(|w|-1)}$ to map a phrase structure tree with $|w|$ words to a sequence of labels of length $|w| - 1$.[2] For each word w_t, the function generates a label $l_t \in L$ of the form l_t=(n_t, c_t, u_t), where:

- n_t encodes the number of ancestors in common between between w_t and w_{t+1}. To reduce the number of possible values, n_t is encoded as the relative variation in the number of common ancestors with respect to n_{t-1}.
- c_t encodes the lowest common ancestor between w_t and w_{t+1}.
- u_t contains the unary branch for w_t, if any.

Figure 1 explains the encoding with an example.

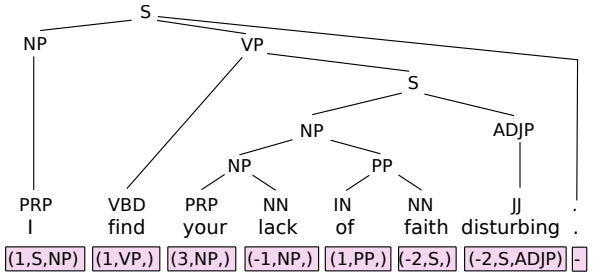

Figure 1: A constituent tree linearized as by Gómez-Rodríguez and Vilares (2018).

[2] They (1) generate a dummy label for the last word and (2) pad sentences with a beginning- and end-of-sentence tokens.

2.2 Sequence Tagging

Sequence tagging is a structured prediction task that generates an output label for every input token. Long short-term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) are a popular architecture for such tasks, often giving state-of-the-art performance (Reimers and Gurevych, 2017; Yang and Zhang, 2018).

Tagging with LSTMs In LSTMs, the prediction for the ith element is conditioned on the output of the previous steps. Let $\text{LSTM}_\theta(\mathbf{x}_{1:n})$ be a parametrized function of the network, where the input is a sequence of vectors $\mathbf{x}_{1:n}$, its output is a sequence of hidden vectors $\mathbf{h}_{1:n}$. To obtain better contextualized hidden vectors, it is possible to instead use bidirectional LSTMS (Schuster and Paliwal, 1997). First, a LSTM^l_θ processes the tokens from left-to-right and then an independent LSTM^r_θ processes them from right-to-left. The ith final hidden vector is represented as the concatenation of both outputs, i.e. $\text{BILSTM}_\theta(\mathbf{x}, i) = \text{LSTM}^l_\theta(\mathbf{x}_{[1:i]}) \circ \text{LSTM}^r_\theta(\mathbf{x}_{[|\mathbf{x}|:i]})$. BILSTMs can be stacked in order to obtain richer representations. To decode the final hidden vectors into discrete labels, a standard approach is to use a feed-forward network together with a softmax transformation, i.e. $P(y|\mathbf{h}_i) = softmax(W \cdot \mathbf{h}_i + \mathbf{b})$. We will use the BILSTM-based model by Yang and Zhang (2018), for direct comparison against Gómez-Rodríguez and Vilares (2018), who use the same model. As input, we will use word embeddings, PoS-tag embeddings and a second word embedding learned by a character-based LSTM layer. The model is optimized minimizing the categorical cross-entropy loss, i.e. $\mathcal{L} = -\sum log(P(y|\mathbf{h}_i))$. The architecture is shown in Figure 2.

2.3 Multi-task Learning

Multi-task learning is used to solve multiple tasks using a single model architecture, with task-specific classifier functions from the outer-most representations (Caruana, 1997; Collobert and Weston, 2008). The benefits are intuitive: sharing a common representation for different tasks acts as a generalization mechanism and allows to address them in a parallel fashion. The *hard-sharing* strategy is the most basic MTL architecture, where the internal representation is fully shared across all tasks. The approach has proven robust for a number of NLP tasks (Bingel and Søgaard, 2017) and comes with certain guarantees if a common, op-

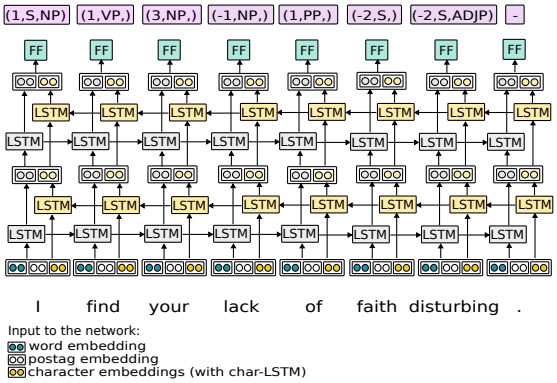

Figure 2: The baseline architecture used in this work. The input to the network is a concatenation of word embeddings, PoS-tag embeddings and a second word embedding learned by a character-based LSTM layer.

timal representation exists (Baxter, 2000). Dong et al. (2015) use it for their multilingual machine translation system, where the encoder is a shared gated recurrent neural network (Cho et al., 2014) and the decoder is language-specific. Plank et al. (2016) also use a hard-sharing setup to improve the performance of BILSTM-based PoS taggers. To do so, they rely on *auxiliary tasks*, i.e, tasks that are not of interest themselves, but that are co-learned in a MTL setup with the goal of improving the network's performance on the main task(s). We will introduce auxiliary tasks for sequence tagging constituent parsing later on in this work. A MTL architecture can also rely on *partial sharing* when the different tasks do not fully share the internal representations (Duong et al., 2015; Rei, 2017; Ruder et al., 2019) and recent work has also shown that hierarchical sharing (e.g. low-level task outputs used as input for higher-level ones) could be beneficial (Søgaard and Goldberg, 2016; Sanh et al., 2018).

2.4 Policy Gradient Fine-tuning

Policy gradient (PG) methods are a class of reinforcement learning algorithms that directly learn a parametrized policy, by which an agent selects actions based on the gradient of a scalar performance measure with respect to the policy. Compared to other reinforcement learning methods, PG is well-suited to NLP problems due to its appealing convergence properties and effectiveness in high-dimensional spaces (Sutton and Barto, 2018).

Previous work on constituent parsing has employed PG methods to mitigate the effect of exposure bias, finding that they function as a model-agnostic substitute for dynamic oracles (Fried and Klein, 2018). Similarly, Le and Fokkens (2017) apply PG methods to Chen and Manning (2014)'s transition-based dependency parser to reduce error propagation. In this work, we also employ PG to fine-tune models trained using supervised learning. However, our setting (sequence tagging) has a considerably larger action space than a transition parser. To deal with that, we will adopt a number of variance reduction and regularization techniques to make reinforcement learning stable.

3 Methods

We describe the methods introduced in this work, motivated by current limitations of existing sequence tagging models, which are first reviewed. The source code can be found as a part of `https://github.com/aghie/tree2labels`.

3.1 Motivation and Analysis

For brevity, we limit this analysis to the English Penn Treebank (PTB) (Marcus et al., 1993). We reproduced the best setup by Gómez-Rodríguez and Vilares (2018), which we are using as baseline, and run the model on the development set. We below show insights for the elements of the output tuple (n_t, c_t, u_t), where n_t is the number of levels in common between w_t and w_{t+1}, c_t is the non-terminal symbol shared at that level, and u_t is a leaf unary chain located at w_t.

High error rate on closing brackets We first focus on predicting relative tree levels (n_t). See Figure 3 for F-scores over n_t labels. The sparsity on negative n_ts is larger than for the positive ones, and we see that consequently, the performance is also significantly worse for negative n_t values, and performance worsens with higher negative values. This indicates that the current model cannot effectively identify the end of long constituents. This is a known source of error for shift-reduce or chart-based parsers, but in the case of sequence tagging parsers, the problem seems particularly serious.

Sparsity The label space is large and sparse: the output labels are simply the possible values in the tuple (n_t, c_t, u_t). An analysis over the PTB training set shows a total of 1423 labels, with 58% of them occurring 5 or less times. These infrequent cases might be difficult to predict, even if some of the elements of the tuple are common.

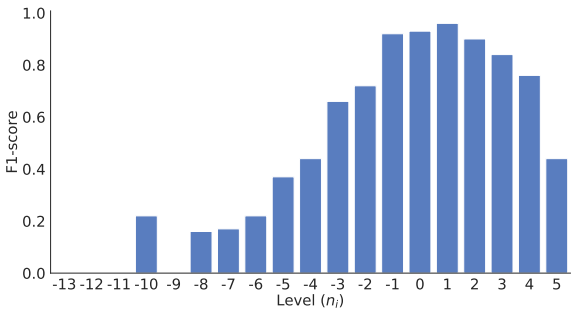

Figure 3: F-score for n_t labels on the PTB dev set using Gómez-Rodríguez and Vilares (2018).

Greedy decoding Greedy decoding is prone to issues such as error propagation. This is a known source of error in transition-based dependency parsing (Qi and Manning, 2017); in contrast with graph-based parsing, in which parsing is reduced to global optimization over edge-factored scores (McDonald et al., 2005).

In the case of BILSTM-based sequence tagging parsers, for a given word w_t, the output label as encoded by Gómez-Rodríguez and Vilares (2018) only reflects a relation between w_t and w_{t+1}. We hypothesize that even if the hidden vector representations are globally contextualized over the whole sequence, the intrinsic locality of the output label also turns into error propagation and consequently causes a drop in the performance. These hypotheses will be tested in §4. In particular, we will evaluate the impact of the different methods intended to perform structured inference (§3.4).

3.2 Dynamic Encodings

Gómez-Rodríguez and Vilares (2018) encode the number of common ancestors n_t, from the output tuple (n_t, c_t, u_t), as the variation with respect to n_{t-1}. We propose instead to encode certain elements of a sentence using a secondary linearization function. The aim is to generate a model that can dynamically switch between different tagging schemes at each time step t to select the one that represents the relation between w_t and w_{t+1} in the most effective way.

On the one hand, the relative-scale encoding is effective to predict the beginning and the end of short constituents, i.e. when a short constituent must be predicted ($|n_t| \leq 2$). On the other hand, with a relative encoding scheme, the F-score was low for words where the corresponding n_t has a large negative value (as showed in Figure 3). This matches a case where a long constituent must be closed: w_t is located at a deep level in the tree and will only (probably) share a few ancestors with w_{t+1}. These configurations are encoded in a more sparse way by a relative scheme, as the n_t value shows a large variability and it depends on the depth of the tree in the current time step. We can obtain a compressed representation of these cases by using a *top-down absolute scale* instead, as any pair of words that share the same m top levels will be equally encoded. The absolute scale becomes however sparse when predicting deep levels. Figure 4 illustrates the strengths and weaknesses of both encodings with an example, and how a dynamically encoded tree helps reduce variability on n_t values.

In our particular implementation, we will be using the following setup:

- $\Phi_{|w|} : T_{|w|} \rightarrow L^{|w|-1}$, the relative-scale encoding function, is used by default.

- $\Omega_{|w|} : T_{|w|} \rightarrow L'^{|w|-1}$ is the secondary linearization function that maps words to labels according to a top-down absolute scale. Ω is used iff: (1) $\Omega(w_{[t:t+1]}) = (n'_t, c'_t, u'_t)$ with $n'_t \leq 3$, i.e. w_t and w_{t+1} share at most the three top levels, and (2) $\Phi(w_{[t:t+1]}) = (n_t, c_t, u_t)$ with $n_t \leq -2$, i.e. w_t is at least located two levels deeper in the tree than w_{t+1}.[3]

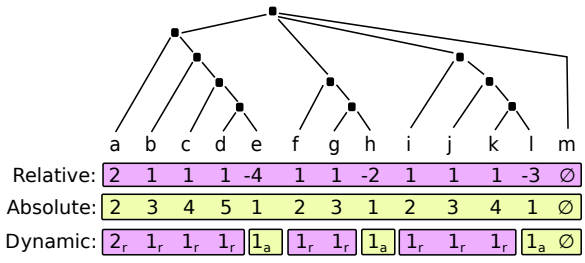

Figure 4: A synthetic constituent tree where n_t is encoded using a relative scheme, a top-down absolute scale, and an ideal dynamic combination. The relative scheme is appropriate to open and close short constituents, but becomes sparse when encoding the large ones, e.g. n_t for the tokens 'e', 'h' and 'l'. The opposite problem is observed for the top-down absolute scheme (e.g. tokens from 'a' to 'd'). The dynamic linearization combines the best of both encodings (we use the subscript 'r' to denote the labels coming from the relative encoding, and 'a' from the absolute one).

[3]The values were selected based on the preliminary experiments of Figure 3.

3.3 Decomposition of the label space

We showed that labels of the form $(n_t, c_t, u_t) \in L$ are sparse. An intuitive approach is to decompose the label space into three smaller sub-spaces, such that $n_i \in N$, $c_i \in C$ and $u_i \in U$. This reduces the output space from potentially $|N| \times |C| \times |U|$ labels to just $|N| + |C| + |U|$. We propose to learn this decomposed label space through a multi-task learning setup, where each of the subspaces is considered a different task, namely task$_N$, task$_C$ and task$_U$. The final loss is now computed as $\mathcal{L} = \mathcal{L}_n + \mathcal{L}_c + \mathcal{L}_u$.

We relied on a hard-sharing architecture, as it has been proved to reduce the risk of overfitting the shared parameters (Baxter, 1997). A natural issue that arises is that the prediction of labels from different label sub-spaces could be interdependent to a certain extent, and therefore a hierarchical sharing architecture could also be appropriate. To test this, in preliminary experiments we considered variants of hierarchical sharing architectures. We fed the output of the task$_U$ as input to task$_N$ and/or task$_C$. Similarly, we tested whether it was beneficial to feed the output of task$_N$ into task$_C$, and viceversa. However, all these results did not improve those of the hard-sharing model. In this context, in addition to a generalization mechanism, the shared representation could be also acting as way to keep the model aware of the potential interdependencies that might exist between sub-tasks.

3.4 Mitigating Effects of Greedy Decoding

We propose two ways to mitigate error propagation arising from greedy decoding in constituent parsing as sequence tagging: auxiliary tasks and policy gradient fine-tuning. Note that we want to optimize bracketing F-score *and* speed. For this reason we do *not* explore approaches that come at a speed cost in testing time, such as beam-search or using conditional random fields (Lafferty et al., 2001) on top of our LSTM.

Auxiliary tasks Auxiliary tasks force the model to take into account patterns in the input space that can be useful to solve the main task(s), but that remain ignored due to a number of factors, such as the distribution of the output label space (Rei, 2017). In a similar fashion, we use auxiliary tasks as a way to force the parser to pay attention to aspects beyond those needed for greedy decoding. We propose and evaluate two separate strategies:

1. Predict partial labels n_{t+k} that are k steps from the current time step t. This way we can jointly optimize at each time step a prediction for the pairs $(w_t, w_{t+1}), \ldots, (w_{t+k}, w_{t+k+1})$. In particular, we will experiment both with previous and upcoming n_k's, setting $|k|$=1.

2. Predict the syntactic distances presented by Shen et al. (2018), which reflect the order a sentence must be split to obtain its constituent tree using a top-down parsing algorithm (Stern et al., 2017). The algorithm was initially defined for binary trees, but its adaptation to n-ary trees is immediate: leaf nodes have a split priority of zero and the ancestors' priority is computed as the maximum priority of their children plus one. In this work, we use this algorithm in a sequence tagging setup: the label assigned to each token corresponds to the syntactic distance of the lowest common ancestor with the next token. This is illustrated in Figure 5.

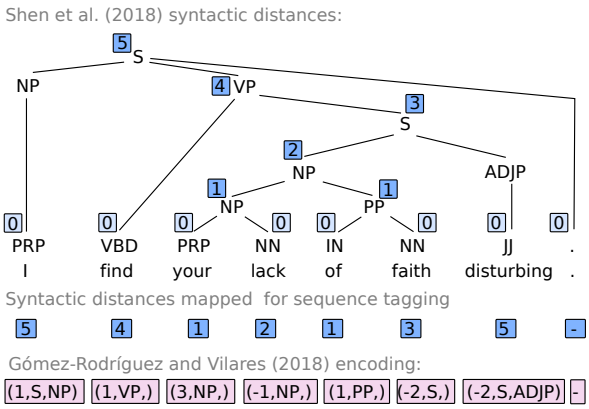

Figure 5: A constituent with syntactic distances attached to each non-terminal symbol, according to Shen et al. (2018). Distances can be used for sequence tagging, providing additional information to our base encoding (Gómez-Rodríguez and Vilares, 2018)

The proposed auxiliary tasks provide different types of contextual information. On the one hand, the encoding of the n_ts by Gómez-Rodríguez and Vilares (2018) only needs to know about w_t and w_{t+1} paths to generate the label for the time step t. On the other hand, to compute the syntactic distance of a given non-terminal symbol, we need to compute the syntactic distances of its subtree, providing a more global, but also sparser context. For training, the loss coming from the auxiliary task(s) is weighted by β=0.1, i.e, the final loss is computed as $\mathcal{L} = \mathcal{L}_n + \mathcal{L}_c + \mathcal{L}_u + \beta \sum_a \mathcal{L}_a$.

Policy gradient fine-tuning Policy gradient training methods allow us to fine-tune our models with a tree-level objective, optimizing directly for bracketing F-score. We start off with a converged supervised model as our initial policy. The sequence labeling model can be seen as a functional approximation of the policy π parametrized by θ, which at timestep t selects a label l_t=(n_t, c_t, u_t)[4] given the current state of the model's parameters, s_t. The agent's reward, R_{tree}, is then derived from the bracketing F-score. This can be seen as a variant of the REINFORCE algorithm (Williams, 1992) where the policy is updated by gradient ascent in the direction of:

$$\Delta_\theta log\pi(l_t|s_t;\theta)R_{tree} \quad (1)$$

Baseline and Variance Reduction We use as baseline a copy of a pre-trained model where the parameters are frozen. The reward used to scale the policy gradient can then be seen as an estimate of the advantage of an action l_t in state s_t over the baseline model. This is equivalent to $R_{tree}-B_{tree}$, where R_{tree} is the bracketing F-score of a sequence sampled from the current policy and B_{tree} is the the tree-level F-score of the sequence greedily predicted by the baseline. To further reduce the variance, we standardize the gradient estimate Δ_θ using its running mean and standard deviation for all candidates seen in training so far. In initial experiments without these augmentations, we observed that fine-tuning with vanilla PG often led to a deterioration in performance. To encourage exploration away from the converged supervised model's policy, we add the entropy of the policy to the objective function (Williams and Peng, 1991). Moreover, following Lillicrap et al. (2015), we optionally add noise sampled from a noise process N to the policy. The gradient of our full fine-tuning objective function takes the following form:

$$\Delta_\theta(log\pi(l_t|s_t;\theta)+N)(R_{tree}-B_{tree}) + \beta\Delta_\theta H(\pi(s_t;\theta)+N) \quad (2)$$

where H is the entropy and β controls the strength of the entropy regularization term.

4 Experiments

We now review the impact of the proposed techniques on a wide variety of settings.

[4] 3 different labels in the MTL setting.

Datasets We use the English Penn Treebank (PTB) (Marcus et al., 1993) and the Chinese Penn Treebank (CTB) (Xue et al., 2005). For these, we use the same predicted PoS tags as Dyer et al. (2016). We also provide detailed results on the SPMRL treebanks (Seddah et al., 2014),[5] a set of datasets for constituent parsing on morphologically rich languages. For these, we use the predicted PoS tags provided together with the corpora. To the best of our knowledge, we provide the first evaluation on the SPMRL datasets for sequence tagging constituent parsers.

Metrics We report bracketing F-scores, using the EVALB and the EVAL-SPMRL scripts. We measure the speed in terms of sentences per second.

Setup We use NCRFpp (Yang and Zhang, 2018), for direct comparison against Gómez-Rodríguez and Vilares (2018). We adopt bracketing F-score instead of label accuracy for model selection and report this performance as our second baseline. After 100 epochs, we select the model that fared best on the development set. We use GloVe embeddings (Pennington et al., 2014) for our English models and `zzgiga` embeddings (Liu and Zhang, 2017) for the Chinese models, for a more homogeneous comparison against other parsers (Dyer et al., 2016; Liu and Zhang, 2017; Fernández-González and Gómez-Rodríguez, 2018). ELMo (Peters et al., 2018) or BERT (Devlin et al., 2018) could be used to improve the precision, but in this paper we focus on keeping a good speed-accuracy tradeoff. For SPMRL, no pretrained embeddings are used, following Kitaev and Klein (2018a). As a side note, if we wanted to improve the performance on these languages we could rely on the CoNLL 2018 shared task pretrained word embeddings (Zeman et al., 2018) or even the multilingual BERT model[6]. Our models are run on a single CPU[7] (and optionally on a consumer-grade GPU for further comparison) using a batch size of 128 for testing. Additional hyperparameters can be found in Appendix A.

4.1 Results

Table 1 contrasts the performance of our models against the baseline on the PTB development set.

[5] Except for Arabic, for which we do not have the license.

[6] https://github.com/google-research/bert/blob/master/multilingual.md

[7] Intel Core i7-7700 CPU 4.2 GHz

Model	F-score	(+/-)	Sents/s
Gómez and Vilares (2018)	89.70	-	109
Our baseline	89.77	(+0.07)	111
+ DE	90.22	(+0.52)	111
+ MTL	90.38	(+0.68)	130
aux(n_{t+1})	90.41	(+0.71)	130
aux(n_{t-1})	90.57	(+0.87)	130
aux(distances)	90.55	(+0.85)	130
+ PG	**90.70**	(+1.00)	130

Table 1: Results on the PTB dev set, compared against Gómez-Rodríguez and Vilares (2018). DE refers to dynamic encoding and MTL to a model that additionally casts the problem as multi-task learning. Each auxiliary task is added separately to the baseline with DE and MTL. Policy gradient fine-tunes the model that includes the best auxiliary task.

To show that the model which employs dynamic encoding is better (+0.52) than the baseline when it comes to closing brackets from long constituents, we compare their F-scores in Figure 6. When we recast the constituent-parsing-as-sequence-tagging problem as multi-task learning, we obtain both a higher bracketing F-score (+0.68) and speed (1.17x faster). Fusing strategies to mitigate issues from greedy decoding also leads to better models (up to +0.87 when adding an auxiliary task[8] and up to +1.00 if we also fine-tune with PG). Note that including auxiliary tasks and PG come at a time cost in training, but not in testing, which makes them suitable for fast parsing.

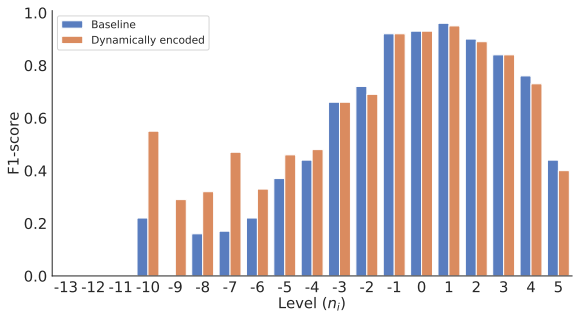

Figure 6: F-score for n_ts on the PTB dev set, obtained by the Gómez-Rodríguez and Vilares (2018) baseline (in blue, first bar for each n_t, already shown in Figure 3) and our model with dynamically encoded trees (in orange, second bar).

Table 2 replicates the experiments on the CTB and the SPMRL dev sets. The dynamic encoding improves the performance of the baseline on large treebanks, e.g. German, French or Korean, but causes some drops in the smaller ones, e.g. Swedish or Hebrew. Overall, casting the problem as multitask learning and the strategies used to mitigate error propagation lead to improvements.

For the experiments on the test sets we select the models that summarize our contributions: the models with dynamic encoding and the multi-task setup, the models including the best auxiliary task, and the models fine-tuned with policy gradient.

Tables 3, 4 and 5 compare our parsers against the state of the art on the PTB, CTB and SPMRL test sets. Gómez-Rodríguez and Vilares (2018) also run experiments without character embeddings, to improve speed without suffering from a big drop in performance. For further comparison, we also include them as additional results (shadowed). In a related line, Smith et al. (2018) show that for dependency parsing two out of three embeddings (word, postag and characters) can suffice.

4.2 Discussion

The results across the board show that the dynamic encoding has a positive effect on 6 out of 10 treebanks. Casting the constituent-parsing-as-sequence-labeling problem as MTL surpasses the baseline for all tested treebanks (and it leads to better parsing speeds too). Finally, by mitigating issues from greedy decoding we further improve the performance of all models that include dynamic encodings and multi-task learning.

On the PTB, our models are both faster and more accurate than existing sequence tagging or sequence-to-sequence models, which already were among the fastest parsers (Gómez-Rodríguez and Vilares, 2018; Vinyals et al., 2015). We also outperform other approaches that were not surpassed by the original sequence tagging models in terms of F-score (Zhu et al., 2013; Fernández-González and Martins, 2015). On the CTB our techniques also have a positive effect. The baseline parses 70 sents/s on the CTB, while the full model processes up to 120. The speed up is expected to be larger than the one obtained for the PTB because the size of the label set for the baseline is bigger, and it is reduced in a greater proportion when the constituent-parsing-as-sequence-labeling problem is cast as MTL.

On the SPMRL corpora, we provide the first evaluation of sequence labeling constituent parsers, to verify if these perform well on mor-

[8] We observed that adding more than one auxiliary task did not translate into a clear improvement. We therefore chose the auxiliary task that performed the best in the development set.

Model	CTB	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish
Our baseline	88.57	87.93	81.09	87.83	89.27	88.85	83.51	92.60	80.11
+DE	88.37	87.91	81.16	**88.81**	89.03	88.70	83.92	93.35	79.57
+MTL	88.57	89.41	81.70	88.52	**92.72**	89.73	84.10	93.81	82.83
aux(n_{t+1})	88.73	89.65	81.95	88.64	92.65	89.69	84.09	93.86	82.82
aux(n_{t-1})	88.48	89.47	81.77	88.58	92.53	89.71	84.13	93.87	82.74
aux(distances)	88.51	89.48	82.02	88.68	92.66	89.80	84.20	93.83	83.12
+PG	**89.01**	**89.73**	**82.13**	88.80	92.66	**89.86**	**84.45**	**93.93**	**83.15**

Table 2: Results on the CTB and SPMRL dev sets

Model	Sents/s	Hardware	F-score
Vinyals et al. (2015)	120	Many CPU	88.30
Coavoux and Crabbé (2016)	168	1 CPU	88.60
Fernández and Martins (2018)	41	1 CPU	90.20
Zhu et al. (2013)	90	1 CPU	90.40
Dyer et al. (2016)	17	1 CPU	91.20
Stern et al. (2017)	76	16 CPU	91.77
Shen et al. (2018)	111	1 GPU	91.80
Kitaev and Klein (2018a) (single model)	213	2 GPU	93.55
Kitaev and Klein (2018a) (with ELMo)	71	2 GPU	95.13
Kitaev and Klein (2018b) (ensemble and BERT)	-	-	*95.77*
Gómez and Vilares (2018)	115	1 CPU	90.00
Our baseline	115	1 CPU	90.06
+DE	115	1 CPU	90.19
+MTL	132	1 CPU	90.36
+ best aux	132	1 CPU	90.59
+PG	132	1 CPU	90.60
+PG	942	1 GPU	90.60
+PG (no char emb)	149	1 CPU	90.50
+PG (no char emb)	1267	1 GPU	90.50

Table 3: Comparison on the PTB test set. Kitaev and Klein (2018b) are results published after this work was submitted (italics represent the cases where they obtain a new state of the art on the corresponding language).

phologically rich languages. We then evaluated whether the proposed techniques can generalize on heterogeneous settings. The tendency observed for the original tagging models by Gómez-Rodríguez and Vilares (2018) is similar to the one for the PTB and CTB: they improve other fast parsers, e.g. Coavoux and Crabbé (2016), in 5 out of 8 treebanks and Fernández-González and Martins (2015) in 7 out of 8, but their performance is below more powerful models. When incorporating the techniques presented in this work, we outperform the original sequence tagging models on all datasets. We outperform the current best model for Basque, Hebrew and Polish (Kitaev and Klein, 2018a) and for Swedish (Björkelund et al., 2014), which corresponds to the four smallest treebanks among the SPMRL datasets. This indicates that even if sequence tagging models are conceptually simple and fast, they can be very suitable when little training data is available. This is also of special interest in terms of research for low-resource languages. Again, casting the problem as MTL reduces the parsing time for all tested treebanks, as reflected in Table 6. Finally, for treebanks such as French, designing methods to handle multi-word expressions could lead to better results, getting closer to other parsers (Coavoux and Crabbé, 2017).

Model	F-score
Zhu et al. (2013)	83.2
Dyer et al. (2016)	84.6
Liu and Zhang (2017)	86.1
Shen et al. (2018)	86.5
Fernández and Gómez-Rodríguez (2018)	86.8
Gómez and Vilares (2018)	84.1
Our baseline	83.90
+DE	83.98
+MTL	84.24
+best aux	85.01
+PG	85.61
+PG (no char emb)	83.93

Table 4: Comparison on the CTB test set

5 Conclusion

We have explored faster and more precise sequence tagging models for constituent parsing. We proposed a multitask-learning architecture that employs dynamic encodings, auxiliary tasks, and policy gradient fine-tuning. We performed experiments on the English and Chinese Penn Treebanks, and also on the SPMRL datasets. Our models improve current sequence tagging parsers on all treebanks, both in terms of performance and speed. We also report state-of-the-art results for the Basque, Hebrew, Polish, and Swedish datasets. The methods presented in this work are specifically designed for constituent parsing. However, it seems natural to apply some of these to other NLP tagging tasks, e.g. using multi-task learning to predict sub-level morphological information for morphologically-rich part-of-speech tagging.

Model	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
Fernández-González and Martins (2015)	85.90	78.75	78.66	88.97	88.16	79.28	91.20	82.80	84.21
Coavoux and Crabbé (2016)	86.24	79.91	80.15	88.69	90.51	85.10	92.96	81.74	85.67
Björkelund et al. (2014) (ensemble)	88.24	82.53	81.66	89.80	91.72	83.81	90.50	85.50	86.72
Coavoux and Crabbé (2017)	88.81	82.49	85.34	89.87	92.34	86.04	93.64	84.00	87.82
Kitaev and Klein (2018a)	89.71	**84.06**	**87.69**	90.35	**92.69**	**86.59**	93.69	84.35	**88.64**
Kitaev and Klein (2018b) (with BERT)	*91.63*	*87.42*	*90.20*	*92.99*	*94.90*	*88.80*	*96.36*	*88.86*	*91.40*
Baseline	89.54	80.56	84.05	88.83	90.42	83.33	92.48	83.67	86.61
+DE	89.56	80.69	84.64	88.80	90.02	83.67	93.20	83.40	86.75
+MTL	90.90	80.98	84.94	91.99	90.63	83.91	93.80	86.26	87.92
+best aux	**91.23**	81.27	84.95	**92.03**	90.60	83.67	93.84	86.62	88.02
+PG	91.18	81.37	84.88	**92.03**	90.65	84.01	**93.93**	**86.71**	88.10
+PG (no char emb)	90.14	81.38	85.09	92.11	90.34	83.46	93.87	86.65	87.88

Table 5: Comparison on the test SPMRL datasets (except Arabic). Kitaev and Klein (2018b) are results published after this work was submitted (italics represent the cases where they obtain a new state of the art on the corresponding language).

Dataset	Baseline speed	Full speed$_{\text{(increase)}}$	Full (no char) speed$_{\text{(increase)}}$
Basque	179	223 (1.25x)	257 (1.44x)
French	76	91 (1.20x)	104 (1.37x)
German	70	100 (1.43x)	108 (1.54x)
Hebrew	44	102 (2.32x)	115 (2.61x)
Hungarian	93	134 (1.44x)	150 (1.61x)
Korean	197	213 (1.08x)	230 (1.17x)
Polish	187	253 (1.35x)	278 (1.49x)
Swedish	98	158 (1.61x)	187 (1.81x)

Table 6: Comparison of speeds on the SPMRL datasets

Acknowlegments

DV has received support from the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150), from the TELEPARES-UDC project (FFI2014-51978-C2-2-R) and the ANSWER-ASAP project (TIN2017-85160-C2-1-R) from MINECO, and from Xunta de Galicia (ED431B 2017/01). MA and AS are funded by a Google Focused Research Award.

References

Jonathan Baxter. 1997. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39.

Jonathan Baxter. 2000. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169. Association for Computational Linguistics.

Anders Björkelund, Özlem Çetinoğlu, Agnieszka Faleńska, Richárd Farkas, Thomas Müller, Wolfgang Seeker, and Zsolt Szántó. 2014. Introducing the ims-wrocław-szeged-cis entry at the spmrl 2014 shared task: Reranking and morpho-syntax meet unlabeled data. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 97–102.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

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

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

Maximin Coavoux and Benoit Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 172–182. Association for Computational Linguistics.

Maximin Coavoux and Benoit Crabbé. 2017. Multilingual lexicalized constituency parsing with word-level auxiliary tasks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 331–336.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the*

eighth conference on European chapter of the Association for Computational Linguistics, pages 16–23. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1723–1732.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 845–850.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018. Faster Shift-Reduce Constituent Parsing with a Non-Binary, Bottom-Up Strategy. *ArXiv e-prints*.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533. Association for Computational Linguistics.

Daniel Fried and Dan Klein. 2018. Policy gradient as a proxy for dynamic oracles in constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 469–476. Association for Computational Linguistics.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent parsing as sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324. Association for Computational Linguistics.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237. Association for Computational Linguistics.

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

Nikita Kitaev and Dan Klein. 2018a. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018b. Multilingual constituency parsing with self-attention and pre-training. *CoRR*, abs/1812.11760.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Minh Le and Antske Fokkens. 2017. Tackling error propagation through reinforcement learning: A case of greedy dependency parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 677–687. Association for Computational Linguistics.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher 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. Association for Computational Linguistics.

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)*, volume 1, pages 2227–2237.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics.

Peng Qi and Christopher D. Manning. 2017. Arc-swift: A novel transition system for dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–117. Association for Computational Linguistics.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2019. Latent multi-task architecture learning. *To appear at AAAI 2019*.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 691–698, Stroudsburg, PA, USA. Association for Computational Linguistics.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2018. A hierarchical multi-task approach for learning embeddings from semantic tasks. *arXiv preprint arXiv:1811.06031*.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109. Dublin City University.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180. Association for Computational Linguistics.

Aaron Smith, Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2018. An investigation of the interactions between pre-trained word embeddings, character models and pos tags in dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720. Association for Computational Linguistics.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Oriol Vinyals, Łũkasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ronald J Williams and Jing Peng. 1991. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.*, 11(2):207–238.

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79, Melbourne, Australia. Association for Computational Linguistics.

Daniel Zeman, Jan Haji, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: Multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171. Association for Computational Linguistics.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443. Association for Computational Linguistics.

A Appendices

For the BILSTM-based model, we essentially follow the configuration of the baseline (Gómez-Rodríguez and Vilares, 2018) for an homogenous comparison. We detail the hyperparameters in Table 7.[9]

Hyperparameter	**Value**
BILSTM size	800
# BILSTM layers	20
optimizer	SGD
loss	cat. cross-entropy
learning rate	0.2
decay (linear)	0.05
momentum	0.9
dropout	0.5
word emb size	100
features size	20
character emb size	50
batch size training	8
training epochs	100
batch size test	128
PG finetuning Hyperparameter	**Value**
# samples	8
learning rate	0.0005
entropy regularization coefficient	0.01
variance reduction burn-in # of examples	1000
layers frozen	word & char embeddings
noise process initial stddev	0.1
noise process desired action stddev	0.5
noise process adaptation coefficient	1.05

Table 7: Additional hyperparameters of the base model and Policy Gradient fine-tuning

[9]Note that the noise sampling is only used for Swedish in the final models based on development set results with and without it.