# Not All Linearizations Are Equally Data-Hungry in Sequence Labeling Parsing

**Alberto Muñoz-Ortiz**[1], **Michalina Strzyz**[1,2], **David Vilares**[1]
[1]Universidade da Coruña, CITIC, Spain
[2]Priberam Labs, Portugal
alberto.munoz.ortiz@udc.es
michalina.strzyz@priberam.pt, david.vilares@udc.es

## Abstract

Different linearizations have been proposed to cast dependency parsing as sequence labeling and solve the task as: (i) a head selection problem, (ii) finding a representation of the token arcs as bracket strings, or (iii) associating partial transition sequences of a transition-based parser to words. Yet, there is little understanding about how these linearizations behave in low-resource setups. Here, we first study their data efficiency, simulating data-restricted setups from a diverse set of rich-resource treebanks. Second, we test whether such differences manifest in truly low-resource setups. The results show that head selection encodings are more data-efficient and perform better in an ideal (gold) framework, but that such advantage greatly vanishes in favour of bracketing formats when the running setup resembles a real-world low-resource configuration.

## 1 Introduction

Dependency parsing (Mel'cuk et al., 1988; Kübler et al., 2009) has achieved clear improvements in recent years, to the point that graph-based (Martins et al., 2013; Dozat et al., 2017) and transition-based (Ma et al., 2018; Fernández-González and Gómez-Rodríguez, 2019) parsers are already very accurate on certain setups, such as English news. In this line, Berzak et al. (2016) have pointed out that the performance on these setups is already on par with that expected from experienced human annotators.

Thus, the efforts have started to focus on related problems such as parsing different domains or multi-lingual scenarios (Sato et al., 2017; Song et al., 2019; Ammar et al., 2016), creating faster models (Volokh, 2013; Chen and Manning, 2014), designing low-resource and cross-lingual parsing techniques (Tiedemann et al., 2014; Zhang et al., 2019), or infusing syntactic knowledge into models (Strubell et al., 2018; Rotman and Reichart, 2019).

This work will lie in the intersection between fast parsing and low-resource languages. Recent work has proposed encodings to cast parsing as sequence labeling (Spoustová and Spousta, 2010; Strzyz et al., 2019; Gómez-Rodríguez et al., 2020; Li et al., 2018; Kiperwasser and Ballesteros, 2018). This approach computes a linearized tree of a sentence of length $n$ in $n$ tagging actions, providing a good speed/accuracy trade-off. Also, it offers a naïve way to infuse syntactic information as an embedding or feature (Ma et al., 2019; Wang et al., 2019). Such encodings have been evaluated on English and multi-lingual setups, but there is no study about their behaviour on low-resource setups, and what strengths and weaknesses they might exhibit.

**Contribution** We study the behaviour of linearizations for dependency parsing as sequence labeling in low-resource setups. First, we explore their data efficiency, i.e. if they can exploit their full potential with less supervised data. To do so, we simulate different data-restricted setups from a diverse set of rich-resource treebanks. Second, we shed light about their performance on truly low-resource treebanks. The goal is to determine whether tendencies from the experiments in the previous phase hold when the language is truly low-resource and when secondary effects of real-world low-resource setups, such as using predicted part-of-speech (PoS) tags or no PoS tags, impact more certain types of linearizations.

## 2 Related work

Low-resource parsing has been explored from perspectives such as unsupervised parsing, data augmentation, cross-lingual learning, or data-efficiency of models. For instance, on unsupervised parsing, Klein and Manning (2004) and Spitkovsky et al. (2010) have worked on generative models to determine whether to continue or stop attaching de-



*Proceedings of Recent Advances in Natural Language Processing*, pages 978–988
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_111

pendents to a token, while others (Le and Zuidema, 2015; Mohananey et al., 2020) have studied how to use self-training for unsupervised parsing.

On data augmentation, McClosky et al. (2006) used self-training to annotate extra data, while others have focused on linguistically motivated approaches to augment treebanks. This is the case of Vania et al. (2019) or Dehouck and Gómez-Rodríguez (2020), who have proposed methods to replace subtrees within a given sentence.

On cross-lingual learning, authors such as Søgaard (2011) or McDonald et al. (2011) trained delexicalized parsers in a source rich-resource treebank, which are then used to parse a low-resource target language. Falenska and Çetinoğlu (2017) explored lexicalized versus delexicalized parsers and compared them on low-resource treebanks, depending on factors such as the treebank size and the PoS tags performance. Wang and Eisner (2018) created synthetic treebanks that resemble the target language by permuting constituents of distant treebanks. Naseem et al. (2012) and Täckström et al. (2013) tackled this same issue, but from the model side, training on rich-resource languages in such way the model learns to detect the aspects of the source languages that are relevant for the target language. Recently, Mulcaire et al. (2019) used a LSTM to build a polyglot language model, which is then used to train on top of it a parser that shows cross-lingual abilities in zero-shot setups.

On data-efficiency, research work has explored the impact of the use of different amounts of data, motivated by the lack of annotated data or by the lack of quality of it. For instance, Lacroix et al. (2016a) showed how a transition-based parser with a dynamic oracle can be used without any modifications to parse partially annotated data. They found that this setup is useful to train low-resource parsers on sentence-aligned texts, from a rich-resource treebank to an automatically translated low-resource language, where only precisely aligned tokens are used for the projection in the target dataset. Lacroix et al. (2016b) studied the effect that pre-processing and post-processing has in annotation projection, and concluded that quality should prevail over quantity. Related to training with restricted data, Anderson and Gómez-Rodríguez (2020) showed that when distilling a graph-based parser for faster inference time, models with smaller treebanks suffered less. Dehouck et al. (2020) also distilled models for Enhanced Universal Dependencies (EUD) parsing with different amounts of data, observing that less training data usually translated into slightly lower performance, while offering better energy consumption. Garcia et al. (2018) showed, in the context of Romance languages, that peeking samples from related languages and adapting them to the target language is useful to train a model that performs on par with one trained on fully (but still limited) manually annotated data. Restricted to constituent parsing, Shi et al. (2020) analyzed the role of the dev data in unsupervised parsing. They pointed out that many unsupervised parsers use the score on the dev set as a signal for hyperparameter updates, and show that by using a handful of samples from that development set to train a counterpart supervised model, the results outperformed those of the unsupervised setup. Finally, there is work describing the impact that the size of the parsing training data has on downstream tasks that use syntactic information as part of the input (Sagae et al., 2008; Gómez-Rodríguez et al., 2019).

## 3 Preliminaries

In what follows, we review the existing families of encodings for parsing as sequence labeling (§3.1) and the models that we will be using (§3.2).

### 3.1 Encodings for sequence labeling dependency parsing

Sequence labeling assigns *one* output label to every input token. Many problems are cast as sequence labeling due to its fast and simple nature, like PoS tagging, chunking, super tagging, named-entity recognition, semantic role labeling, and parsing. For dependency parsing, to create a linearized tree it suffices to assign each word $w_i$ a *discrete* label of the form $(x_i, l_i)$, where $l_i$ is the dependency type and $x_i$ encodes a subset of the arcs of the tree related to such word. Although only labels seen in the training data can be predicted, Strzyz et al. (2020) show that the coverage is almost complete. We distinguish three families of encodings, which we now review (see also Figure 1).

**Head-selection encodings** (Spoustová and Spousta, 2010; Li et al., 2018; Strzyz et al., 2019). Each word label component $x_i$ encodes its head as an index or an (abstracted) offset. This can be done by labeling the target word with the (absolute) index of its head token, or by using a relative offset that accounts for the difference between the dependent and head indexes. In this work, we

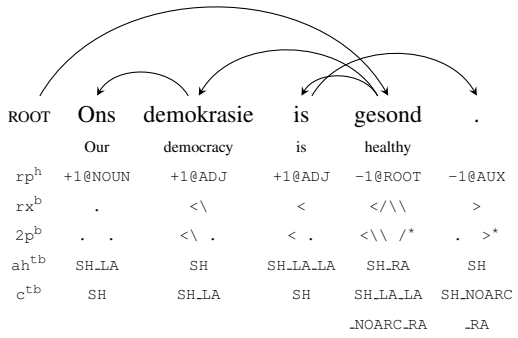


Figure 1: Example of the linearizations used in this work in a sentence from the Afrikaans$_{\text{AfriBooms}}$ treebank. Dependency types are omitted for simplicity.

chose a relative PoS-based encoding ($\texttt{rp}^{\texttt{h}}$) that has shown to perform consistently better among the linerarizations of this family. Here, $x_i$ is a tuple $(p_i, o_i)$, such that if $o_i > 0$ the head of $w_i$ is the $o_i$th word to the right of $w_i$ whose PoS tag is $p_i$; if $o_i < 0$, the head of $w_i$ is the $o_i$th word to the left of $w_i$ whose PoS tag is $p_i$. Among its advantages, we find the capacity to encode any non-projective tree and words being directly and only linked to its head, but on the other hand it is dependent on external factors (e.g. PoS tags).[1]

**Bracketing-based encodings** (Yli-Jyrä and Gómez-Rodríguez, 2017). Each $x_i$ encodes a sort of incoming and outgoing arcs of a given word and its neighbors, represented as bracket strings. More particularly, in Strzyz et al. (2019) each $x_i$ is a string that follows the expression $(\texttt{<})\,?\,(\,(\texttt{\textbackslash})\star\,|\,(\texttt{/})\star)\,(\texttt{>})\,?$, where < means that $w_{i-1}$ has an incoming arc from the right, $k$ times \ means that $w_i$ has $k$ outgoing arcs towards the left, $k$ times / means that $w_{i-1}$ has $k$ outgoing arcs to the right, and > means that $w_i$ has an arc coming from the left. This encoding produces a compressed label set while not relying on external features, such as PoS tags. However, when it comes to non-projectivity, it can only analyze crossing arcs in opposite directions. To counteract this, it is possible to define a linearization using a second independent pair of brackets (denoted with '*') to encode a 2-planar tree (Strzyz et al., 2020).[2] In this work we are considering experiments with both the restricted non-projective ($\texttt{rx}^{\texttt{b}}$) and the 2-planar bracketing encodings ($\texttt{2p}^{\texttt{b}}$).

**Transition-based encodings** (Gómez-Rodríguez et al., 2020). Each $x_i$ encodes a sub-sequence of the transitions to be generated by a *left-to-right* transition-based parser. Given a sequence of transitions $t = t_1, ..., t_m$ with exactly $n$ read transitions[3], it splits $t$ into $n$ chunks and assigns the $i$th chunk to the $i$th word. Its main advantage is more abstract, allowing to *automatically derive encodings* relying on any left-to-right transition based parser (including dependency, constituency and semantic parsers). According to Gómez-Rodríguez et al. (2020), they produce worse results than the bracketing encodings, but we include them in this work for completeness. In particular, we consider mappings from arc-hybrid (Kuhlmann et al., 2011) ($\texttt{ah}^{\texttt{tb}}$) and Covington (2001) ($\texttt{c}^{\texttt{tb}}$), which are projective and non-projective transition-based algorithms.

To post-process corrupted predicted labels, we follow the heuristics described in each encoding paper.

### 3.2 Sequence labeling framework

**Notes** Let $w$ be a sequence of words $[w_1, w_2, ..., w_{|w|}]$, then $\vec{w}$ is a sequence of word vectors that will be used as the input to our models. Each $\vec{w}_i$ will be a concatenation of: (i) a word embedding, (ii) a second word embedding computed through a char-LSTM, (iii) and *optionally* a PoS tag embedding (we will discuss more about this last point in §4).

We use bidirectional long short-term memory networks (biLSTMs; Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) to train our sequence labeling parsers. BiLSTMs are a strong baseline used in recent work across a number of tasks (Yang and Zhang, 2018; Reimers and Gurevych, 2017). More particularly, we use two layers of biLSTMs, and each hidden vector $\vec{h}_i$ from the last biLSTM layer (associated to each input vector $\vec{w}_i$) is fed to separate feed-forward networks that are in charge of predicting each of the label components of the linearization (i.e. $x_i$ and $l_i$) using softmaxes, relying on hard-sharing multi-task learning (MTL; Caruana, 1997; Ruder, 2017). Following

[1] Other head-selection variants encode arcs based on word properties different than PoS tags (Lacroix, 2019).

[2] An $x$-planar tree can be separated into $x$ planes, where the arcs belonging to the same plane do not cross.

[3] In left-to-right parsers, a read transition is an action that puts a word from the buffer into the stack. For algorithms such as the arc-standard or arc-hybrid this is only the `shift` action, while in the arc-eager both the `shift` and `right-arc` actions are read transitions. See also (Nivre, 2008).

§3.1, for all the encodings, except the 2-planar encoding, we will use a 2-task MTL setup: one task will predict $x_i$ according to each encoding specifics, and the other one will predict the dependency type, $l_i$. For the 2-planar bracketing encoding, which uses a second pair of brackets to predict the arcs from the second plane, we use instead a 3-task MTL setup, where the difference is that the prediction of $x_i$ is split into two tasks: one that predicts the first plane brackets and another task that predicts the brackets from the second plane.[4]

It is worth noting that for this particular work we skipped computational expensive models, such as BERT (Devlin et al., 2019). There are three main reasons for this. First, the experiments in this paper imply training a total of 760 parsing models (see more details in §4), making the training on BERT (or variants) less practical. Second, there is not a multilingual or specific-language BERT model for all languages, and this could be the source of uncontrolled variables that could have an impact on the performance, and thereof on the conclusions.[5] Third, even under the assumption of all language-specific BERT models being available, these are pre-trained on different data that add extra noise, which could be undesirable for our purpose.

## 4 Methodology and experiments

We design two studies, detailed in §4.1 and 4.2:

1. We explore if some encodings are more data-efficient than others. To do so, we will simulate data-restricted setups, selecting rich-resource languages and using partial data. The goal is to test if some encodings are learnable with fewer data, or if other ones could obtain a better performance instead, but only under the assumption of very large data being available.

2. We focus on truly low-resource setups. This can be seen as a confirmation experiment to see if the findings under data-restricted setups hold for under-studied languages, and to confirm what sequence labeling linearizations are more recommendable under these conditions.

[4] We used 3 tasks because it establishes a more fair comparison in terms of label sparsity and follows previous work.

[5] Also, for the case of multilingual models, there is literature that concludes different about what makes a language beneficial for other under a BERT-based framework. For instance, Wu and Dredze (2019) conclude that sharing a large amount sub-word pieces is important, while authors such as Pires et al. (2019) or Artetxe et al. (2020) state otherwise.

**Experimental setups** For experiments 1 and 2, we consider three setups that might have a different impact across the encodings:

1. Gold PoS tags setup: We train and run the models under an ideal framework that uses gold PoS tags as part of the input. The reason is that encodings such as $\texttt{rp}^{\texttt{h}}$ rely on PoS tags to rebuild the linearized tree. This way, using gold PoS tags helps estimate the *optimal* data-efficiency and learnability of these parsers under perfect (but unreal) conditions.

2. Predicted PoS tags setup: Setup 1 cannot truly reflect the performance that the encodings would obtain under real-world data-restricted conditions. Predicted PoS tags will be less helpful because their quality will degrade. This issue can affect more to the $\texttt{rp}^{\texttt{h}}$ encoding, since it requires them to rebuild the tree from the labels, and miss-predicted PoS tags could propagate errors during decoding. Here, we train taggers for each treebank, using the same architecture used for the parsers. To be coherent with the data-restricted setups, taggers will be trained on the same amount of data used for the parsers. Appendix A discusses the PoS taggers performance.

3. No PoS tags setup: We train the models without using any PoS tags as part of the input. It is worth noting that the setup is somewhat forced for the $\texttt{rp}^{\texttt{h}}$ encoding, since we will still need to externally run the taggers to obtain the PoS tags and rebuild the tree. Yet, we include the PoS-based encoding for completeness, and to have a better understanding about how different families of encodings suffer from not (or minimally) using PoS tags. For instance, that is a simple way to obtain simpler and faster parsing models, as part of the pipeline does not need to be executed, and the input vectors to the models will be smaller, translating into faster executions too. Also, in low-resource setups, PoS tags might not be available or the tagging models are not accurate enough to help deep learning models (Zhou et al., 2020; Anderson and Gómez-Rodríguez, 2021).

### 4.1 Experiment 1: Encodings data-efficiency

**Data** We chose 11 treebanks from UD2.7 (Zeman et al., 2020) with more than 10 000 training sentences: $\text{German}_{\text{HDT}}$, $\text{Czech}_{\text{PDT}}$,

Russian$_{\text{SynTagRus}}$, Classical Chinese$_{\text{Kyoto}}$, Persian$_{\text{PDT}}$, Estonian$_{\text{EDT}}$, Romanian$_{\text{Nonstandard}}$, Korean$_{\text{Kaist}}$, Ancient Greek$_{\text{PROIEL}}$, Hindi$_{\text{HDTB}}$ and Latvian$_{\text{LVTB}}$. They consider different families, scripts and levels of non-projectivity (see Appendix B). To simulate data-restricted setups, we created training subsets of 100, 500, 1 000, 5 000 and 10 000 samples, as well as the total training set. The training sets were shuffled before the division.

**Setup** To assess the data-efficiency, we proceed as follows. As the rp$^{\text{h}}$ encoding has showed the strongest performance in previous work for multi-lingual setups (Strzyz et al., 2019; Gómez-Rodríguez et al., 2020), we are taking these models as the reference and an *a priori* upper bound. Then, we compute the difference of the mean UAS (across the 11 treebanks) between the rp$^{\text{h}}$ and each of the other linearizations, for all the models trained up to 10 000 sentences. The goal is to determine which encodings suffer more when training with limited data and monitor to what extent the tendency holds as more data is introduced. We compute the statistically significant difference between the rp$^{\text{h}}$ and the other encodings, using the p-value ($p < 0.05$) of a paired t-test on the scores distribution, following recommended practices for dependency parsing (Dror et al., 2018). Finally, we show specific results for the models trained on the whole treebanks. In this work, we will report UAS over LAS, since the differences in the encodings lie in how they encode the dependency arcs and not their types.

**Results** Tables 1, 2 and 3 show the difference of the mean UAS for each encoding with respect to the rp$^{\text{h}}$ one; for the gold PoS tags, predicted PoS tags and no PoS tags setups, respectively. For the gold PoS tags setup, the rp$^{\text{h}}$ encoding performs better than the bracketing (rx$^{\text{b}}$ and 2p$^{\text{b}}$) and the transition-based (ah$^{\text{tb}}$ and c$^{\text{tb}}$) encodings, for all the training splits. Yet, the gap narrows as the number of training sentence increases. For the predicted PoS tags setup, the relative PoS-based encoding performs better for the smallest set of 100 sentences, but slightly worse for the sets of 500 and 1 000 sentences with respect to rx$^{\text{b}}$ and 2p$^{\text{b}}$. With more data, the tendency resembles the one from the gold PoS tags setup. Third, for the setup without PoS tags, the tendency reverses. The bracketing encodings perform better, particularly for the smallest test sets, but the gap narrows as the number of training sentences increases.

| # Sentences | rp$^{\text{h}}$ | rx$^{\text{b}}$ | 2p$^{\text{b}}$ | ah$^{\text{tb}}$ | c$^{\text{tb}}$ |
|---|---|---|---|---|---|
| 100 | 68.34 | -2.15 | -2.42 | -5.82 | -9.96 |
| 500 | 76.94 | -1.58 | -1.5 | -5.21 | -9.35 |
| 1 000 | 80.29 | -1.42 | -1.43 | -5.16 | -8.9 |
| 5 000 | 86.54 | -1.16 | -1.26 | -3.62 | -7.04 |
| 10 000 | 88.26 | -0.8 | -0.72 | -3.52 | -5.67 |

Table 1: Average UAS difference for the subsets of the rich-resource treebanks under the gold PoS tags setup. Blue and yellow cells show the UAS increase and decrease with respect to the rp$^{\text{h}}$ encoding, respectively.

| # Sentences | rp$^{\text{h}}$ | rx$^{\text{b}}$ | 2p$^{\text{b}}$ | ah$^{\text{tb}}$ | c$^{\text{tb}}$ |
|---|---|---|---|---|---|
| 100 | 41.87 | -0.42 | -0.19 | -1.9 | -3.59 |
| 500 | 63.45 | -0.01 | 0.14 | -1.96 | -5.73 |
| 1000 | 68.10 | 0.25 | 0.17 | -2.44 | -5.53 |
| 5 000 | 78.56 | -0.62 | -0.63 | -2.53 | -5.44 |
| 10 000 | 82.29 | -0.37 | -0.36 | -2.49 | -4.44 |

Table 2: Average UAS difference for the subsets of the rich-resource treebanks under the predicted PoS tags setup.

| # Sentences | rp$^{\text{h}}$ | rx$^{\text{b}}$ | 2p$^{\text{b}}$ | ah$^{\text{tb}}$ | c$^{\text{tb}}$ |
|---|---|---|---|---|---|
| 100 | 35.60 | 9.06 | 9.31 | 7.57 | 4.83 |
| 500 | 58.63 | 3.04 | 2.45 | 0.99 | -2.26 |
| 1 000 | 63.99 | 3.59 | 3.42 | 0.83 | -2.24 |
| 5 000 | 75.57 | 1.47 | 1.55 | -0.19 | -3.07 |
| 10 000 | 79.90 | 1.22 | 1.54 | -0.87 | -2.93 |

Table 3: Average UAS difference for the subsets of the rich-resource treebanks under the no PoS tags setup.

**Discussion** The results from the experiments shed light about differences existing across different encodings and running configurations. First, under an ideal, gold environment, the rp$^{\text{h}}$ encoding makes a better use of limited data than the bracketing and transition-based encodings. Second, the predicted PoS tag setup shows that the performance of the PoS taggers can have a significant impact on the performance for the rp$^{\text{h}}$ encoding. More interestingly, weaknesses from different encodings seem to manifest to different extents depending on the amount of training data. For instance, when training data is scarce (100 sentences), bracketing encodings still cannot outperform the rp$^{\text{h}}$ encoding, despite the lower performance of the PoS taggers. However, when working with setups ranging from 500 to 1000 sentences, there is a slight advantage of the bracketing encodings with respect to rp$^{\text{h}}$, suggesting that with this amount of data, bracketing encodings could be the preferable choice, since they seem able to exploit their potential in a better way than the rp$^{\text{h}}$ encoding can exploit not fully accurate PoS tags. With more training samples, the relative PoS-based

encoding is again the best performing model across the board. In §4.2 we will discuss deeper how for truly low-resources languages the advantage in favour of bracketing representations exacerbates more for the predicted and no PoS tags setups.

| | rp$^{h}$ | rx$^{b}$ | 2p$^{b}$ | ah$^{tb}$ | c$^{tb}$ |
|---|---|---|---|---|---|
| grc | 83.09 | 79.89$^{--}$ | 81.7$^{-}$ | 78.57$^{--}$ | 79.86$^{--}$ |
| lzh | 90.21 | 89.56$^{-}$ | 89.24$^{-}$ | 89.04$^{--}$ | 89.18$^{-}$ |
| cs | 91.61 | 90.49$^{--}$ | 90.91$^{--}$ | 88.18$^{--}$ | 85.64$^{--}$ |
| et | 85.62 | 84.79$^{-}$ | 84.91$^{-}$ | 81.86$^{--}$ | 81.11$^{--}$ |
| de | 96.69 | 95.95$^{--}$ | 96.38$^{--}$ | 95.15$^{--}$ | 86.51$^{--}$ |
| hi | 94.69 | 94.09$^{--}$ | 94.43$^{-}$ | 93.05$^{--}$ | 85.02$^{--}$ |
| ko | 87.26 | 86.24$^{--}$ | 86.52$^{-}$ | 85.68$^{--}$ | 84.06$^{--}$ |
| lv | 85.3 | 83.88$^{-}$ | 84.01$^{-}$ | 80.88$^{--}$ | 81.38$^{--}$ |
| fa | 92.61 | 92.07$^{-}$ | 92.44$^{-}$ | 90.45$^{--}$ | 87.09$^{--}$ |
| ro | 90.49 | 89.68$^{-}$ | 89.63$^{--}$ | 87.39$^{--}$ | 86.38$^{--}$ |
| ru | 91.23 | 90.1$^{--}$ | 90.1$^{--}$ | 88.19$^{--}$ | 84.96$^{--}$ |
| Avg | **89.89** | 88.79 | 89.12 | 87.13 | 84.65 |

Table 4: UAS for the rich-resource treebanks, using the whole training set and the gold PoS tags setup. The red (--) and green cells (++) show that a given encoding performed worse or better than the rp$^{h}$ model, and that the difference is statistically significant. Lime and yellow cells mean that there is no a significant difference between a given encoding and the rp$^{h}$, appending a $^{+}$ or a $^{-}$ when they performed better or worse than the rp$^{h}$.

| | rp$^{h}$ | rx$^{b}$ | 2p$^{b}$ | ah$^{tb}$ | c$^{tb}$ |
|---|---|---|---|---|---|
| grc | 80.2 | 77.61$^{--}$ | 79.21$^{-}$ | 76.49$^{--}$ | 77.71$^{-}$ |
| lzh | 79.93 | 79.8$^{-}$ | 79.42$^{-}$ | 79.41$^{-}$ | 79.54$^{-}$ |
| cs | 90.04 | 88.93$^{--}$ | 89.34$^{--}$ | 86.67$^{--}$ | 84.25$^{--}$ |
| et | 81.07 | 80.36$^{-}$ | 80.34$^{-}$ | 77.71$^{--}$ | 76.95$^{--}$ |
| de | 95.85 | 95.14$^{--}$ | 95.54$^{--}$ | 94.34$^{--}$ | 85.79$^{--}$ |
| hi | 92.22 | 91.76$^{-}$ | 92.21$^{-}$ | 90.72$^{--}$ | 83.24$^{--}$ |
| ko | 84.25 | 83.44$^{-}$ | 83.42$^{-}$ | 82.98$^{--}$ | 81.25$^{--}$ |
| lv | 70.65 | 71.98$^{++}$ | 71.08$^{+}$ | 68.9$^{-}$ | 68.97$^{-}$ |
| fa | 90.39 | 89.8$^{-}$ | 90.32$^{-}$ | 88.27$^{--}$ | 85.28$^{--}$ |
| ro | 87.32 | 86.64$^{-}$ | 86.49$^{-}$ | 84.44$^{--}$ | 83.5$^{--}$ |
| ru | 88.71 | 88.13$^{-}$ | 88.24$^{-}$ | 85.93$^{--}$ | 82.96$^{--}$ |
| Avg | **85.51** | 84.87 | 85.06 | 83.26 | 80.86 |

Table 5: UAS for the rich-resource treebanks, using the whole training set and the predicted PoS tags setup.

Tables 4, 5 and 6 show the UAS on the full training sets of the rich-resource treebanks for the gold PoS tags, predicted PoS tags, and no PoS tags setups. The goal is to show if under large amounts of data some of the encodings could perform on par with rp$^{h}$, since Tables 1 and 2 indicated that differences in performance across encodings decreased when the number of training samples increase. Although performance across encodings becomes closer, their ranking remains the same.

| | rp$^{h}$ | rx$^{b}$ | 2p$^{b}$ | ah$^{tb}$ | c$^{tb}$ |
|---|---|---|---|---|---|
| grc | 77.84 | 77.41$^{-}$ | 79.16$^{+}$ | 75.64$^{-}$ | 76.99$^{-}$ |
| lzh | 79.99 | 81.02$^{+}$ | 80.75$^{+}$ | 81.11$^{++}$ | 81.42$^{++}$ |
| cs | 88.67 | 88.2$^{-}$ | 88.64$^{-}$ | 85.8$^{--}$ | 84.23$^{--}$ |
| et | 77.85 | 79.69$^{++}$ | 79.99$^{++}$ | 77.15$^{-}$ | 76.27$^{-}$ |
| de | 94.51 | 95.09$^{++}$ | 95.41$^{++}$ | 94.18$^{-}$ | 83.54$^{--}$ |
| hi | 89.43 | 91.7$^{++}$ | 91.98$^{++}$ | 90.72$^{++}$ | 82.86$^{--}$ |
| ko | 79.39 | 82.18$^{++}$ | 82.15$^{++}$ | 81.88$^{++}$ | 80.3$^{++}$ |
| lv | 62.56 | 71.17$^{++}$ | 72.38$^{++}$ | 66.78$^{++}$ | 69.38$^{++}$ |
| fa | 89.14 | 90.39$^{++}$ | 90.48$^{++}$ | 88.49$^{-}$ | 84.54$^{--}$ |
| ro | 85.28 | 86.41$^{+}$ | 86.94$^{++}$ | 84.25$^{-}$ | 83.04$^{--}$ |
| ru | 83.35 | 83.98$^{++}$ | 84.5$^{++}$ | 83.42$^{++}$ | 80.26$^{--}$ |
| Avg | 82.55 | 84.29 | **84.76** | 82.67 | 80.26 |

Table 6: UAS for the rich-resource treebanks, using the whole training set and the no PoS tags setup.

| | rp$^{h}$ | rx$^{b}$ | 2p$^{b}$ | ah$^{tb}$ | c$^{tb}$ |
|---|---|---|---|---|---|
| af | 88.02 | 85.7$^{--}$ | 85.48$^{--}$ | 81.84$^{--}$ | 78.6$^{--}$ |
| cop | 88.73 | 88.43$^{-}$ | 88.72$^{-}$ | 85.5$^{--}$ | 84.35$^{--}$ |
| fo | 84.04 | 83.76$^{-}$ | 84.09$^{+}$ | 81.78$^{--}$ | 79.53$^{--}$ |
| hu | 79.75 | 76.14$^{--}$ | 76.13$^{--}$ | 71.66$^{--}$ | 64.27$^{--}$ |
| lt | 51.98 | 50.28$^{-}$ | 50.19$^{-}$ | 45.0$^{-}$ | 46.6$^{-}$ |
| mt | 81.81 | 81.05$^{-}$ | 80.82$^{-}$ | 76.78$^{--}$ | 74.98$^{--}$ |
| mr | 77.43 | 76.46$^{-}$ | 75.97$^{-}$ | 76.94$^{-}$ | 73.54$^{-}$ |
| ta | 74.96 | 73.1$^{-}$ | 71.9$^{-}$ | 71.74$^{-}$ | 66.01$^{--}$ |
| te | 90.01 | 91.26$^{+}$ | 90.43$^{+}$ | 90.01$^{+}$ | 89.46$^{-}$ |
| wo | 86.19 | 84.64$^{--}$ | 84.51$^{--}$ | 80.65$^{--}$ | 77.43$^{--}$ |
| Avg | **80.29** | 79.08 | 78.82 | 76.19 | 73.48 |

Table 7: UAS for the low-resource treebanks for the gold PoS tags setup.

### 4.2 Experiment 2: Encodings performance on truly low-resource languages

**Data** We choose the 10 smallest treebanks[6] (in terms of training sentences) that had a dev set: Lithuanian$_{\text{HSE}}$, Marathi$_{\text{UFAL}}$, Hungarian$_{\text{Szeged}}$, Telugu$_{\text{MTG}}$, Tamil$_{\text{TTB}}$, Faroese$_{\text{FarPaHC}}$, Coptic$_{\text{Scriptorium}}$, Maltese$_{\text{MUDT}}$, Wolof$_{\text{WTB}}$ and Afrikaans$_{\text{AfriBooms}}$ (see Appendix B). Their sizes range between 153 and 1350 training sentences, most being around or between 500 and 1 000 (see Appendix C).

**Setup** We rerun a subset of the experiments from §4.1, to check if the results follow the same trends, and conclusions are therefore similar.

**Results** Tables 7, 8 and 9 show the UAS for each encoding and treebank for the gold PoS tags setup, the predicted PoS tags setup and the no PoS tags setup, respectively. Again, under perfect conditions, the relative PoS-based encoding performs overall better, except for Telugu, which seems to

[6]Code switching treebanks and small treebanks of rich-resource languages were not considered.

| | rp$^{h}$ | rx$^{b}$ | 2p$^{b}$ | ah$^{tb}$ | c$^{tb}$ |
|---|---|---|---|---|---|
| af | 81.84 | 80.29$^{-}$ | 79.9$^{-}$ | 77.3$^{--}$ | 73.61$^{--}$ |
| cop | 85.77 | 86.25$^{+}$ | 85.92$^{+}$ | 83.14$^{--}$ | 81.84$^{--}$ |
| fo | 77.04 | 76.97$^{-}$ | 77.52$^{+}$ | 75.23$^{-}$ | 74.24$^{-}$ |
| hu | 70.52 | 68.51$^{-}$ | 68.77$^{-}$ | 64.98$^{--}$ | 58.37$^{--}$ |
| lt | 30.28 | 34.53$^{+}$ | 33.11$^{+}$ | 31.23$^{+}$ | 29.91$^{-}$ |
| mt | 74.6 | 75.64$^{+}$ | 75.07$^{+}$ | 71.17$^{--}$ | 70.35$^{--}$ |
| mr | 66.99 | 67.96$^{+}$ | 67.23$^{+}$ | 68.93$^{+}$ | 67.23$^{+}$ |
| ta | 57.11 | 60.73$^{+}$ | 57.57$^{+}$ | 58.77$^{+}$ | 55.51$^{-}$ |
| te | 86.41 | 87.93$^{+}$ | 87.93$^{+}$ | 86.96$^{+}$ | 86.69$^{+}$ |
| wo | 76.88 | 76.4$^{-}$ | 76.3$^{-}$ | 73.24$^{--}$ | 70.84$^{--}$ |
| Avg | 70.74 | **71.52** | 70.93 | 69.10 | 66.86 |

Table 8: UAS for the low-resource treebanks for the predicted PoS tags setup.

| | rp$^{h}$ | rx$^{b}$ | 2p$^{b}$ | ah$^{tb}$ | c$^{tb}$ |
|---|---|---|---|---|---|
| af | 79.86 | 80.78$^{+}$ | 80.07$^{+}$ | 75.47$^{--}$ | 73.76$^{--}$ |
| cop | 84.36 | 85.76$^{+}$ | 85.13$^{+}$ | 83.07$^{-}$ | 81.28$^{--}$ |
| fo | 73.98 | 77.08$^{++}$ | 77.04$^{++}$ | 75.07$^{+}$ | 73.67$^{-}$ |
| hu | 63.63 | 65.21$^{+}$ | 64.8$^{+}$ | 62.04$^{-}$ | 56.17$^{--}$ |
| lt | 26.89 | 34.62$^{++}$ | 35.38$^{++}$ | 34.06$^{++}$ | 32.92$^{+}$ |
| mt | 70.95 | 75.5$^{++}$ | 75.3$^{++}$ | 71.69$^{+}$ | 70.32$^{-}$ |
| mr | 64.08 | 66.75$^{+}$ | 67.96$^{+}$ | 69.66$^{+}$ | 64.56$^{+}$ |
| ta | 52.79 | 60.03$^{++}$ | 56.61$^{+}$ | 59.58$^{++}$ | 54.95$^{+}$ |
| te | 85.44 | 88.49$^{+}$ | 88.63$^{+}$ | 87.1$^{+}$ | 86.82$^{+}$ |
| wo | 73.11 | 77.17$^{++}$ | 76.95$^{++}$ | 74.01$^{+}$ | 70.86$^{-}$ |
| Avg | 67.51 | **71.14** | 70.79 | 69.18 | 66.53 |

Table 9: UAS for the low-resource treebanks for the no PoS tags setup.

be an outlier. For the predicted PoS tags setup, the bracketing-based encodings perform consistently better for most of the treebanks. For the no PoS tags setup, the bracketing-based encodings obtain, on average, more than 3 points than the relative PoS-head selection encoding, which even performs worse than the transition-based encodings.

**Discussion** These experiments help elaborate on the findings of §4.1. With respect to the ideal gold PoS tags setup, things do not change much, and the relative PoS-based encoding performs overall better. Still, this should not be taken as a ground truth about how the encodings will perform in real-world setups. For instance, for the predicted PoS tags setup, the bracketing-based encodings perform consistently better in most of the treebanks. This reinforces some of the suspicions found in the experiments of Table 2, where training on rich-resource languages, but with limited data, revealed that bracketing encodings performed better, although just slightly. Also, it is worth noting that most of the low-resource treebanks tested in this work have a number of training sentences in the range where the bracketing-based encodings performed better for the predicted PoS tags setup in Table 2, i.e. from 500 to 1 000 sentences (see Appendix C). Yet, the better performance of bracketing encodings is more evident when running on real low-resource treebanks. This does not only suggest that the bracketing encodings are better for real low-resource sequence labeling parsing, but it could also pose more general limitations for other low-resource NLP tasks that are evaluated only on 'faked' low-resource setups, and that could lead to incomplete or even misleading conclusions.

Overall, the results suggest that bracketing encodings are the most suitable linearizations for real low-resource sequence labeling parsing.

## 5 Conclusion

We have studied sequence labeling encodings for dependency parsing in low-resource setups. First, we explored which encodings are more data-efficient under different conditions that include the use of gold PoS tags, predicted PoS tags and no PoS tags as part of the input. By restricting training data for rich-resource treebanks, we observe that although bracketing encodings are less data-efficient than head-selection ones under ideal conditions, this disadvantage can vanish when the input conditions are not gold and data is limited. Second, we studied their performance under the same running configurations, but on truly low-resource languages. These results show more clearly the greatest utility of bracketing encodings over the rest of the ones when training data is limited and the quality of external factors, such as PoS tags, is affected by the low-resource nature of the problem.

## Acknowledgements

This work is supported by a 2020 Leonardo Grant for Researchers and Cultural Creators from the FBBVA.[7] The work also receives funding from the European Research Council (FASTPARSE, grant agreement No 714150), from ERDF/MICINN-AEI (ANSWER-ASAP, TIN2017-85160-C2-1-R, SCANNER, PID2020-113230RB-C21), from Xunta de Galicia (ED431C 2020/11), and from Centro de Investigación de Galicia 'CITIC', funded by Xunta de Galicia and the European Union (European Regional Development Fund- Galicia 2014-2020 Program) by grant ED431G 2019/01.

[7]FBBVA accepts no responsibility for the opinions, statements and contents included in the project and/or the results thereof, which are entirely the responsibility of the authors.

## References


Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Mark Anderson and Carlos Gómez-Rodríguez. 2020. Distilling neural networks for greener and faster dependency parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 2–13, Online. Association for Computational Linguistics.

Mark Anderson and Carlos Gómez-Rodríguez. 2021. What taggers fail to learn, parsers need the most. *arXiv preprint arXiv:2104.01083*.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Yevgeni Berzak, Yan Huang, Andrei Barbu, Anna Korhonen, and Boris Katz. 2016. Anchoring and agreement in syntactic annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2215–2224, Austin, Texas. Association for Computational Linguistics.

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

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, Doha, Qatar. Association for Computational Linguistics.

Michael A Covington. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, volume 1. Citeseer.

Mathieu Dehouck, Mark Anderson, and Carlos Gómez-Rodríguez. 2020. Efficient EUD parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 192–205, Online. Association for Computational Linguistics.

Mathieu Dehouck and Carlos Gómez-Rodríguez. 2020. Data augmentation via subtree swapping for dependency parsing of low-resource languages. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3818–3830, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

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.

Agnieszka Falenska and Özlem Çetinoğlu. 2017. Lexicalized vs. delexicalized parsing in low-resource scenarios. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 18–24, Pisa, Italy. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Left-to-right dependency parsing with pointer networks. 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 710–716, Minneapolis, Minnesota. Association for Computational Linguistics.

Marcos Garcia, Carlos Gómez-Rodríguez, and Miguel A Alonso. 2018. New treebank or repurposed? on the feasibility of cross-lingual parsing of romance languages with universal dependencies. *Natural Language Engineering*, 24(1):91–122.

Carlos Gómez-Rodríguez, Iago Alonso-Alonso, and David Vilares. 2019. How important is syntactic parsing accuracy? an empirical evaluation on rule-based sentiment analysis. *Artificial Intelligence Review*, 52(3):2081–2097.

Carlos Gómez-Rodríguez, Michalina Strzyz, and David Vilares. 2020. A unifying theory of transition-based and sequence labeling parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3776–3793, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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

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

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 478–485, Barcelona, Spain.

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

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682, Portland, Oregon, USA. Association for Computational Linguistics.

Ophélie Lacroix. 2019. Dependency parsing as sequence labeling with head-based encoding and multi-task learning. In *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pages 136–143.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016a. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California. Association for Computational Linguistics.

Ophélie Lacroix, Guillaume Wisniewski, and François Yvon. 2016b. Cross-lingual dependency transfer : What matters? assessing the impact of pre- and post-processing. In *Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP*, pages 20–29, San Diego, California. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 651–661, Denver, Colorado. Association for Computational Linguistics.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. Improving neural machine translation with neural syntactic distance. 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 2032–2037, Minneapolis, Minnesota. Association for Computational Linguistics.

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, Melbourne, Australia. Association for Computational Linguistics.

André FT Martins, Miguel B Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622.

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

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Igor Aleksandrovic Mel'cuk et al. 1988. *Dependency syntax: theory and practice*. SUNY press.

Anhad Mohananey, Katharina Kann, and Samuel R. Bowman. 2020. Self-training for unsupervised parsing with PRPN. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 105–110, Online. Association for Computational Linguistics.

Phoebe Mulcaire, Jungo Kasai, and Noah A. Smith. 2019. Low-resource parsing with crosslingual contextualized representations. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 304–315, Hong Kong, China. Association for Computational Linguistics.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637, Jeju Island, Korea. Association for Computational Linguistics.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. 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*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Guy Rotman and Roi Reichart. 2019. Deep contextualized self-training for low resource dependency parsing. *Transactions of the Association for Computational Linguistics*, 7:695–713.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Kenji Sagae, Yusuke Miyao, Rune Saetre, and Jun'ichi Tsujii. 2008. Evaluating the effects of treebank size in a practical application for parsing. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 14–20, Columbus, Ohio. Association for Computational Linguistics.

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, Vancouver, Canada. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2020. On the role of supervision in unsupervised constituency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7611–7621, Online. Association for Computational Linguistics.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 682–686, Portland, Oregon, USA. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Daniel Gildea, Mo Yu, Zhiguo Wang, et al. 2019. Leveraging dependency forest for neural medical relation extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 208–218.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California. 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.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Viable dependency parsing as sequence labeling. 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 717–723, Minneapolis, Minnesota. Association for Computational Linguistics.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2020. Bracketing encodings for 2-planar dependency parsing. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2472–2484, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071, Atlanta, Georgia. Association for Computational Linguistics.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 130–140.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1105–1116, Hong

Kong, China. Association for Computational Linguistics.

Alexander Volokh. 2013. Performance-oriented dependency parsing.

Dingquan Wang and Jason Eisner. 2018. Synthetic data made to order: The case of parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1325–1337, Brussels, Belgium. Association for Computational Linguistics.

Yufei Wang, Mark Johnson, Stephen Wan, Yifang Sun, and Wei Wang. 2019. How to best use syntax in semantic role labelling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5338–5343, Florence, Italy. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China. Association for Computational Linguistics.

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.

Anssi Yli-Jyrä and Carlos Gómez-Rodríguez. 2017. Generic axiomatization of families of noncrossing graphs in dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1745–1755, Vancouver, Canada. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, et al. 2020. Universal dependencies 2.7. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2019. Cross-lingual dependency parsing using code-mixed TreeBank. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 997–1006, Hong Kong, China. Association for Computational Linguistics.

Houquan Zhou, Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Is pos tagging necessary or even helpful for neural dependency parsing? In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 179–191. Springer.

## Appendix A Taggers accuracy

| # Sentences | Average accuracy |
|---|---|
| 100 | 57.08 |
| 500 | 81.24 |
| 1 000 | 85.03 |
| 5 000 | 90.90 |
| 10 000 | 92.77 |
| Low-resource | 85.03 |

Table 10: Average accuracy of the taggers for the splits of the rich-resource treebanks and the complete low-resource treebanks.

## Appendix B Treebanks information

| | % Non-projective sentences | Family | Script |
|---|---|---|---|
| de | 6.76 | IE (Germanic) | Latin |
| cs | 11.49 | IE (West Slavic) | Latin |
| ru | 7.53 | IE (East Slavic) | Cyrillic |
| lzh | 0.01 | Sino-Tibetan | Chinese characters |
| fa | 14.22 | IE (Iranian) | Persian |
| et | 3.22 | Uralic | Latin |
| ro | 5.43 | IE (Romance) | Latin |
| ko | 21.70 | Korean | Korean |
| grc | 37.52 | IE (Greek) | Greek |
| hi | 13.60 | IE (Indo-Aryan) | Devanagari |
| lv | 6.53 | IE (Baltic) | Latin |
| af | 22.23 | IE (Germanic) | Latin |
| cop | 13.24 | Afro-Asiatic | Coptic |
| fo | 0.19 | IE (Germanic) | Latin |
| hu | 27.10 | Uralic | Latin |
| lt | 14.07 | IE (Baltic) | Latin |
| mt | 3.86 | Semitic | Latin |
| mr | 6.01 | IE (Indo-Aryan) | Devanagari |
| ta | 1.67 | Dravidian | Tamil |
| te | 0.15 | Dravidian | Telugu |
| wo | 2.99 | Niger-Congo | Latin |

Table 11: Information about the treebanks used.

## Appendix C Low-resource treebank sizes

| | # Sentences |
|---|---|
| $\text{Afrikaans}_{\text{AfriBooms}}$ | 1 315 |
| $\text{Coptic}_{\text{Scriptorium}}$ | 1 089 |
| $\text{Faroese}_{\text{FarPaHC}}$ | 1020 |
| $\text{Hungarian}_{\text{Szeged}}$ | 910 |
| $\text{Lithuanian}_{\text{HSE}}$ | 153 |
| $\text{Maltese}_{\text{MUDT}}$ | 1 123 |
| $\text{Marathi}_{\text{UFAL}}$ | 373 |
| $\text{Tamil}_{\text{TTB}}$ | 400 |
| $\text{Telugu}_{\text{MTG}}$ | 1 051 |
| $\text{Wolof}_{\text{WTB}}$ | 1 188 |

Table 12: Number of training sentences for the low-resource treebanks used.