The Fragility of Multi-Treebank Parsing Evaluation

Iago Alonso-Alonso, David Vilares and Carlos Gómez-Rodríguez
Universidade da Coruña, CITIC
Departamento de Ciencias de la Computación y Tecnologías de la Información
Campus de Elviña s/n, 15071
A Coruña, Spain
{iago.alonso,david.vilares,carlos.gomez}@udc.es

Abstract

Treebank selection for parsing evaluation and the spurious effects that might arise from a biased choice have not been explored in detail. This paper studies how evaluating on a single subset of treebanks can lead to weak conclusions. First, we take a few contrasting parsers, and run them on subsets of treebanks proposed in previous work, whose use was justified (or not) on criteria such as typology or data scarcity. Second, we run a large-scale version of this experiment, create vast amounts of random subsets of treebanks, and compare on them many parsers whose scores are available. The results show substantial variability across subsets and that although establishing guidelines for good treebank selection is hard, it is possible to detect potentially harmful strategies.

1 Introduction

A limitation in NLP evaluation lies in the association between solving a dataset *versus* solving a task. Datasets are domain-specific, their sizes differ and they are only available for a handful of languages and cultures (Hershcovich et al., 2022). Yet, we often ignore that the chances that these results generalize in the real world are scarce. In this context, the conclusions extracted from a single dataset should be taken with caution.

For dependency parsing, the Universal Dependencies framework (UD; Zeman et al., 2020) mitigates some of these issues. For instance, version 2.8 of UD includes 202 treebanks and 114 languages covering diverse linguistic typologies, treebanks with different amounts of data, and domains. Paradoxically, this also complicates decisions when it comes to comparing dependency parsers in multilingual environments, which can be summarized as: *how to choose a small but representative set of treebanks?* Although there are shared tasks (Zeman et al., 2017, 2018) that do consider experiments over a wide set of treebanks and help understand parsing models, such setups do not usually stick when the shared tasks end, and authors often run their models only in a handful of treebanks (de Lhoneux and Nivre, 2016; Ma et al., 2018; Kulmizev et al., 2019, *inter alia*). This mostly happens for justified reasons: lack of computational resources to train the models in a reasonable amount of time, energy usage concerns, difficulties to summarize large experiments, or interest in specific phenomena (e.g. non-projectivity). Thus, a good treebank selection strategy is crucial to reduce the chances of selecting an unrepresentative subset of treebanks, which could lead to weak conclusions. Furthermore, even when using the whole UD collection is viable, treebank selection can still be relevant as UD is not a representative sample of languages (e.g., 62 out of the 114 languages in v2.8 are Indo-European), so coarse-grained measures like averages over all treebanks may be misleading.

Contribution We hypothesize that using a single subset of treebanks can be a weak approach to extract conclusions about the performance of parsers and their rankings. To test so, we design two experiments. First, we choose representative models of different paradigms: a graph-based (Dozat et al., 2017), a transition-based (Fernández-González and Gómez-Rodríguez, 2019), and a sequence tagging (Strzyz et al., 2019) parser; and evaluate them on a few subsets defined in the literature, looking for different trends. Then, we redefine the previous experiment on a large scale. We take the output of dozens of parsers on the treebanks used at the UD CoNLL 2018 shared task (Zeman et al., 2018) to study the variability of parsing rankings over a million of fixed-size, randomly generated subsets.

2 Related work

The appropriateness of experimental setups for parsing evaluation has been studied in recent years from different perspectives.

Some authors have focused on determining what

Proceedings of the 29th International Conference on Computational Linguistics, pages 5345–5359
October 12–17, 2022.

are the treebank particularities that make some of them easier to parse than others. For instance, the size of the training set is widely known to be an important factor to obtain accurate results in dependency parsing (Dehouck and Denis, 2019; Vania et al., 2019). Other aspects such as domain similarity (Wisniewski and Yvon, 2019) or annotation similarity (Dredze et al., 2007; Cohen et al., 2012) between the training and test sets have also been studied, showing that they can greatly affect the performance of parsers. Other particularities that can also affect the performance on a treebank are linguistic variation (Nivre et al., 2007), annotation criteria (Kübler et al., 2008; Rosa, 2015), arc direction (Rehbein et al., 2017), average dependency length (Gulordava and Merlo, 2016), non-projectivity (Kuhlmann and Nivre, 2010), morphological richness (Tsarfaty et al., 2013) or information-theoretic metrics (Corazza et al., 2013), among other factors.

Although not specifically for parsing but NLP, Gorman and Bedrick (2019) and Søgaard et al. (2021) comment that the way data is split can play a role on test results, and thus on conclusions. Extrapolating this to parsing, it would suggest that some parsers could obtain better results for certain treebanks just due to data splitting decisions, and not due to a linguistic motivation that would explain a given language being harder to parse than other. Recently, Søgaard (2020) studied the influence of overlap between trees in training and test sets in a given split, and concluded that (the amount of) graph isomorphism between the training and test set trees partially explains why some treebanks are easier or harder to parse than others. However, Anderson et al. (2021) replicated the study, controlling for covariants, and proved that much of this observation is explained by relevant covariants like treebank size and mean test sentence length.

Another line of research more related to our work involves the studies that compare how different parsing algorithms behave on the same held-out test sets. McDonald and Nivre (2007, 2011) showed that non-neural transition-based and graph-based parsers perform overall similarly, but produce different types of errors, with transition-based parsers being weaker for long dependencies and graph-based parsers weaker for shorter, more local ones. Relatedly, de Lhoneux et al. (2017a) compared a neural and non-neural transition based parser, showing that the former is not only clearly better at longer dependencies, but that it also needs less training data to parse effectively. Kulmizev et al. (2019) replicated the work by McDonald and Nivre for neural versions of those parsers and, contrarily, demonstrated that the contextualization of the input vectors with recurrent networks results into both types of parsers showing a much more homogeneous behavior. Also related to this, Anderson and Gómez-Rodríguez (2020b) showed how different transition-based algorithms are prone to outperform others on a specific treebank according to their inherent dependency displacement biases.

To the best of our knowledge, there have been only two papers in the literature that specifically focus on presenting methodologies to choose a suitable set of treebanks for parsing evaluation, both centered on UD and with the goal of obtaining a small sample of treebanks that is representative of the full UD collection (not necessarily of human languages as a whole). de Lhoneux and Nivre (2016); de Lhoneux et al. (2017b) do so by manually selecting treebanks to enforce typological diversity as well as representativity in other relevant aspects for parsing, like projectivity or treebank size. In turn, Schluter and Agić (2017) take an automatic, quantitative approach, obtaining a sample by clustering using delexicalized parsing performance. While many other papers have presented and used subsets of UD treebanks for evaluation, they either do not focus on representativity (e.g. Ma et al. (2018)) or follow one of these methodologies (e.g. Anderson and Gómez-Rodríguez (2020a)).

3 Hyphothesis and methodology

As suggested above, parsing conclusions on multilingual environments are usually drawn from empirical research, which are prone to be parser-specific, experiment-specific, as well as treebank dependent.

Hypothesis We delve into this problem and hypothesize that parser comparisons based on running experiments and taking accuracy metrics on a given (reasonably-sized) subset of treebanks may lead to weak conclusions on rankings or differences in performance; as the magnitude and/or sign of the differences between parsers can change substantially depending on the choice of said subset.

3.1 Methodology

To test our hypothesis, we design two experiments:

Experiment 1: *few* controlled parsers, *few* pre-existing subsets In §4, we choose three con-

trasting parsers belonging to different parsing paradigms. Then, we train and evaluate them on a number of pre-defined (multilingual) subsets that were proposed in previous work (and later adopted by other authors as well). These existing subsets present different particularities, such as a high Indo-European bias (Ma et al., 2018), rich and diverse typologies (de Lhoneux et al., 2017b), or data scarcity issues (Dehouck et al., 2021), among others. Our aim is to see whether considering only a few robust parsers (treated as black boxes) and only a few already established subsets of treebanks, we can obtain different conclusions about their behaviors.

Experiment 2: *many* parsers, *many* randomized subsets In §5, we design a large-scale variant of the previous experiment. Assuming access to many parsers and treebanks, we ask: *could we obtain (reasonably-sized) subsets of treebanks that show very different behaviors?*, or to state it differently, *can parsing rankings be sensitive to the subset of treebanks where they are evaluated?* To do so, we use as a proxy the results from the CoNLLU Shared Task 2018 (Zeman et al., 2018), where 26 parsers participated and presented their experiments for 82 treebanks. We then create a random sample of 1 million subsets out of the $\sim 2.13 \times 10^{12}$ possible (multilingual) subsets of size 10. If a parser is cross-linguistically robust, then the variability of its position in the ranking should be small across all the studied subsets, while if their behavior is more unstable it could change dramatically, indicating that evaluating on a single subset of treebanks is not desirable.

4 Experiment 1: *few* controlled parsers, *few* pre-existing subsets

We take a few representative parsers (§4.1) and pre-defined subsets from the literature (§4.2) based, sometimes, on a careful treebank selection strategy.

4.1 The parsing models

We choose a graph-based (Dozat et al., 2017), a transition-based (Fernández-González and Gómez-Rodríguez, 2019), and a sequence labeling parser (Strzyz et al., 2019). We review them briefly, but we refer the reader to the papers for the details.

Bi-affine graph-based parser (`gb-DM17`; Dozat et al., 2017) It first computes contextualized vectors for each word using bidirectional LSTMs (biLSTMs; Hochreiter and Schmidhuber, 1997). After that, the model computes for each word a *head* and a *dependent* representation, which are sent through a bi-affine attention, determining for each token which is the most likely head. Here, we rely on the `supar`[1] package, which has been widely adopted by the community. We detail its hyperparameters in Appendix A (Table 7).

Left-to-right, transition-based, pointer network parser (`tb-FG19`; Fernández-González and Gómez-Rodríguez, 2019) It is a transition-based system, where at each time-step the pointer network predicts the index of the head for the focus token, and moves to the next one. The model uses an encoder-decoder architecture that in the first stage computes a hidden state representation for each token using biLSTMs. After that, the decoder predicts the tree left to right, computing a score attention between the current focus word and the encoder output sequence, excluding the own vector. We use the `syntacticpointer`[2] package. Appendix A (Table 8) details the hyperparameters.

Sequence labeling parser (`sl-S+19`; Strzyz et al., 2019) It outputs a dependency tree for each sentence of size n, using exactly n predictions and biLSTM tagging models. There are different ways to encode the trees (Spoustová and Spousta, 2010; Lacroix, 2019; Gómez-Rodríguez et al., 2020), but we will rely on the 2-planar bracketing encoding (Strzyz et al., 2020), which encodes 99.9% of non-projective trees and offers a robust behavior, including low-resource setups (Muñoz-Ortiz et al., 2021). We use the `dep2labels`[3] package. The hyperparameters are indicated in Appendix A (Table 9).

Parser comparability Sequence labeling parsers often underperform biaffine and pointer network parsers (Anderson and Gómez-Rodríguez, 2021), but we include them as a lower bound control parser. We kept fundamental architectural decisions of the parsers, e.g. how they compute character-level vectors or the strategies for cycle deletion, as it is not clear (or viable) that the same setup is optimal for all models. Also, we value to try these models as used by the community.

[1] https://github.com/yzhangcs/parser
[2] https://github.com/danifg/SyntacticPointer
[3] https://github.com/mstrise/dep2label

4.1.1 Experiment setup

The parsers are trained 3 times for each treebank, to then take the average as the final result.[4,5]

Input For all parsers, the embedding for each word is composed of a pre-trained word vector, a character-based vector and a PoS tag vector. For the word vectors, we use `fastText` (Bojanowski et al., 2017).[6] For PoS tags, we considered experiments both with gold and predicted PoS tags - using UDpipe (Straka et al., 2016)[7].

4.2 Datasets

Now, we review subsets that have been proposed, and summarize the criteria used to create them. While the subsets were defined on different versions of UD depending on the moment in which they were proposed, we use UD v2.8 for comparability. In case different treebanks are available for a given language and the authors did not specify which one they used for any reason (e.g. because in previous UD versions there was only one treebank, and therefore it was not necessary to name it), we chose the largest freely-available one. For space reasons, we include the specific treebanks for each subset in Appendix B (Table 10).

1. Ma et al. (2018) subset (Ma18): It has been widely adopted (Fernández-González and Gómez-Rodríguez, 2019; Li et al., 2020; Yang and Tu, 2021, *inter alia*), but it presents two weaknesses: (i) a high presence of Indo-European treebanks, ignoring diverse typologies, and (ii) as reported in their paper, all these treebanks are *easy*[8] treebanks.

2. de Lhoneux and Nivre (2016); de Lhoneux et al. (2017b) subset (Lh16): They were the first to address the problem of selecting a diverse sample of UD treebanks, establishing the following requirements: (i) include only one treebank from coarse-grained language families, (ii) include treebanks with certain morphological particularities, (iii) ensure different amounts of data, and (iv) include at least a highly non-projective treebank.

3. Schluter and Agić (2017) subset (SA17): Rather than manually choosing treebanks, this subset was chosen by an empirical method based on using delexicalized parsing performance to construct a similarity network, cluster it, and take one representative of each cluster. They concluded that their subset overestimates performance, while that of de Lhoneux and Nivre (2016) underestimates it.

4. Smith et al. (2018) subset (Sm18): The selection criteria for this subset were inspired in the criteria of de Lhoneux and Nivre (2016), but in this case aiming to be representative of different writing systems, character set sizes, and morphological complexity.

5. Kulmizev et al. (2019) subset (Ku19): The authors selected 13 treebanks, inspired in the criteria by de Lhoneux and Nivre (2016) and Smith et al. (2018). Apart from script, character set size and morphological complexity, they also aimed to have a representation of different training sizes and domains, and selected treebanks with good annotation quality.

6. Anderson and Gómez-Rodríguez (2020a) subset (AG20): Highly inspired by de Lhoneux et al. (2017b), but with a few changes. First, they exchanged $\text{Kazakh}_{\text{KTB}}$ for $\text{Uyghur}_{\text{UDT}}$, as Kazakh lacked an official development set. Second, they exchanged $\text{Ancient Greek}_{\text{PROIEL}}$ for $\text{Ancient Greek}_{\text{Perseus}}$, since it's more non-projective. Third, $\text{Czech}_{\text{PDT}}$ is swapped with $\text{Russian}_{\text{GSD}}$, as the Czech treebank took too long to train. Finally, they included $\text{Wolof}_{\text{WTB}}$ since African languages were not present. We included it to see if partial and justified changes over a diverse treebank subset could still lead to non-negligible changes.

7. Dehouck et al. (2021) subset (D21): This subset is dedicated to *true* data scarce treebanks. In the case of treebanks without a dev file, the

[4] Some treebanks ($\text{Kazakh}_{\text{KTB}}$, $\text{Galician}_{\text{TreeGal}}$ and $\text{Old-East-Slavic}_{\text{RNC}}$) do not have an official dev set, so we used 20% of the training set as the development set. Also, due to hardware limitations, the longest sentence (682 tokens) in the test file for the Old East Slavic (RNC) language was removed, since `syntacticpointer` ran out of memory during evaluation.

[5] The tools used for the experiments are described at this repository: https://github.com/MinionAttack/fragility_coling_2022

[6] We use `fastText` vectors except for some language treebanks that lacked embeddings. Particularly, for Ancient Greek we use UD embeddings, and for Wolof we used random initialized embeddings according to a uniform distribution in the range $[\frac{-1}{2\times300}, \frac{1}{2\times300}]$ (Goldberg, 2017).

[7] For Kazakh, Old East Slavic and Welsh there are no UDPipe models, so we only include their results with gold tags.

[8] We use *easy* in an informal sense, referring to treebanks where parsers obtain a higher performance. In *no way* we relate this term with a language being easier than other.

Set	LAS			$\mathcal{E}$-LAS			UAS			$\mathcal{E}$-UAS		
	gb-DM17	tb-FG19	sl-S+19	(tb-FG19, gb-DM17)	(sl-S+19, gb-DM17)	(sl-S+19, tb-FG19)	gb-DM17	tb-FG19	sl-S+19	(tb-FG19, gb-DM17)	(sl-S+19, gb-DM17)	(sl-S+19, tb-FG19)
Ma18	87.74	87.77	83.96	-0.14	23.93	23.93	91.07	91.13	87.68	-0.33	27.98	28.16
Lh16	80.33	79.68	74.20	1.83	24.04	22.49	85.03	84.45	79.90	2.21	26.33	24.51
SA17	84.85	84.97	80.30	-1.03	22.97	23.48	89.11	89.25	85.22	-1.46	26.76	27.64
Sm18	83.78	83.61	78.55	1.11	24.21	23.35	87.38	87.31	83.19	0.45	25.12	24.82
Ku19	83.36	83.08	77.98	-0.29	24.79	24.50	87.43	87.03	83.19	0.94	24.89	23.72
AG20	76.14	75.26	69.36	3.01	23.01	20.48	82.49	81.69	76.83	3.83	25.04	21.95
D21	59.00	57.04	51.38	4.57	17.00	12.97	68.60	67.38	62.96	3.94	16.18	12.92
Easy	89.59	89.65	85.88	-0.63	25.90	26.42	92.42	92.55	89.33	-1.58	28.61	29.85

Table 1: Average LAS and UAS scores for each subset in the predicted PoS tags setup. $\mathcal{E}(M1, M2)$ stands for error reduction between two models, where M1 is the reference system.

Set	LAS			$\mathcal{E}$-LAS			UAS			$\mathcal{E}$-UAS		
	gb-DM17	tb-FG19	sl-S+19	(tb-FG19, gb-DM17)	(sl-S+19, gb-DM17)	(sl-S+19, tb-FG19)	gb-DM17	tb-FG19	sl-S+19	(tb-FG19, gb-DM17)	(sl-S+19, gb-DM17)	(sl-S+19, tb-FG19)
Ma18	90.51	90.06	88.29	4.62	19.29	15.45	93.10	92.77	91.00	4.70	23.59	19.89
Lh16	78.89	76.71	74.44	7.20	19.94	13.55	84.30	83.13	80.84	6.10	21.24	16.13
SA17	85.52	84.57	81.92	5.71	20.79	15.89	89.52	88.83	86.65	5.32	23.40	19.00
Sm18	87.42	86.74	83.01	4.89	25.63	21.86	89.89	89.36	86.07	4.82	26.93	23.32
Ku19	87.18	86.14	82.99	6.22	24.65	19.58	89.82	89.04	86.18	5.87	26.13	21.52
AG20	81.04	79.54	77.53	7.17	14.08	7.35	85.77	84.82	82.72	6.46	16.26	10.50
D21	67.99	63.74	67.30	11.71	4.14	-8.82	75.26	72.61	75.52	9.92	2.14	-8.73
Easy	92.62	92.10	89.50	6.71	29.81	24.84	94.75	94.41	92.19	6.13	32.76	28.41

Table 2: Average LAS and UAS scores for each subset in the gold PoS tags setup.

training file was split in two, with a ratio of 80-20 for the training file and the dev file.

8. Easy subset: We propose an explicit *easy* subset to compare against other easy ones (e.g. Ma18). We used the results from the CoNLL 2018 Shared Task[9], and chose the 10 treebanks with the best LAS (no repeated languages). We list them in Appendix B.

4.3 Results

Table 1 shows the macro-average LAS (Labeled Attachment Score) and UAS (Unlabeled Attachment Score) results, using predicted PoS tags, for each subset, i.e. *the subset*, and not the treebank, is considered as *the atomic unit* for evaluation. For informative purposes, Table 2 shows the equivalent evaluation with gold PoS tags, but we will focus on the results with predicted PoS tags, unless stated otherwise. We also show error reduction ratios on LAS and UAS between parsers. This metric provides a better picture of differences between parsers than absolute LAS/UAS differences would, as it is less affected by treebank difficulty differences (e.g., it is much harder to achieve a given absolute LAS and UAS difference on easy treebanks than on more difficult ones, due to less available room for improvement and diminishing returns). The error reduction shown for each subset is calculated by first computing the error reduction for each treebank in the subset, and then averaging these error reductions (rather than by averaging the LAS/UAS for each treebank in the subset, and computing a single error reduction on that average). While this choice can cause some superficially counterintuitive phenomena like a parser having more average LAS than another but negative LAS error reduction (this happens with `tb-FG19` and `gb-DM17` on Ku19 on Table 1), it provides the desired semantics: for example, if a parser improves LAS from 98% to 99% in one treebank and from 50% to 90% in another, on average it is removing 65% of errors (50% of the errors in the first corpus, 80% in the second) and not 78.8% which we would obtain if we computed error reduction on average LAS.

Next, we discuss factors that seem to play a role in the subset performance.

Influence of parsing difficulty From the results, easier subsets tend to correspond to larger error reductions when comparing the (state-of-the-art) parsers `gb-DM17` and `tb-FG19` with respect to `sl-S+19` (the control parser). This is most evident for the Easy subset: all parsers obtain their best performance across all subsets, and the error reductions with respect to the control parser are also the largest, for all setups. The opposite happens with the D21 subset, the hardest one. In this context, when optimizing for other dimensions than performance, such as speed, training efficiency or architectural simplicity, relying (exclusively) on easy treebanks could thus be a sub-optimal strategy. The sense of the decrease in performance could

[9] https://universaldependencies.org/conll18/results-las.html

be larger on these easy datasets than when evaluating on random treebanks, or on more difficult cases as suggested by the results on the subsets of Lh16, D21, or AG20 to a lesser extent. On the contrary, dimensions such as the ones mentioned above are often not expected to benefit more from the particularities of easy treebanks. Also, there are trends related to parsing difficulty between the state-of-the-art parsers `gb-DM17` and `tb-FG19`: `gb-DM17` seems to be superior to `tb-FG19` when the subset becomes harder to parse, and *vice versa*.

Differences on representative subsets While both the Lh16 and the SA17 subsets were designed to enforce representativity, the ranking of the tested parsers changes: `tb-FG19` performs better (in LAS error reduction terms) than `gb-DM17` on SA17 (automatically picked) and Ku19 (manually constructed), but worse on the (manually constructed) subsets of Lh16, Sm18 and AG20. This highlights that even when treebanks are sampled with attention to representativity, results can still show instability - be it due to different possible notions of representativity, or statistical variation.

Developing and testing on the same treebanks While there is no clear performance difference between `gb-DM17` and `tb-FG19`, as each of them surpasses the other in some subsets; one of the subsets where `tb-FG19` takes the lead is Ma18, where that parser was developed and reported its results. This leads to the question whether developing and evaluating on a given subset of treebanks could induce bias in favor of those treebanks. While the available data is not enough to give an answer in this specific instance, we can draw similar conclusions either way. If this were the case, it would mean that in the context of multilingual, language-agnostic parsers, and when data for a wide range of languages is available, it would be advisable to go beyond separating development and test sets for each language or treebank, and instead use different languages for development than for evaluation to avoid this kind of bias. Conversely, if this were not the case, it would mean that we could choose one of the human-defined subsets and obtain state-of-the-art results for one parser or the other, purely by chance. This makes us reflect about using a single subset of treebanks to justify the superior performance of a model, and might again make advisable to develop and test on different subsets - to reduce the element of chance.

Experimentation with data scarcity For the D21 subset, centered exclusively on extremely low-resource treebanks, the error reduction computed between the best performing parser (`gb-DM17`) and the control parser (`sl-S+19`) is the lowest among all tested treebanks. As mentioned above, the opposite happens for the easiest subsets. Yet, we feel these type of subsets would not be optimal either for evaluating parsers in a general sense, as they might not capture how a given parser can fully exploit its learning capabilities. Overall, the evaluation on this setup seems more volatile. We see a few differences between the predicted and gold PoS tags setups, causing even changes in the parsing ranking. For instance, `sl-S+19` outperforms `tb-FG19` in the gold setup by a clear margin, an issue that does not arise in any other subset.

5 Experiment 2: *many* parsers, *many* randomized subsets

This experiment can be seen as a re-definition of Experiment 1 at a large scale. Above, we compared a few competitive parsers only on a handful of subsets of treebanks that were human-defined, and observed different trends. Yet, this is a limited view of the problem. If we take as reference the CoNLLU 2018 Shared Task (Zeman et al., 2018) and the 82 treebanks that were evaluated, considering subsets of size 10 (meaning that each is composed of 10 different treebanks), we would obtain up to $\sim 2.13 \times 10^{12}$ possible combinations. Many of those subsets will not be a representative sample of languages, but we already saw that there are subsets that are used in parsing as a benchmark that are not either, and that even when they are considered representative, the criteria varies, and the parsing performance, differences among parsers and error reductions vary too. Here, we generate subsets, similar in size to typical human-defined ones, and see how subset differences affect parsing rankings.

Similarly to Experiment 1, we do not analyze here algorithms and parsing architectures, or their correctness, but the appropriateness of evaluation procedures. In this context, some shared-task systems have reported bugs in their pipeline: this was mostly evident for some systems that consistently ranked in the last positions (see Table 3), but not so noticeable for high-scoring ones, such as the Stanford system (Qi et al., 2018), which later reported a preprocessing bug that affected the low-resource treebanks more. Thus, multi-treebank evaluation

Parser	Best rank	Worst rank	μ	$\tilde{\mu}$	σ
HIT-SCIR	1	6	1.14	1.00	0.54
UDPipe Future	1	12	4.38	4.00	1.65
TurkuNLP	1	12	3.97	4.00	1.53
LATTICE	1	13	4.99	5.00	2.27
ICS PAS	2	13	4.71	5.00	2.01
CEA LIST	1	13	6.12	6.00	2.28
Stanford	1	21	6.30	6.00	3.47
Uppsala	1	13	6.62	7.00	2.36
NLP-Cube	2	20	10.02	10.00	1.79
AntNLP	3	18	9.94	10.00	1.82
ParisNLP	4	20	10.39	11.00	1.62
SLT-Interactions	2	23	11.28	11.00	3.91
IBM NY	2	20	13.05	13.00	1.64
LeisureX	8	20	14.36	14.00	1.81
UniMelb	8	19	13.80	14.00	1.13
KParse	9	22	16.78	17.00	1.35
Fudan	10	22	17.16	17.00	1.60
BASELINE UDPipe	13	22	18.08	18.00	1.11
Phoenix	13	22	18.69	19.00	1.07
CUNI x-ling	2	22	19.24	20.00	2.21
BOUN	16	23	20.81	21.00	0.79
ONLP lab	20	25	22.57	23.00	0.62
iParse	9	25	22.36	23.00	2.76
HUJI	21	25	23.63	24.00	0.89
ArmParser	22	25	24.62	25.00	0.59
SParse	26	26	26.00	26.00	0.00

Table 3: Ranking stats for LAS and the parsers of the Zeman et al. (2018) Shared task, over the 1 million random subsets. Table sorted by $\tilde{\mu}$ (the median).

Parser	Best rank	Worst rank	μ	$\tilde{\mu}$	σ
TurkuNLP	1	7	1.51	1.00	0.78
HIT-SCIR	1	6	2.37	2.00	1.13
ICS PAS	1	13	3.83	4.00	1.37
UDPipe Future	1	8	3.62	4.00	0.98
Stanford	1	15	4.21	4.00	1.75
LATTICE	1	11	6.23	6.00	1.02
CEA LIST	2	12	6.58	6.00	0.99
ParisNLP	5	16	8.87	9.00	0.91
AntNLP	3	14	8.59	9.00	0.95
SLT-Interactions	2	20	10.09	10.00	2.25
LeisureX	7	18	11.54	11.00	1.21
UniMelb	7	16	11.27	11.00	0.82
BASELINE UDPipe	10	20	14.53	14.00	1.09
NLP-Cube	6	22	15.07	14.00	2.81
Phoenix	10	20	14.89	15.00	1.15
KParse	9	21	15.54	16.00	1.95
CUNI x-ling	4	21	17.43	18.00	1.60
BOUN	12	22	18.22	18.00	1.33
Fudan	9	22	17.34	18.00	1.94
iParse	9	25	19.12	20.00	3.19
HUJI	15	25	20.45	21.00	1.02
ArmParser	19	25	22.10	22.00	0.95
Uppsala	19	25	23.29	23.00	0.66
IBM NY	16	25	23.44	24.00	0.82
ONLP lab	22	25	24.88	25.00	0.37
SParse	26	26	26.00	26.00	0.00

Table 4: Ranking statistics for BLEX and the parsers of the Zeman et al. (2018) Shared task, over 1 million randomly generated subsets. Table sorted by $\tilde{\mu}$.

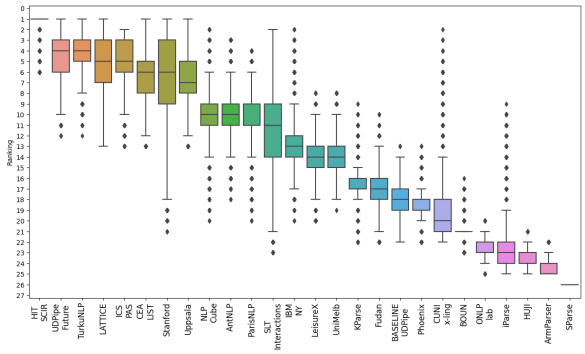

Figure 1: Corresponding box plot for Table 3. For an easy correspondence with the table, the x-axis (from left to right) is sorted as the Table is (i.e. by $\tilde{\mu}$).

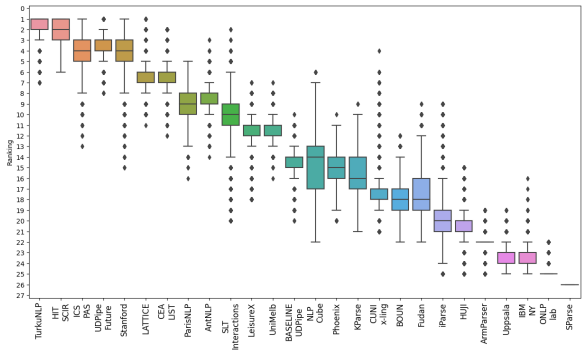

Figure 2: Corresponding box plot for Table 4. For an easy correspondence with the table, the x-axis (from left to right) is sorted as the table is (i.e. by $\tilde{\mu}$).

procedures should also be robust for systems suffering bugs that affect treebanks differently.

5.1 Experimental setup

We compare the available results of the 26 parsers[10] that participated in the CoNLLU Shared Task (Zeman et al., 2018), sampling random subsets over the 82 evaluated treebanks. We generate 1 million random subsets made of size 10,[11] and we do not control the subsets' content (e.g. subsets with higher presence of a language or family).

[10]The parsers are not necessarily a diverse sample of parsing models (many are based on `gb-DM17`) but they are a realistic sample of a ranking of parsers made in a real shared task. For model representativity, we refer the reader to Experiment 1.

[11]The subset size is in the range of those of Experiment 1.

5.2 Results

The pair Table 3 - Figure 1 shows statistics about the 26 parsers that participated in the shared task and the 1 million randomly generated subsets. Some (top) parsers show a stable performance. For instance, the HIT-SCIR parser (Che et al., 2018) mostly ranks at the first position, except for a few outliers that show that the parser potentially could go down as far as the 6th position. This tendency is also observed in a few other - and worse-performing - systems, such as Kparse (Kırnap et al., 2018) or Phoenix (Wu et al., 2018).

However, for many other parsers the variability is larger. The interquartile range of the UDpipe-Future system (2nd place) (Straka, 2018) is small (from 3rd to 6th), but its fourth quartile (excluding outliers) ranges between the 6th and the 10th posi-

Parser	Avg. rank	Best subset outliers		Worst subset outliers	
		Rank	Treebanks	Rank	Treebanks
UDPipe Future (Straka, 2018)	$4.38_{\pm 1.65}$	1	pl_{sz}, ga_{idt}, fro_{srcmf}, sr_{set}, kmr_{mg}, el_{gdt}, en_{ewt}, fr_{gsd}, th_{pud}, hy_{armtdp}	12	bxr_{bdt}, la_{ittb}, hsb_{ufal}, pcm_{nsc}, th_{pud}, ro_{rrt}, br_{keb}, $no_{bokmaal}$, sl_{ssj}, sv_{pud}
Stanford (Qi et al., 2018)	$6.30_{\pm 3.47}$	1	de_{gsd}, cs_{cac}, el_{gdt}, he_{htb}, es_{ancora}, sv_{lines}, ja_{modern}, bg_{btb}, $no_{bokmaal}$, cs_{pud}	21	hsb_{ufal}, pt_{bosque}, ga_{idt}, vi_{vtb}, ko_{kaist}, $sv_{talbanken}$, $gl_{treegal}$, sme_{giella}, hy_{armtdp}, th_{pud}
SLT-Interactions (Bhat et al., 2018)	$11.28_{\pm 3.91}$	2	sk_{snk}, hsb_{ufal}, uk_{iu}, $cs_{fictree}$, sv_{pud}, ja_{gsd}, pt_{bosque}, hr_{set}, fa_{seraji}, sl_{ssj}	23	hr_{set}, fa_{seraji}, pl_{lfg}, kmr_{mg}, de_{gsd}, hy_{armtdp}, nl_{alpino}, th_{pud}, sme_{giella}, fro_{srcmf}

Table 5: Qualitative results of subsets that cause anomalous rankings of parsers in Experiment 2.

tion. The situation is almost identical for the next 4 averaged best performing systems. Across the board, there are even more severe examples, such as Stanford (Qi et al., 2018) (7th place), whose interquartile range spans from the 3rd to the 9th position; or the SLT-Interactions parser (Bhat et al., 2018) whose first quartile ranges from 2nd to 9th, while its fourth quartile ranges from 14th to 21th. Exemplifying it with the Stanford system, we will also discuss below how randomized multi-treebank evaluation would have been useful to detect the anomalous performance on low-resource treebanks, that later on turned out to be a bug, or on the other hand how a weak subset selection could cause potential anomalous performances or bugs to go unnoticed.

Parser	#Outliers	Avg. size	$\mathcal{R}$(lr)	$\mathcal{R}$(Slavic)
UDPipe Future	10 best	168.87	0.30	0.17
	10 worst	157.51	0.39	0.21
Stanford	10 best	221.03	0.19	0.23
	10 worst	124.44	0.61	0.18
SLT-Interactions	10 best	203.92	0.30	0.36
	10 worst	146.72	0.39	0.14

Table 6: Quantitative study expanding Table 5. $\mathcal{R}$ refers to the average ratio across subsets of the presence of low-resource (lr) and Slavic treebanks.

Subsets that cause anomalous results Table 5 shows a few examples of parsers and subsets that caused atypical results. For each parser, we show an advantageous and a disadvantageous subset, randomly picked among those for which a parser obtained its best and worst rankings.

A qualitative analysis of these results yields several insights. For the first two parsers, the advantageous and disadvantageous outliers are linguistically diverse, but there is a clear trend that the disadvantageous subsets are heavily biased towards small treebanks: for both of the parsers, the favorable subset contains only 2 treebanks that are low-resource according to the shared task criteria, while the disfavorable one contains 6 and 5, respectively. This is very unlikely to happen by chance: the probability of randomly drawing a subset with 6 or more low-resource treebanks is 0.00214, and with 5 or more, 0.01538 (this is calculated from a hypergeometric distribution with parameters $N = 82$, the total number of treebanks, $K = 21$, the number of low-resource treebanks, and $n = 10$, the number of treebanks per subset). Thus, this variability seems to owe to the fact that the UDPipe Future (Straka, 2018) and Stanford (Qi et al., 2018) parsers struggle (relatively to competitors) when training data is scarce. The situation is different for the third parser considered. In this case, there are no substantial differences in treebank size (2 vs. 3 low-resource treebanks) but instead there is a clear linguistic pattern: the advantageous subset has a heavy bias towards Slavic languages (6 out of the 10 languages are Slavic, compared to 2 in the disadvantageous subset - and the probability of choosing a subset with 6 or more Slavic languages by chance is 0.00043, from a hypergeometric distribution with parameters $N = 82, K = 17, n = 10$). This seems to reflect that the SLT-Interactions parser (Bhat et al., 2018) is especially adequate for Slavic languages. It is worth noting that the authors did not implement any language-specific adaptation or report anything in the paper that suggests that they specifically addressed these languages, so this serves as an example that a parser can show linguistic biases towards certain language families even if it has been developed in a language-agnostic way.

We propose a complementary quantitative analysis in Table 6. We randomly take 10 of the best and worst performing subsets for the above studied parsers and compute, across subsets, the average size of treebanks, the presence of low-resource treebanks, and the presence of Slavic languages. The analysis confirms the bias towards rich-resource treebanks for the Stanford parser, and towards Slavic languages for SLT-Interactions (while not being biased towards rich- or low-resouce treebanks). On the other hand, the hypothesis of UDpipe Future being biased towards rich-resource languages is not clearly confirmed by this analysis.

To sum up, this reinforces the idea (hypothesized in papers like de Lhoneux and Nivre (2016)) that

both treebank sizes and linguistic factors are important for a treebank subset to be representative; and highlight that the latter can have a huge influence even in parsers that have been developed without specific language families in mind.

More robust metrics? LAS and UAS are the most popular metrics to report dependency parsing performance. Yet, there are other metrics, such as CLAS[12], MLAS[13] or BLEX[14], but they have not been widely adopted (maybe because they have a not so straightforward interpretation). Yet, from Experiment 2 we observed that some of these metrics, especially BLEX, produced narrower standard deviations and more stable rankings. We leave interpretations of this phenomenon as an open question for future work, but refer the reader to (Table 4, Figure 2) and (Table 3, Figure 1), which show a summary of the ranking statistics for the BLEX and LAS metrics, respectively, on the 26 parsers that participated in the ConLLU Shared task 2018 (Zeman et al., 2018). Overall, but especially for the top parsers, BLEX results produce more stable rankings and narrower interquartile ranges.

6 Discussion

We have designed two experiments that revealed issues of relying on a single subset of treebanks for parsing evaluation. More particularly, we have shown that: (i) existing human-defined subsets show high variability in terms of rankings and performance across parsers, (ii) parsers that have been developed on a concrete subset might be biased towards performing better on that subset, (iii) it is relatively easy to come up with subsets that generate different parsing rankings, (iv) this can even happen across subsets that have been purposefully defined to be representative, (v) both linguistic typology and resource size have a large influence in the variability of results between parsers, and (vi) linguistic factors can be crucial even when parsers are designed in a language-agnostic way.

Overall, some advice can be given: (a) claims that "parser X is more accurate than parser Y" can be weak even on carefully selected samples of UD treebanks (and perhaps it is recommendable to consider metrics that take into account dimensions such as speed and efficiency), (b) for language-agnostic parsers, it is worth noting that there can still be biases towards certain linguistic families, and (c) for such parsers, it can be advisable to develop on one set of treebanks and evaluate on another, to avoid bias in favor of the languages used for development.

Finally, there are aspects that we did not study in this piece of work, but that could affect the robustness of parsing evaluation as well, e.g., automatically *versus* manually annotated treebanks, and interactions between language and treebank properties (e.g. morphological complexity, dependency distance, ...) and parsing models.

7 Conclusion

Different subsets of treebanks have been proposed to try to capture the essence of the whole set of UD treebanks, so that the performance of parsers in such subsets would be representative of that obtained in the full set. We have empirically shown limitations of this approach, and also how establishing guidelines for good treebank selection can be hard, although some bad practices can be avoided.

Acknowledgments

This work was supported by a 2020 Leonardo Grant for Researchers and Cultural Creators from the FBBVA,[15] as well as by the European Research Council (ERC), under the European Union's Horizon 2020 research and innovation programme (FASTPARSE, grant agreement No 714150). The work is also supported by ERDF/MICINN-AEI (SCANNER-UDC, PID2020-113230RB-C21), by Xunta de Galicia (ED431C 2020/11), and by Centro de Investigación de Galicia "CITIC" which is funded by Xunta de Galicia, Spain and the European Union (ERDF - Galicia 2014–2020 Program), by grant ED431G 2019/01.

References

Mark Anderson and Carlos Gómez-Rodríguez. 2020a. 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.

[12] CLAS: It ignores selected relations which attach function words to content words.

[13] MLAS: It is inspired by the CLAS metric, and extended with evaluation of POS tags and morphological features.

[14] BLEX (bi-lexical dependency score): it combines content-word relations with lemmatization.

[15] 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.

Mark Anderson and Carlos Gómez-Rodríguez. 2020b. Inherent dependency displacement bias of transition-based algorithms. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5147–5155, Marseille, France. European Language Resources Association.

Mark Anderson and Carlos Gómez-Rodríguez. 2021. A modest Pareto optimisation analysis of dependency parsers in 2021. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 119–130, Online. Association for Computational Linguistics.

Mark Anderson, Anders Søgaard, and Carlos Gómez-Rodríguez. 2021. Replicating and extending "Because their treebanks leak": Graph isomorphism, covariants, and parser performance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1090–1098, Online. Association for Computational Linguistics.

Riyaz A. Bhat, Irshad Bhat, and Srinivas Bangalore. 2018. The SLT-interactions parsing system at the CoNLL 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 153–159, Brussels, Belgium. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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

Raphael Cohen, Yoav Goldberg, and Michael Elhadad. 2012. Domain adaptation of a dependency parser with a class-class selectional preference model. In *Proceedings of ACL 2012 Student Research Workshop*, pages 43–48, Jeju Island, Korea. Association for Computational Linguistics.

Anna Corazza, Alberto Lavelli, and Giorgio Satta. 2013. An information-theoretic measure to evaluate parsing difficulty across treebanks. *ACM Trans. Speech Lang. Process.*, 9(4).

Miryam de Lhoneux and Joakim Nivre. 2016. Ud treebank sampling for comparative parser evaluation. In *Proc. of the Sixth Swedish Language Technology Conference (SLTC)*.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017a. Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement. In *TLT*, pages 99–110.

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement. In *TLT*.

Mathieu Dehouck, Mark Anderson, and Carlos Gómez-Rodríguez. 2021. A falta de pan, buenas son tortas: The efficacy of predicted upos tags for low resource ud parsing. *arXiv preprint arXiv:2106.04222*.

Mathieu Dehouck and Pascal Denis. 2019. Phylogenic multi-lingual dependency parsing. 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 192–203, 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.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João Graça, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1051–1055, Prague, Czech Republic. 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.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309.

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.

Kyle Gorman and Steven Bedrick. 2019. We need to talk about standard splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational*

Linguistics, pages 2786–2791, Florence, Italy. Association for Computational Linguistics.

Kristina Gulordava and Paola Merlo. 2016. Multilingual dependency parsing evaluation: a large-scale analysis of word order properties using artificial data. *Transactions of the Association for Computational Linguistics*, 4:343–356.

Daniel Hershcovich, Stella Frank, Heather Lent, Miryam de Lhoneux, Mostafa Abdou, Stephanie Brandl, Emanuele Bugliarello, Laura Cabello Piqueras, Ilias Chalkidis, Ruixiang Cui, Constanza Fierro, Katerina Margatina, Phillip Rust, and Anders Søgaard. 2022. Challenges and strategies in cross-cultural NLP. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6997–7013, Dublin, Ireland. Association for Computational Linguistics.

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

Ömer Kırnap, Erenay Dayanık, and Deniz Yuret. 2018. Tree-stack LSTM in transition based dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 124–132, Brussels, Belgium. Association for Computational Linguistics.

Sandra Kübler, Wolfgang Maier, Ines Rehbein, and Yannick Versley. 2008. How to compare treebanks. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Marco Kuhlmann and Joakim Nivre. 2010. Transition-based techniques for non-projective dependency parsing. *Northern European Journal of Language Technology*, 2(1):1–19.

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. 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 2755–2768, Hong Kong, China. 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, Paris, France. Association for Computational Linguistics.

Zuchao Li, Hai Zhao, and Kevin Parnow. 2020. Global greedy dependency parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8319–8326.

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.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131, Prague, Czech Republic. Association for Computational Linguistics.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

Alberto Muñoz-Ortiz, Michalina Strzyz, and David Vilares. 2021. Not all linearizations are equally data-hungry in sequence labeling parsing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 978–988, Held Online. INCOMA Ltd.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 915–932, Prague, Czech Republic. Association for Computational Linguistics.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal Dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Ines Rehbein, Julius Steen, Bich-Ngoc Do, and Anette Frank. 2017. Universal Dependencies are hard to parse – or are they? In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pages 218–228, Pisa,Italy. Linköping University Electronic Press.

Rudolf Rosa. 2015. Multi-source cross-lingual delexicalized parser transfer: Prague or Stanford? In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 281–290, Uppsala, Sweden. Uppsala University, Uppsala, Sweden.

Natalie Schluter and Željko Agić. 2017. Empirically sampling Universal Dependencies. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 117–122, Gothenburg, Sweden. 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, Brussels, Belgium. Association for Computational Linguistics.

Anders Søgaard. 2020. Some languages seem easier to parse because their treebanks leak. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2765–2770, Online. Association for Computational Linguistics.

Anders Søgaard, Sebastian Ebert, Jasmijn Bastings, and Katja Filippova. 2021. We need to talk about random splits. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1823–1832, Online. 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.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, Portorož, Slovenia. European Language Resources Association (ELRA).

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.

Reut Tsarfaty, Djamé Seddah, Sandra Kübler, and Joakim Nivre. 2013. Parsing morphologically rich languages: Introduction to the special issue. *Computational Linguistics*, 39(1):15–22.

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.

Guillaume Wisniewski and François Yvon. 2019. How Bad are PoS Tagger in Cross-Corpora Settings? Evaluating Annotation Divergence in the UD Project. 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 218–227, Minneapolis, Minnesota. Association for Computational Linguistics.

Yingting Wu, Hai Zhao, and Jia-Jun Tong. 2018. Multilingual Universal Dependency parsing from raw text with low-resource language enhancement. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 74–80, Brussels, Belgium. Association for Computational Linguistics.

Songlin Yang and Kewei Tu. 2021. Headed span-based projective dependency parsing. *arXiv preprint arXiv:2108.04750*.

Daniel Zeman, Jan Hajič, 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. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. 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.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Ma-

nurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

A Experiment 1: models, resources, and hyperparameters

To train the models, we used 2 NVIDIA GeForce RTX 2080 Ti@11GB and an Intel® Core™ i7-9700K@3.60GHz×8. Training times usually took from 1 to 7 hours, depending on the parsing model and the treebank training size. The three used parsers and the UD treebanks have free software licenses that allow free use and distribution.

Tables 7, 8 and 9 show the hyperparameters used for the `gb-DM17` (Dozat et al., 2017) (using the `supar` software package), `tb-FG19` (Fernández-González and Gómez-Rodríguez, 2019) (using the `syntacticpointer` package) and `sl-S+19` parsers (Strzyz et al., 2019) (using the `dep2labels` package), respectively.

Hyperparameter	Value	Hyperparameter	Value
n_char_hidden	100	ν	.9
n_feat_embed	100	ϵ	1^{-12}
embed_dropout	.33	weight_decay	0
n_lstm_hidden	400	clip	5.0
n_lstm_layers	3	min_freq	2
encoder_dropout	.33	fix_len	20
n_arc_mlp	500	decay	.75
n_rel_mlp	100	decay_steps	5000
mlp_dropout	.33	update_steps	1
encoder	lstm	feats	['tag', 'char']

Table 7: Hyperparameters used to train the `supar` models. In the case of Ancient Greek the hyperparameter *n_embed* is 100.

Hyperparameter	Value	Hyperparameter	Value
model	L2RPtr	–learning_rate	0.001
word_dim	300	–lr_decay	0.999997
char_dim	100	–beta1	0.9
pos	true	–beta2	0.9
rnn_mode	FastLSTM	–grad_clip	5.0
encoder_layers	3	–loss_type	token
decoder_layers	1	–warmup_steps	40
hidden_size	512	–reset	20
arc_space	512	–weight_decay	0.0
type_space	128	–unk_replace	0.5
p_in	0.33	–beam	5
p_out	0.33	–char_embedding	random
p_rnn	[0.33, 0.33]	–opt	adam
prior_order	inside_out	–batch_size	32
grandPar	false	–num_epochs	600
sibling	false		
activation	elu		

Table 8: Hyperparameters used to train the `syntacticpointer` models. Parameters specified from the configuration file on the left, and from the command line on the right.

Hyperparameter	Value
cnn_layer	4
char_hidden_dim	100
hidden_dim	800
dropout	0.5
lstm_layer	3
bilstm	True
learning_rate	0.02
lr_decay	0.05
momentum	0.9
l2	0
gpu	True

Table 9: Hyperparameters used to train the `dep2labels` models.

B Treebanks in each subset

In §4.2, we reviewed the related work and briefly discussed several human-defined subsets that were proposed in the past, according to a number of criteria, and that we used to report the results from our Experiment 1. Due to space reasons, we detail here in this appendix (Table 10) the specific treebanks that are part of each subset, and their sizes, for a better understanding of the particularities of each of them.

	Size	Ma18	Lh16	AG20	D21	SA17	Sm18	Ku19	Easy
Ancient Greek (PROIEL)	213K		✓				✓		
Ancient Greek (Perseus)	202K			✓					
Arabic (PADT)	282k						✓	✓	
Basque (BDT)	121K							✓	
Belarusian (HSE)	305K				✓				
Bulgarian (BTB)	156K	✓							✓
Catalan (AnCora)	546K	✓							✓
Chinese (GSD)	123K		✓	✓			✓	✓	
Coptic (Scriptorium)	48K					✓			
Czetch (FicTree)	167K								✓
Czetch (PDT)	1509K	✓	✓						
Dutch (Alpino)	208K	✓				✓			
English (EWT)	254K	✓	✓	✓			✓	✓	
Finnish (TDT)	202K		✓	✓			✓	✓	
French (GSD)	400K	✓							
Galician (TreeGal)	25K				✓				
German (GSD)	292K	✓							
Hebrew (HTB)	161K		✓	✓		✓	✓	✓	
Hindi (HDTB)	351K							✓	✓
Indonesian (GSD)	120K					✓			
Italian (ISDT)	298K	✓				✓		✓	✓
Japanese (GSD)	193K							✓	
Kazakh (KTB)	10K		✓						
Korean (GSD)	80K							✓	
Korean (Kaist)	350K						✓		
Lithuanian (HSE)	5K				✓				
Marathi (UFAL)	3K				✓				
Norwegian (Bokmaal)	310K	✓				✓			✓
Old Church Slavonic (PROIEL)	57K					✓			
Old East Slavic (RNC)	30K				✓				
Polish (LFG)	130K								✓
Polish (PDB)	350K					✓			
Romanian (RRT)	218K	✓							
Russian (GSD)	98K			✓					
Russian (SynTagRus)	1107K	✓					✓	✓	✓
Sanskrit (Vedic)	27K					✓			
Slovenian (SSJ)	140K								✓
Spanish (AnCora)	560K	✓							✓
Swedish (Talbanken)	96K						✓	✓	
Tamil (TTB)	9K		✓	✓	✓				
Turkish (IMST)	57K							✓	
Uyghur (UDT)	40K			✓					
Welsh (CCG)	36K				✓				
Wolof (WTB)	44K			✓					

Table 10: Treebanks per set