Assessment of Pre-Trained Models Across Languages and Grammars

Alberto Muñoz-Ortiz, 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
{alberto.munoz.ortiz, david.vilares, carlos.gomez}@udc.es

Abstract

We present an approach for assessing how multilingual large language models (LLMs) learn syntax in terms of multi-formalism syntactic structures. We aim to recover constituent and dependency structures by casting parsing as sequence labeling. To do so, we select a few LLMs and study them on 13 diverse UD treebanks for dependency parsing and 10 treebanks for constituent parsing. Our results show that: (i) the framework is consistent across encodings, (ii) pre-trained word vectors do not favor constituency representations of syntax over dependencies, (iii) sub-word tokenization is needed to represent syntax, in contrast to character-based models, and (iv) occurrence of a language in the pretraining data is more important than the amount of task data when recovering syntax from the word vectors.

1 Introduction

Large Language Models (LLMs) are the backbone for most NLP architectures. Their performance has not yet reached a plateau, and factors such as scale, language objective, token segmentation or amount of pre-training time - among many others - play a role in their capabilities.

To shed light on what is being learned, work on interpretability explains what these models encode in their representational space. Authors have explored whether these models exhibit stereotypical biases (Nadeem et al., 2021), encode facts (Poerner et al., 2020) or capture structural knowledge in multi-modal environments (Milewski et al., 2022). Whether LLMs encode syntaxin their latent space has also been studied. In this respect, different *probing frameworks* (Kulmizev and Nivre, 2022; Belinkov, 2022) have been introduced to measure the syntactic capability of models, although authors such as Maudslay and Cotterell (2021) point out that we need to take this concept with caution, since they might not be completely isolating syntax.

Still, interpretability work on parsing focuses on either multilingual and mono-paradigm setups, or English and multi-paradigm setups. But we are not aware of *multi-dimensional* work. This relates to the problem of square one bias in NLP research (Ruder et al., 2022), that states that most work expands the current knowledge along just one dimension (e.g., a single language, or a single task). Related to our work, Kulmizev et al. (2020) study if LLMs showed preferences across two annotation *styles*: deep syntactic and surface-syntactic universal dependencies, but both schemes were dependency-based. Vilares et al. (2020) did study two different syntactic formalisms, dependencies and constituents, and used a sequence-labeling-like recovery framework, relying on the pretraining architectures to associate output vectors with syntactic labels. We will build on top of this framework. Yet, they only studied English, and their analysis focused on static vectors and early LLMs; apart from other limitations that we discuss later.

Contribution We move from square one bias in syntax assessment, and propose the first multi-paradigm, multilingual, recovery framework for dependency and constituent structures learned by LLMs. We select representative LLMs that vary in scale, language pretraining objectives, and token representation formats. We then study their capability to retrieve syntax information from the pretrained representations on a diverse set of constituent and dependency treebanks, that vary in factors such as language family or size, as well as the presence or absence of their languages among the pretraining data of the LLMs. The code is available at https://github.com/amunozo/multilingual-assessment.

2 Related work

There is a long-standing effort in the NLP community to model syntax, either as a final goal or as

Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 359–373
November 1–4, 2023.

a way to model compositionality. Yet, the ways in which this has been pursued have evolved with time.

Modeling syntax in the pre-neural times. Learning grammars through corpus-based approaches (Marcus et al., 1993; Collins, 1996; Charniak, 1997; Petrov and Klein, 2007) has been the dominating approach in the last decades. However, early models required extensive feature engineering to obtain competitive parsers. This suggested that support vector machines (SVMs) had severe limitations understanding language structure, and needed the help of parsing algorithms (Nivre, 2008; Martins et al., 2010), language-dependent features (Ballesteros and Nivre, 2012), or tree-kernels (Lin et al., 2014; Zhang and Li, 2009) to model syntax properly.

Modeling syntax in neural times. With the rise of word vectors (Mikolov et al., 2013), LSTMs (Hochreiter and Schmidhuber, 1997) and Transformers (Vaswani et al., 2017), modeling structure has become less relevant to obtain a good performance, both for parsing and downstream tasks. For instance, while the classic parser by Zhang and Nivre (2011) used a rich set of features (including third-order, distance, and valency features, among others) to be competitive, the parser by Chen and Manning (2014) only needed 18 word and PoS tag features (and 6 dependency features) to obtain strong results, which was possible thanks to their reliance on pre-trained word vectors and neural networks. The need for feature engineering was reduced further with bidirectional LSTMs, e.g., Kiperwasser and Goldberg (2016) showed that four vectors corresponding to elements in the buffer and the stack sufficed to obtain state-of-the-art performance, while Shi et al. (2017) showed that competitive accuracies were possible with only two features.

Modeling syntax in the era of language models. In the context of these (almost) end-to-end parsers performing very competitively without the need of explicitly modeling syntactic linguistic features, recent efforts have been dedicated to interpret to what extent syntax is encoded in the representational space of neural networks, and in particular of LLMs. Tenney et al. (2019) and Liu et al. (2019a) proposed probing frameworks for partial parsing, in the sense that they tried to demonstrate that certain syntactic information, such as dependency types, was encoded in pre-trained models. Vilares et al. (2020) defined a probing framework for full dependency and constituent parsing. They cast dependency and constituent parsing as sequence labeling and associated output vectors with syntactic labels by freezing their models. Hewitt and Manning (2019) proposed a structural probing framework and identified that pre-trained models encoded a linear transformation that indicates the distance between words in a dependency tree. The framework was later upgraded to extract directed and labeled trees, while using fewer parameters (Müller-Eberstein et al., 2022a). Hewitt and Liang (2019) pointed out that we need to be careful with probing frameworks, since the probe might be learning the linguistic task itself, instead of demonstrating the presence of the target linguistic property. For that, they recommend to use control experiments, and relied on control tasks, i.e., learning a random task with the same dimensional output space. Maudslay and Cotterell (2021) showed that semantic cues in the data might guide the probe and therefore they might not isolate syntax, although their experiments still outperformed the baselines. Müller-Eberstein et al. (2022b) found the most suitable pre-trained LLMs to plug into a dependency parser for a given treebank. Particularly, they proposed to rank frozen encoder representations by determining the percentage of trees that are recoverable from them, and based on that ranking choose which LLM to plug. Focused on morphology, Stanczak et al. (2022) showed that subsets of neurons model morphosyntax across a variety of languages in multilingual LLMs.

3 Multilingual probing frameworks

Let $w = [w_1, w_2, ..., w_n]$ be an input sentence. We are interested in linear probing frameworks that can associate a sequence of word vectors $\vec{w} = [\vec{w}_1, \vec{w}_2, ..., \vec{w}_n]$ to a given linguistic property $[p_1, p_2, ..., p_n]$. For some properties, the mapping can be quite direct, such as for instance the case of part-of-speech (PoS) tagging (by putting a linear layer on top of w and outputting the PoS tag category), or lexical semantics (e.g. computing word vector similarity). We want an analogous mapping, but for multiple syntactic formalisms. In this case, the association is not trivial since syntactic parsing is a tree-based structured prediction problem. Also, we are interested in multilingual pre-trained models, which have gained interest in recent years.

Then, the goal is to associate their word vectors to an estimate of to what extent characteristics of a given formalism are encoded in their representational space, and whether this can differ across dimensions such as tested models, formalisms, and treebanks.

Linear probing framework for parsing We take the study by Vilares et al. (2020) as our starting point. However, we first identify some weaknesses in their work: (i) it is limited to English, (ii) they do not give specific estimates of the amount of trees recoverable with respect to control experiments, and (iii) they only test one type of tree linearization. For the latter, the main motivation, in particular for the case of dependency parsing, was that the chosen linearization had performed the best in previous work (Strzyz et al., 2019) when training from scratch a transducer without pre-training. However, later work suggests that that is debatable: for instance, Muñoz-Ortiz et al. (2021) show that different tree linearizations might be better suited to different languages, and Vacareanu et al. (2020)'s results indicate that other encodings worked better when pre-trained language models are used.

To recover dependency and constituent structures, we will represent the trees using existing encodings for parsing as sequence labeling (Gómez-Rodríguez and Vilares, 2018; Strzyz et al., 2019). Under this configuration, the interaction between learning a model and searching for linguistic properties is now direct. We can use probing architectures that rely entirely on the pretrained representations, and simply add a linear layer on top to map continuous vectors to discrete labels. We can expect that the capabilities of the output layer are not enough to learn the syntactic tasks at hand by themselves, so it must rely on the quality of the pretrained representations. Yet, we also will include control baselines that we will discuss later.

Research questions We want to answer two questions: (i) how much syntax is recoverable from different LLMs? and (ii) how is it affected by aspects such as the models, the type of formalism, and the pretraining and assessment data?

In what follows, we describe the sequence labeling encodings, both for dependency and constituent paradigms (§3.1), and the specifics of the probing setup used for our experiments (§3.2).

3.1 Sequence labeling encodings of syntax

Parsing as sequence labeling can be defined as learning a function $f_n : V^n \rightarrow L^n$ to map a sequence of words into a sequence of linearized labels that can be decoded to fully recover a constituent or dependency tree. Here we are not interested in the parsers *per se*, but in whether the sequence-labeling encodings defined for them provide a simple, lossless representation of dependency and constituent trees that is useful for probing. In what follows, we briefly describe these representations.

3.1.1 Dependency parsing

Dependencies between tokens can be encoded using labels of the form (x_i, l_i), where x_i is a subset of the arcs related to the token w_i, and l_i denotes the dependency relation (Strzyz et al., 2019). There are different ways of encoding x_i[1]. We compare three families of linearizations (due to brevity, we refer to the references below for the details):

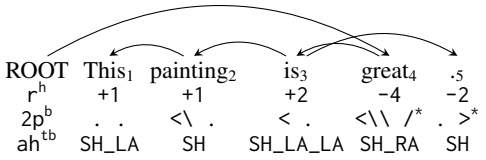


Figure 1: Example of a dependency tree linearization. Dependency types are omitted. For $2p^b$, the dot indicates no bracket in the first and/or second plane.

Head-selection (Spoustová and Spousta, 2010; Li et al., 2018; Strzyz et al., 2019). x_i encodes the dependency arc pointing directly to w_i. This can be done using an absolute index or a relative offset computing the difference between w_i's index and its head. We use (r^h) encoding where the head of w_i is the x_ith word to the right, if $x_i > 0$, and the x_ith word to the left if $x_i < 0$.[2]

Bracketing-based (Yli-Jyrä and Gómez-Rodríguez, 2017; Strzyz et al., 2020). x_i encodes the arcs using strings of brackets to represent a subset of the incoming and outgoing arcs of w_i and its direct neighbors. We use a 2-planar bracketing

[1]To ensure that the labels produce a valid tree, we apply the postprocessing described in the paper of each encoding.

[2]There are other head-selection encodings where the offset depends on some word property, e.g., PoS tags like in (Vilares et al., 2020), but using these encodings can blur the probing, since we need to access such external information.

encoding ($2p^b$) that uses two independent planes of brackets to encode non-projective trees.

Transition-based (Gómez-Rodríguez et al., 2020). x_i encodes a sub-sequence of the transitions that are generated by a left-to-right transition-based parser. Given a transition list $t = t_1, ..., t_m$ with n read transitions, t is split into n sub-sequences such that the ith sub-sequence is assigned to w_i. We use a mapping from the arc-hybrid algorithm (ah^{tb}) (Kuhlmann et al., 2011). These mappings are implicit and often perform worse than more direct encodings, but they are learnable.

These encodings produce labels with different information. Following Figure 1, for w_2 (painting), the $2p^b$ encoding states that the previous word w_1 has one incoming arc from the right ("<" symbol, but it does not say from where, as that information is encoded in other labels) and that w_2 has one outgoing arc to the left ("\" symbol, but it does not specify where). For the transition-based encoding, the mapping is less straightforward across words, but still connected to them. For instance, for w_1 ('This') the label indicates that the w_1 has no connection to w_0, that it is a dependent of w_1, and that it has no children. The motivation to compare encodings is to test: (i) the consistency of the framework, i.e., if trends across LLMs remain, and (ii) to see what information is easier to recover when the LLM weights are frozen.

3.1.2 Constituent parsing

We here use the encoding approach by Gómez-Rodríguez and Vilares (2018), which encodes common levels in the tree between pairs of tokens.[3] The labels are of the form (n_i, c_i, u_i). The element n_i encodes the number of tree levels that are common between w_i and w_{i+1}, computed as the difference with respect to n_{i-1}. The element c_i encodes the lowest non-terminal symbol that is shared between those two words. u_i encodes the leaf unary branch located at w_i, if it exists. An example is shown in Figure 2.

[3]To our knowledge, when we did the experiments, this encoding (together with variants) was the only available family of sequence-labeling encodings for constituency parsing. Contemporaneously to the end of this work, another family of encodings - based on the tetra-tagging (Kitaev and Klein, 2020) - has been proposed and implemented as a pure tagging approach (Amini and Cotterell, 2022).

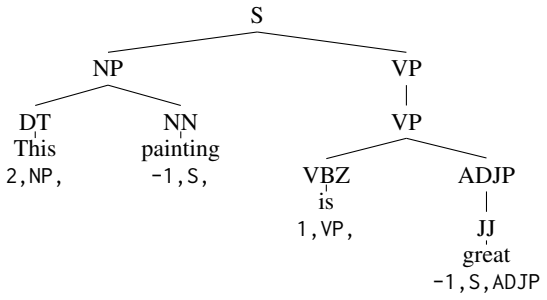


Figure 2: Example of a constituent tree linearization.

3.2 Probing architecture

We use a 1-layered feed-forward network on top of the LLMs to predict the labels. We propose three setups (training hyperparameters are detailed in Appendix A):

Frozen weights (`frz`) The LLM weights are frozen and only the weights of the linear output layer are updated during fine-tuning.

Random weights (`rnd`) Only the weights of the linear classifier layer are updated, but the weights of the encoders are randomized. We aim to prevent misleading conclusions in the hypothetical case that the linear layer can learn the mapping itself, i.e., we use this setup as a lower bound baseline. It is also a control experiment, as the difference between the results of this setup and the `frz` setup would be the measure we are looking for to estimate the amount of syntax information encoded in the representational space of pre-trained LLMs.

Fine-tuned weights (`ftd`) A fine-tuned LLM where all weights are updated, i.e., this setup is used as an upper bound baseline.

3.3 Multilingual Language Models

The method here proposed is model-agnostic. Our aim is not to obtain the highest results or to use the largest LLM. We select a few LLMs that are representative and runnable with our resources:

`mBERT` (Devlin et al., 2019) It uses WordPiece tokenization. While subword tokenizers are effective with representative splits, they yield suboptimal subtokens for low-resource languages (Agerri et al., 2020; Virtanen et al., 2019), as wrong subtokens will not encode meaningful information. mBERT is pretrained on 104 languages from the dump of the largest Wikipedias.

xlm-roberta (Conneau et al., 2020) A multilingual LLM trained as RoBERTa (Liu et al., 2019b). It has the same architecture as BERT, but only pretrained on the masked word prediction task and uses a byte-level BPE for tokenization. It has been pretrained on 2.5TB of filtered CommonCrawl data that contains text in 100 languages (XLM-100), and for longer time than mBERT.

canine (-c and -s) (Clark et al., 2022) It uses char-based tokenization, which is believed to perform better in languages that are challenging for subword tokenization, such as those with vowel harmony. It eliminates the issue of unknown tokens. It is pre-trained on masked language modeling and next sentence prediction on the same data as mBERT: canine-c is pretrained using a char-level loss, while canine-s includes a previous subword tokenization to predict masked subword tokens.

In all models, labels are first broken down into subtokens before being processed by the LLMs to assign them to the n input tokens. The classifier layer then assigns a label to each subtoken (i.e. subword for mBERT and xlm-roberta and character for canine). Then, we select the label assigned to the first sub-element, which is a common approach.

4 Methodology and Experiments

Data for dependency parsing For the assessment of dependency structures, we selected 13 Universal Dependencies (UD 2.9; Nivre et al., 2020) treebanks from different language families and with different amounts of annotated data. Although mBERT, xlm-roberta, and canine have been pretrained on different (multilingual) crawled datasets, we select treebanks whose languages are either present in all our LLMs' pretraining data or in none of them (although presence proportions might vary in the case of xlm-roberta). For more details, see Table 1. Data sizes have been obtained from Wu and Dredze (2020) for Wiki-100 and Conneau et al. (2020) for XLM-100.

Data for constituent parsing We assess constituent structures on the PTB (Marcus et al., 1993), the CTB (Xue et al., 2005), and 8 constituent treebanks from the SPMRL shared task (Seddah et al., 2014)[4], whose languages are shown in Table 2.

[4]We do not have the license for the Arabic treebank.

Treebank	Family	# Trees	# Tokens	Wiki-100 size (GB)	XLM-100 size (GB)
Skolt Sami$_{\text{Giellagas}}$	Sami	200	2 461	-	-
Guajajara$_{\text{TuDeT}}$	Tupi-Guarani	284	2 052	-	-
Ligurian$_{\text{GLT}}$	Romance	316	6 928	-	-
Bhojpuri$_{\text{BHTB}}$	Indic	357	6 665	-	-
Kiche$_{\text{IU}}$	Mayan	1 435	10 013	-	-
Welsh$_{\text{CCG}}$	Celtic	2 111	41 208	<0.1	0.8
Armenian$_{\text{ArmTDP}}$	Armenian	2 502	52 630	0.2-0.4	5.5
Vietnamese$_{\text{VTB}}$	Viet-Muon)	3 000	43 754	0.4-0.7	137.3
Chinese$_{\text{GSDSimp}}$	Sinitic	4 997	128 291	1.4-2.8	46.9
Basque$_{\text{BDT}}$	Basque	8 993	121 443	0.1-0.2	2.0
Turkish$_{\text{BOUN}}$	Turkic	9 761	122 383	0.4-0.7	20.9
Bulgarian$_{\text{BTB}}$	Slavic	11 138	146 159	0.2-0.4	57.5
Ancient Greek$_{\text{Perseus}}$	Greek	13 919	202 989	-	-

Table 1: Dependency treebanks used in this work.

Language disparity We use (mostly) different languages for each paradigm. For constituent treebanks, we only have access to rich-resource languages, so we prioritize diversity for dependencies. Comparing languages across syntax paradigms is not particularly useful, due to varying metrics, annotation complexity, and treebank comparisons. Instead, we compare error reductions against control models to estimate the recoverability of specific syntactic formalisms by an LLM (see §5).

Treebank	Family	# Trees	# Tokens	Wiki-100 size (GB)	XLM-100 size (GB)
Swedish	Germanic	5 000	81 333	0.7-1.4	12.1
Hebrew	Semitic	5 000	133 047	0.4-0.7	31.6
Polish	Slavic	6 578	73 357	1.4-2.8	44.6
Basque	Basque	7 577	103 946	0.1-0.2	2.0
Hungarian	Finno-Ugric	8 146	178 278	0.8-1.4	58.4
French	Romance	14 759	457 873	2.8-5.7	56.8
Korean	Korean	23 010	319 457	0.4-0.7	54.2
English	Germanic	39 832	989 861	11.3-22.6	300.8
German	Germanic	40 472	760 003	2.8-5.7	66.6
Chinese	Sinitic	50 734	1 235 267	1.4-2.8	46.9

Table 2: Constituent treebanks used in this work

Metrics For dependency parsing, we use Labeled Attachment Score (LAS). For constituent parsing, we use the labeled bracketing F1-score.

5 Results

We present the assessment for dependency structures in §5.1, and for constituent structures in §5.2.

5.1 Dependency parsing results

We break down the results comparing frozen vs: (i) random, and (ii) fine-tuned weights.

Frozen (frz) *vs* random weights (rnd) setups Table 3 shows the LAS results across treebanks and dependency encodings (head-based, bracketing-based, and transition-based). For mbert and xlm-roberta the performance in the frz setup

Treebank	mBERT						xlm-roberta						canine-c						canine-s					
	$2p^b$		ah^{tb}		r^h		$2p^b$		ah^{tb}		r^h		$2p^b$		ah^{tb}		r^h		$2p^b$		ah^{tb}		r^h	
	rnd	frz	rnd	frz	rnd	frz	rnd	frz	rnd	frz	rnd	frz	rnd	frz	rnd	frz	rnd	frz	rnd	frz	rnd	frz	rnd	frz
Skolt Sami	11.5	9.2	8.0	8.4	10.4	13.5	14.2	6.9	6.5	3.0	10.5	8.5	7.6	7.2	9.0	5.1	9.2	6.2	10.5	10.3	9.3	8.0	9.2	8.0
Guajajara	31.8	30.9	26.4	26.4	27.9	22.2	35.3	19.0	26.4	12.1	31.1	12.2	29.2	22.2	24.0	15.3	27.9	22.2	29.4	29.8	22.0	21.5	27.9	27.9
Ligurian	2.9	7.2	12.1	21.7	16.6	21.2	3.8	1.6	14.6	9.8	16.7	6.6	4.4	3.6	12.8	8.2	13.2	10.5	4.8	5.7	10.8	11.7	13.5	12.5
Bhojpuri	14.4	17.0	17.3	26.0	24.3	28.3	13.2	11.8	17.5	18.6	24.6	26.8	13.3	9.1	16.9	4.8	22.4	13.8	13.4	11.3	17.7	10.3	22.4	17.8
Kiche	45.2	51.0	43.0	49.0	42.3	45.6	41.6	33.4	41.2	31.5	39.5	33.1	47.4	25.7	43.2	24.3	43.1	25.3	47.8	43.1	43.3	40.7	43.1	40.3
Welsh	22.4	44.9	23.0	42.6	22.3	43.6	23.6	28.0	23.3	29.7	21.6	30.1	25.7	12.5	24.6	11.9	27.9	15.6	25.8	20.5	24.3	19.8	27.9	23.9
Armenian	15.1	38.8	13.5	33.7	19.9	34.8	13.3	31.0	12.4	25.9	18.1	30.9	15.1	13.8	13.5	10.3	18.9	16.8	14.8	19.9	13.8	15.2	18.9	21.2
Vietnamese	14.7	37.4	19.6	37.8	14.7	31.8	14.7	24.6	18.4	26.7	14.6	19.2	13.9	10.0	13.3	6.3	16.1	12.0	14.1	15.3	13.4	13.3	16.1	16.7
Chinese	11.0	42.1	14.2	39.1	21.0	38.8	1.9	17.6	5.7	18.9	11.5	25.1	13.6	15.6	15.0	14.7	20.1	19.4	13.6	24.9	15.0	23.5	20.4	27.3
Basque	17.9	45.5	16.3	41.9	19.6	40.2	17.2	40.9	15.6	37.6	18.7	32.8	18.8	14.8	16.2	12.6	20.7	16.4	18.8	22.4	16.2	19.5	20.7	22.9
Turkish	20.0	42.9	19.2	41.5	25.1	40.8	19.4	41.3	18.7	40.2	23.9	39.1	18.9	18.5	16.4	14.1	21.8	20.1	18.8	23.7	16.5	21.5	21.8	24.3
Bulgarian	20.8	63.4	22.4	56.4	25.7	54.3	22.3	55.3	22.7	47.9	26.2	46.0	23.9	18.0	21.4	16.2	26.3	21.2	23.8	29.4	21.2	25.5	26.3	30.5
A. Greek	6.6	23.7	14.5	24.3	14.9	23.8	5.4	23.7	12.9	27.3	14.3	25.6	13.4	11.3	15.4	14.0	17.3	16.4	13.6	18.4	15.4	20.3	17.3	20.6
Average	18.0	34.9	19.2	34.5	22.2	34.2	17.4	25.8	18.1	25.3	20.9	25.8	18.9	13.9	18.6	12.1	21.9	16.6	19.2	21.1	18.4	19.3	22.0	22.6

Table 3: LAS for the test sets of the dependency treebanks. LLMs and dependency encodings analyzed for the frz and rnd setups. Languages *in italics* are absent among the crawled data used to pre-train the LLMs.

clearly surpasses the rnd baseline, i.e., the control experiment. The results suggest that under the frozen setups, mbert is better than xlm-roberta at recovering dependencies, although pre-trained xlm-roberta models are usually better at downstream tasks (Liu et al., 2019b). The ranking of the LLMs is stable across treebanks. The LAS scores across encodings are in a similar range, and the average LAS across different encodings is very similar too (bottom row in Table 3). On the other hand, the results for canine do not surpass the lower bound baseline in most cases. This is unlikely to be because of a bad fitting, since the random weights baselines perform almost the same across pre-trained models, encodings and treebanks. Also, while canine-s outperforms the random baseline for the highest-resourced languages, canine-c underperforms it for all languages except for Chinese.

For a clearer picture, Figure 3 shows the relative LAS error reductions ϵ_{LAS}(rnd,frz) for the 2-planar encoding and sorted by the size of the training set used for the probe. Next, we focus on $2p^b$ as previous work has demonstrated its robustness across various configurations (Muñoz-Ortiz et al., 2021; Strzyz et al., 2019, 2020).[5] For larger treebanks, whose languages are supported by LLMs, the error reductions between the frz and rnd setups are large, showing that the LLMs encode to some extent dependency structures in their representational space. For languages that are not supported by the LLMs, the error reductions are clearly smaller. This happens for low-resource treebanks, in which only mBERT is able to obtain improvements over the rnd baseline, but also for high-resource ones, such as Ancient Greek (the largest tested treebank), suggesting that the treebank size is not a key factor for the probes (we discuss this in detail §5.3).

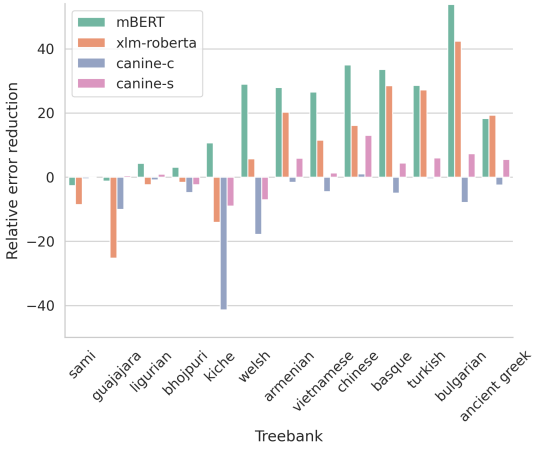


Figure 3: ϵ_{LAS}(rnd,frz) on the dependency treebanks test sets for the $2p^b$ encoding.

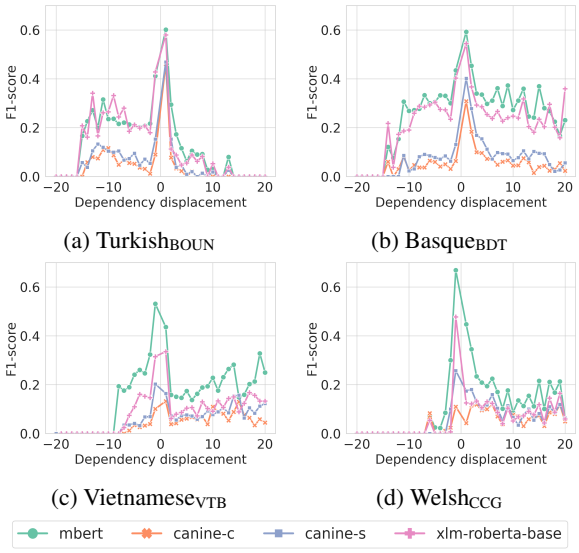
Frozen (frz) vs fine-tuned (ftd) setup Table 4 shows the scores for the fine-tuned models. In this case, xlm-roberta sequence labeling parsers obtain a larger average error reduction, while mbert obtains slightly better results for the ftd setup. The results show that even if under the frz setup dependency structures can be recovered, fine-tuning the whole architecture gives significant improvements. Also, the performance across the board for the fine-tuned models is very competitive for all treebanks supported by the LLMs. Note that even if such results lag below the state of the art (not the target of our work), we rely exclusively on multilingual pretraining vectors, without any powerful parser decoder, such as Kitaev and Klein (2018)

[5]The trends for the other encodings are similar and they can be seen in Appendix B.

for constituent parsing, or Dozat et al. (2017) for dependencies.

Encoding comparison Results from Table 3 show that the three encodings are able to recover a similar amount of syntax. It is worth noting that, although $\texttt{r}^{\texttt{h}}$ performs better for the `rnd` setup, this does not translate into a better recovering from `frz` representations. It seems also that $\texttt{2p}^{\texttt{b}}$ recovers more syntactic information in higher-resourced setups (i.e. Bulgarian), while $\texttt{r}^{\texttt{h}}$ and $\texttt{ah}^{\texttt{tb}}$ perform better in lower-resourced configurations (i.e Skolt Sami, Ligurian).

Dependency displacements Figure 4 shows the performance across arcs of different length and direction for the `frz` models with the $\texttt{2p}^{\texttt{b}}$ encoding over 4 languages: the one with most left arcs (Turkish), with most right arcs[6] (Vietnamese), and two balanced ones (Basque and Welsh). The multilingual LLMs capture the particularities of languages (for the case of the $\text{Welsh}_{\text{CCG}}$ treebank, even if it is balanced in terms of the number of left/right arcs, left arcs are on average of a distance of $1.6_{\pm 1.8}$ units while right arcs are of $3.9_{\pm 4.9}$ units). Also, the LLMs keep the trends across displacements, i.e., no LLM notably changes their expected performance with respect to the others for a specific subset of dependencies.



(a) $\text{Turkish}_{\text{BOUN}}$ (b) $\text{Basque}_{\text{BDT}}$

(c) $\text{Vietnamese}_{\text{VTB}}$ (d) $\text{Welsh}_{\text{CCG}}$

Figure 4: Average F1 score using $\texttt{2p}^{\texttt{b}}$ for different dependency displacements (signed lengths) and LLMs. We removed displacements occurring less than 10 times.

[6]Guajajara is excluded due to dataset size limitations.

Treebank	mBERT			xlm-roberta			canine-c			canine-s		
	frz	ftd	err	frz	ftd	err	frz	ftd	err	frz	ftd	err
Skolt Sami	9.2	14.6	5.9	6.9	11.2	4.6	7.2	2.6	-5.0	10.3	6.3	-4.5
Guajajara	30.9	46.6	22.6	19.0	39.0	24.7	22.2	16.6	-7.2	29.8	29.9	0.1
Ligurian	7.2	28.4	22.8	1.6	24.7	23.5	3.6	0.5	-3.2	5.7	0.8	-5.2
Bhojpuri	17.0	24.9	9.5	11.8	15.0	3.6	9.1	7.4	-1.9	11.3	14.2	3.6
Kiche	51.0	69.2	37.1	33.4	61.8	42.6	25.7	22.8	-3.9	43.1	51.1	13.9
Welsh	44.9	68.9	43.6	27.9	69.0	57.0	12.5	8.0	-5.1	20.5	31.5	13.8
Armenian	38.8	71.7	53.8	31.0	74.5	63.1	13.8	16.7	3.2	19.9	31.9	15.0
Vietnamese	37.4	58.5	33.7	24.6	60.8	48.0	10.0	7.7	-2.6	15.3	24.4	10.7
Chinese	42.1	76.2	58.9	17.6	75.9	70.7	15.6	20.0	6.3	24.9	50.4	34.0
Basque	45.5	77.4	58.8	40.9	79.3	65.0	14.8	32.9	21.2	22.4	31.9	31.6
Turkish	42.9	68.8	45.4	41.3	48.7	53.8	18.5	37.1	22.8	23.7	48.7	32.8
Bulgarian	63.4	90.4	74.0	55.3	88.1	82.1	18.0	54.1	44.0	29.4	67.0	53.3
A. Greek	23.7	51.7	36.7	23.7	67.7	57.7	11.3	37.1	29.0	18.4	45.8	33.6
Average	34.9	57.5	38.7	25.7	55.1	45.9	14.0	20.3	7.5	21.1	34.4	17.9

Table 4: LAS for the `frz` and `ftd` setups on the test sets, together with ϵ_{LAS}(`frz`,`ftd`) for the $\texttt{2pb}^{\texttt{b}}$ encoding for all treebanks and LLMs tested. Languages in italics are absent in the pretraining data of the LLMs.

5.2 Constituent parsing results

We break down the results comparing frozen *vs*: (i) random, and (ii) fine-tuned weights.

Frozen (`frz`) *vs* random weights (`rnd`) setups Table 5 shows the bracketing F1 score across treebanks and the encodings for the two setups. The trend from dependency parsing remains: `mBERT` outperforms `xlm-roberta` for all languages, while `canine-s` outperforms `canine-c`. In this case, `canine-s` improves over the random baseline for all treebanks, while `canine-c` only outperforms the random baseline for 3 out of 10 models, which suggests the difficulties that these character-level language models have to model syntax, even if they perform well on other downstream tasks. The exceptions are Korean, German and Chinese. Chinese was also an exception in the case of dependency parsing, so an explanation might be that its writing systems encode more information per character than other languages. Chinese characters represent a whole morpheme, being more similar to a subword token, while Korean Hangul encodes a syllable per character, instead of a single sound as alphabets of the other languages tested.

Figure 5 shows the error reductions across the board, sorted by the size of the training data used for the probing. In this case, all tested languages are supported by the LLMs, but there are large differences in the size of the training data (e.g., Swedish with 5 000 sentences *vs* German with 40 472 sentences). However, we do not see an increase in error reduction when the size of training data grows.

Frozen (`frz`) vs fine-tuned (`ftd`) setup Table 6 compares the bracketing F1-scores for the frozen

Treebank	mBERT		xlm-roberta		canine-c		canine-s	
	rnd	frz	rnd	frz	rnd	frz	rnd	frz
Swedish	29.8	56.0	30.1	42.3	25.5	22.7	25.5	28.7
Hebrew	41.5	74.5	43.3	60.0	40.8	29.8	41.0	40.2
Polish	42.8	77.0	41.8	68.0	40.1	33.9	40.1	42.9
Basque	32.5	56.6	33.7	47.1	36.2	33.0	36.2	41.4
Hungarian	40.0	69.9	39.6	66.0	37.4	31.5	37.4	41.0
French	14.5	50.1	15.4	32.4	14.1	12.1	13.9	20.1
Korean	33.2	57.4	32.9	53.2	33.8	37.5	33.8	42.0
English	12.9	57.3	14.4	40.5	9.9	9.6	9.9	17.5
German	18.9	45.4	18.1	41.2	16.4	18.5	16.4	24.0
Chinese	16.1	56.6	8.2	45.6	16.9	25.6	17.3	39.3
Average	28.2	60.1	27.8	49.6	27.0	25.4	27.2	33.7

Table 5: F-score for the test sets of the constituent treebanks. LLMs analyzed for the `frz` and `rnd` setups.

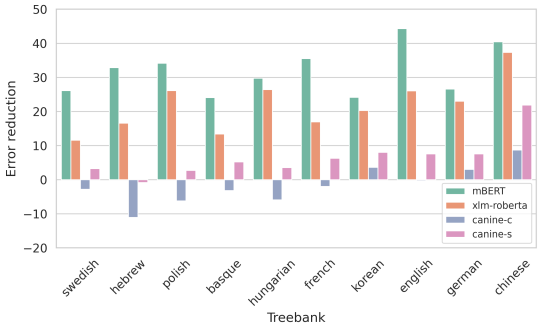


Figure 5: ϵ_{F1}(rnd,frz) on the constituent test sets.

and fine-tuned setups, and the behaviors are similar to those obtained in the case of dependency parsing, except for what looks like some empirical outlier, e.g., the fine-tuned `mBERt` for Hebrew. Hebrew also obtains the lowest error reductions for all LLMs.

Treebank	mBERT			xlm-roberta			canine-c			canine-s		
	frz	ftd	err	frz	ftd	err	frz	ftd	err	frz	ftd	err
Swedish	56.0	79.4	53.2	42.3	79.9	65.2	22.7	29.6	8.9	28.7	47.8	65.2
Hebrew	74.5	75.4	3.5	59.9	76.2	40.6	29.8	36.4	9.4	40.1	53.8	22.9
Polish	77.0	93.4	70.9	68.0	94.0	81.2	33.9	56.6	34.3	42.9	73.9	54.3
Basque	56.6	85.0	65.4	47.1	85.1	71.6	33.0	49.1	24.0	41.4	62.7	36.2
Hungarian	69.8	91.5	71.9	66.0	92.1	76.8	31.5	52.7	30.9	41.0	65.8	42.0
French	50.1	82.2	64.3	32.4	82.6	74.3	12.1	70.2	66.1	20.1	76.0	70.0
Korean	57.4	86.4	68.1	53.2	88.0	74.4	37.5	66.0	45.6	41.9	71.4	50.8
English	57.2	91.9	81.1	40.5	92.8	87.9	9.6	82.4	80.5	17.5	86.6	83.8
German	45.4	87.3	76.7	41.1	88.4	80.3	18.5	71.7	65.3	24.0	77.3	70.1
Chinese	56.6	85.5	66.6	45.6	88.9	79.6	25.6	67.0	55.6	39.3	74.4	57.8
Average	60.1	85.8	62.2	49.6	86.8	73.2	25.4	58.2	42.1	33.7	69.0	55.3

Table 6: F-score for the test sets of the constituent treebanks, LLMs analyzed for the `ftd` *vs* the `frz` setup.

Span lengths Plotting the F1-score for each span length is the rough alternative to dependency displacements in the context of constituent parsing. In Figure 6 we again show specific examples for some of the studied languages: the most left-branching language (Korean), two balanced ones (Basque and Hungarian), and the most right-branching one (English). Similarly to the case of dependency parsers,

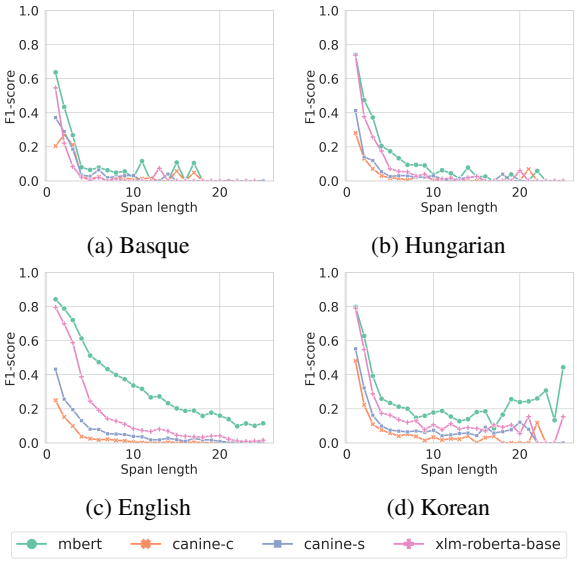


Figure 6: Average F1 score for different span lengths and LLMs.

the trends across models persist across different span lengths. They show that, regarding LLMs, `mbert` obtains the highest F-score for longer spans, while `xlm-roberta` shows great differences between shorter and longer spans. The `canine` models perform worse for all lengths.

5.3 Discussion

We now discuss the main insights and potential limitations of the proposed assessment framework.

Pretraining data *versus* Assessment data An interesting question that arises from multilingual recovery is whether the probe is able to recover the trees due to the size of the training data used for the assessment, although in theory it should be hard to learn by itself by an initially clueless classifier (the random baseline). The experiments show evidence that the size of the training data is not a primary factor to do multilingual, multi-formalism linear probing as sequence labeling. For constituent parsing, we observed that larger treebanks did not come with an increment in the error reductions between the frozen and the random setups, and that the control experiment can thus be used to give an estimate of the amount of structure recoverable from pretrained representations. Similarly, in the context of dependency parsing, we encountered an analogous situation. Despite the existence of treebanks for languages unsupported by the LLMs, spanning both large (Ancient Greek) and small treebanks (Skolt Sami or Bhojpuri), we observe that treebank size does not significantly impact the reduction in

errors between the frozen and random setups.

Either with big or small data, the error reduction between the random and the frozen models is clearly lower than for the treebanks where the language is supported by the LLMs. Among rich-resource treebanks, the size of the data does not have a great influence on the error reductions between the random and frozen weights setups, suggesting that dataset size does not influence the estimates of the dependency structure that are recoverable from the representations.

Language model differences The results on the tested LLMs suggest that subword tokenization is necessary to represent syntax, in contrast with token-free models, even if these can later perform well on downstream tasks that require compositionality. Particularly, not only do subword-based models outperform char-based ones, but also `canine-s`, which is trained using subword loss even though it is a char-level model, performs significantly better than `canine-c`. It is noteworthy that `xlm-roberta` generally outperforms `mBERT` in most downstream tasks, including parsing, as previous studies showed (Conneau et al., 2020) and in our fine-tuned results, it performs on par on dependency parsing (Table 4) and outperforms `mBERT` in constituency parsing (Table 6). Yet, for the frozen weights setup, `mBERT`'s representations recovered slightly but consistently better syntactic representations. This suggests that the improvements in how `xlm-roberta` was trained with respect to `mBERT`, e.g., training for longer time, or more data, are not key factors to better encode syntax. Additionally, based on our experiments, it appears that `mBERT` demonstrates a certain level of proficiency in recovering syntax information for the smallest treebanks, particularly for languages not included in the pretraining data (such as Ligurian, Bhojpuri, and Kiche). This suggests a capacity to extend its syntactic knowledge to previously unseen languages, albeit to a limited extent, unlike the other models.

Syntactic formalism Previous studies (e.g., Vilares et al. (2020)), hypothesized that pre-trained word vectors might fit better constituent- than dependency-based tasks, since the masked language objective links better with the former formalism, i.e., when a model is learning to unblur a masked token, the constituent structure is to some extent implicit (e.g., an adjective is missing between the determiner and the noun, forming a noun phrase), while dependencies are less obvious. We could not find a clear evidence of this. Although some of the `frz` models are unable to surpass the `rnd` baseline in the case of dependencies (while this is not the case for constituents), these instances are languages that are not present in the pretraining data, except for the `canine` models.

6 Conclusion

We proposed a sequence-labeling framework to recover multi-formalism syntactic structures from multilingual LLMs. By mapping syntactic trees to labels we associated output word vectors to labels that encode a portion of the tree, while using a single assessment framework for both constituent and dependency structures. We compared three popular multilingual language models. The results show that subword LLMs can recover a percentage of these structures. We evaluated the outcomes by calculating the reduction in errors compared to control models, aiming to gauge the extent to which an LLM can recover specific syntactic structures. The assessment appears reliable and unaffected by variables like the training set's size employed for probing, highlighting that pretraining data is an important factor for recoverability. Last, we found no clear evidence that contextualized vectors encode constituent structures better than dependencies (nor the opposite).

Limitations

Physical resources We did not consider larger language models as we do not have access to the necessary computational resources to run then, hence limiting the scope of our study. We only had access to 2 GeForce RTX 3090, having a total GPU memory of 48 GB, insufficient for fine-tuning many LLMs over different treebanks and formalisms, as in this work.

Language diversity The constituent treebanks used are all from languages that are relatively rich-resource and are present on the pretraining data of the LLMs. To the best of our knowledge there are no available constituent treebanks from lower-resource languages that are also absent in multilingual LLMs. In consequence, we could not test the effect of absence of pretraining data in order to see if the trends obtained in dependency treebanks prevail here. In addition, for dependency parsing, even

a large multilingual resource like Universal Dependencies only has data for about 100 languages, a tiny fraction of the 7 000 existing human languages.

Interpretation As mentioned in the introduction, we have to be careful when dealing with probing frameworks. Although we developed solid experiments, and also included control experiments, syntax knowledge is hard to isolate, measure and interpret, so we have tried to be careful with our conclusions.

Acknowledgments

We acknowledge the European Research Council (ERC), which has funded this research under the Horizon Europe research and innovation programme (SALSA, grant agreement No 101100615), ERDF/MICINN-AEI (SCANNER-UDC, PID2020-113230RB-C21), Xunta de Galicia (ED431C 2020/11), grant FPI 2021 (PID2020-113230RB-C21) funded by MCIN/AEI/10.13039/501100011033, and Centro de Investigación de Galicia "CITIC", funded by the Xunta de Galicia through the collaboration agreement between the Consellería de Cultura, Educación, Formación Profesional e Universidades and the Galician universities for the reinforcement of the research centres of the Galician University System (CIGUS).

References

Rodrigo Agerri, Iñaki San Vicente, Jon Ander Campos, Ander Barrena, Xabier Saralegi, Aitor Soroa, and Eneko Agirre. 2020. Give your text representation models some love: the case for Basque. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4781–4788, Marseille, France. European Language Resources Association.

Afra Amini and Ryan Cotterell. 2022. On parsing as tagging. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8884–8900, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Miguel Ballesteros and Joakim Nivre. 2012. MaltOptimizer: A system for MaltParser optimization. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2757–2763, Istanbul, Turkey. European Language Resources Association (ELRA).

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI*, 2005(598-603):18.

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.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, Santa Cruz, California, USA. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for 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.

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.

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

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. 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 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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

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

Nikita Kitaev and Dan Klein. 2020. Tetra-tagging: Word-synchronous parsing with linear-time inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6255–6261, Online. Association for Computational Linguistics.

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.

Artur Kulmizev and Joakim Nivre. 2022. Schrödinger's tree – on syntax and neural language models. *Frontiers in Artificial Intelligence*, 5:796788.

Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do neural language models show preferences for syntactic formalisms? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4077–4091, Online. 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.

Chen Lin, Timothy Miller, Alvin Kho, Steven Bethard, Dmitriy Dligach, Sameer Pradhan, and Guergana Savova. 2014. Descending-path convolution kernel for syntactic structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 81–86, Baltimore, Maryland. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

André Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mário Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44, Cambridge, MA. Association for Computational Linguistics.

Rowan Hall Maudslay and Ryan Cotterell. 2021. Do syntactic probes probe syntax? experiments with jabberwocky probing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 124–131, Online. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Victor Milewski, Miryam de Lhoneux, and Marie-Francine Moens. 2022. Finding structural knowledge in multimodal-BERT. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5658–5671, Dublin, Ireland. Association for Computational Linguistics.

Max Müller-Eberstein, Rob van der Goot, and Barbara Plank. 2022a. Probing for labeled dependency trees. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7711–7726, Dublin, Ireland. Association for Computational Linguistics.

Max Müller-Eberstein, Rob van der Goot, and Barbara Plank. 2022b. Sort by structure: Language model ranking as dependency probing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1307, Seattle, United States. Association for Computational Linguistics.

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.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

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

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

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

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. E-BERT: Efficient-yet-effective entity embeddings for BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 803–818, Online. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2022. Square one bias in NLP: Towards a multi-dimensional exploration of the research manifold. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2340–2354, Dublin, Ireland. Association for Computational Linguistics.

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

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 12–23, Copenhagen, Denmark. Association for Computational Linguistics.

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

Karolina Stanczak, Edoardo Ponti, Lucas Torroba Hennigen, Ryan Cotterell, and Isabelle Augenstein. 2022. Same neurons, different languages: Probing morphosyntax in multilingual pre-trained models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1589–1598, Seattle, United States. 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.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

Robert Vacareanu, George Caique Gouveia Barbosa, Marco A. Valenzuela-Escárcega, and Mihai Surdeanu. 2020. Parsing as tagging. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5225–5231, Marseille, France. European Language Resources Association.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

David Vilares, Michalina Strzyz, Anders Søgaard, and Carlos Gómez-Rodríguez. 2020. Parsing as pretraining. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9114–9121.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

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

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.

Min Zhang and Haizhou Li. 2009. Tree kernel-based SVM with structured syntactic knowledge for BTG-based phrase reordering. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 698–707, Singapore. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

A Hyperparameters

We selected a learning rate of $5 \cdot 10^{-5}$ for the `ftd` models and $2 \cdot 10^{-3}$ for the `rnd` and `frz` models based on the results of our preliminary experiments, as the `ftd` models showed faster convergence. For the three setups, we trained the models during 20 epochs (models had converged at this point). We trained our models on two GeForce RTX 3090 using a batch of 32 on each and a gradient accumulation of 2 for a total batch of 128. Training time of the final models accounts for approximately 60 GPU hours (24 for constituent, 6 per LLM, and 36 for dependency, 8 per LLM).

A.1 `mBERT` hyperparameters

Hyperparameter	Value
"attention_probs_dropout_prob"	0.1
"classifier_dropout"	null
"directionality"	"bidi"
"hidden_act"	"gelu"
"hidden_dropout_prob"	0.1
"hidden_size"	768
"layer_norm_eps"	1e-12
"max_position_embeddings"	512
"model_type"	"bert"
"num_attention_heads"	12
"num_hidden_layers"	12
"pad_token_id"	0
"pooler_fc_size"	768
"pooler_num_attention_heads"	12
"pooler_num_fc_layers"	3
"pooler_size_per_head"	128
"pooler_type"	"first_token_transform"
"position_embedding_type"	"absolute"
"torch_dtype"	"float32"
"transformers_version"	"4.25.1"
"type_vocab_size"	2
"use_cache"	true
"vocab_size"	119547

Table 7: Hyperparameters for `mBERT` models.

A.2 `xlm-roberta-base` hyperparameters

Hyperparameter	Value
"attention_probs_dropout_prob"	0.1
"classifier_dropout"	null
"eos_token_id"	2
"hidden_act"	"gelu"
"hidden_dropout_prob"	0.1
"hidden_size"	768
"layer_norm_eps"	1e-05
"max_position_embeddings"	514
"model_type"	"xlm-roberta"
"num_attention_heads"	12
"num_hidden_layers"	12
"pad_token_id"	1
"position_embedding_type"	"absolute"
"torch_dtype"	"float32"
"transformers_version"	"4.25.1"
"type_vocab_size"	1
"use_cache"	true
"vocab_size"	250002

Table 8: Hyperparameters for `xml-roberta` models.

A.3 `canine` hyperparameters

Hyperparameter	Value
"attention_probs_dropout_prob"	0.1
"bos_token_id"	57344
"downsampling_rate"	4
"eos_token_id"	57345
"hidden_act"	"gelu"
"hidden_dropout_prob"	0.1
"hidden_size"	768
"layer_norm_eps"	1e-12
"local_transformer_stride"	128
"max_position_embeddings"	16384
"model_type"	"canine"
"num_attention_heads"	12
"num_hash_buckets"	16384
"num_hash_functions"	8
"num_hidden_layers"	12
"pad_token_id"	0
"torch_dtype"	"float32"
"transformers_version"	"4.25.1"
"type_vocab_size"	16
"upsampling_kernel_size"	4
"use_cache"	true

Table 9: Hyperparameters for `canine-c and -s` models.

B Error reduction for $\texttt{r}^{\texttt{h}}$ and $\texttt{ah}^{\texttt{tb}}$

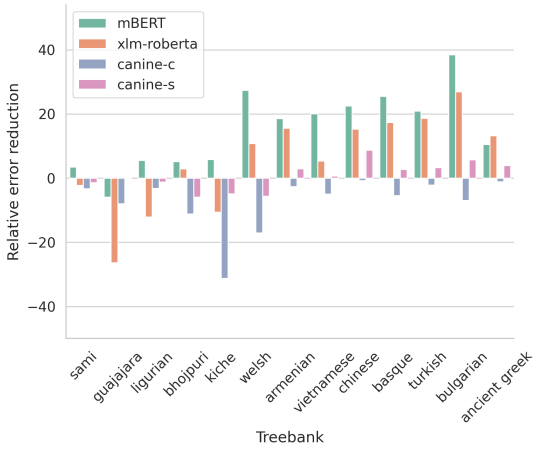


Figure 7: ϵ_{LAS}(rnd,frz) for the $\texttt{r}^{\texttt{h}}$ encoding for all LLMs tested.

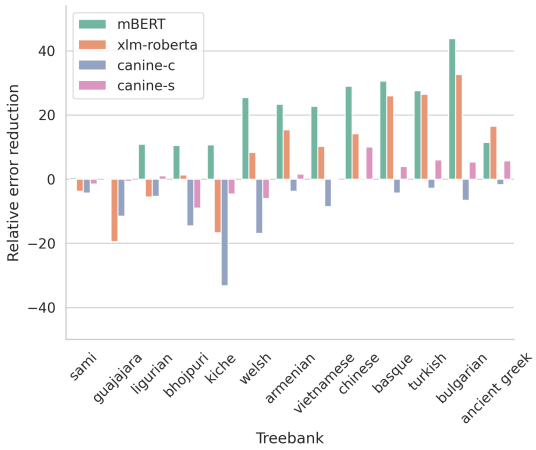


Figure 8: ϵ_{LAS}(rnd,frz) for the $\texttt{ah}^{\texttt{tb}}$ encoding for all LLMs tested.

C Evaluation scripts

We used the evaluation scripts `conll18_eval.` for dependencies and `EVALB` for constituencies.