Probing for Constituency Structure in Neural Language Models

David Arps[†] **Younes Samih**[†] **Laura Kallmeyer**[†] **Hassan Sajjad**[‡]
[†]Heinrich Heine University Düsseldorf, Germany
[‡]Faculty of Computer Science, Dalhousie University, Canada
{david.arps,younes.samih,laura.kallmeyer}@hhu.de
hsajjad@dal.ca

Abstract

In this paper, we investigate to which extent contextual neural language models (LMs) implicitly learn syntactic structure. More concretely, we focus on constituent structure as represented in the Penn Treebank (PTB). Using standard probing techniques based on diagnostic classifiers, we assess the accuracy of representing constituents of different categories within the neuron activations of a LM such as RoBERTa. In order to make sure that our probe focuses on syntactic knowledge and not on implicit semantic generalizations, we also experiment on a PTB version that is obtained by randomly replacing constituents with each other while keeping syntactic structure, i.e., a semantically ill-formed but syntactically well-formed version of the PTB. We find that 4 pretrained transfomer LMs obtain high performance on our probing tasks even on manipulated data, suggesting that semantic and syntactic knowledge in their representations can be separated and that constituency information is in fact learned by the LM. Moreover, we show that a complete constituency tree can be linearly separated from LM representations.[1]

1 Introduction

Over the last years, neural language models (LMs), such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019b), and DistilBERT (Sanh et al., 2020), have delivered unmatched results in multiple key Natural Language Processing (NLP) benchmarks (Wang et al., 2018). Despite the impressive performance, the black-box nature of these models makes it difficult to ascertain whether they implicitly learn to encode linguistic structures, such as constituency or dependency trees.

There has been a considerable amount of research conducted on questioning which types of linguistic structure are learned by LMs (Tenney et al., 2019b; Conneau et al., 2018; Liu et al., 2019a). The motivation behind asking this question is two-fold. On the one hand, we want to better understand how pre-trained LMs solve certain NLP tasks, i.e., how their input features and neuron activations contribute to a specific classification success. A second motivation is an interest in distributional evidence for linguistic theory. That is, we are interested in assessing which types of linguistic categories emerge when training a contextual language model, i.e., when training a model only on unlabeled text. The research in this paper is primarily motivated by this second aspect, focusing on syntactic structure, more concretely on constituency structure. We investigate, for instance, for pairs of tokens in a sentence whether a LM implicitly learns which constituent (NP, VP, ...) the two tokens belong to as their lowest common ancestor (LCA). We use English Penn Treebank data (PTB, Marcus et al., 1993) to conduct experiments.

A number of studies have probed LMs for dependency structure (Hewitt and Manning, 2019; Chen et al., 2021) and constituency structure (Tenney et al., 2019b). We probe constituency structure for the following reasons. In contrast to dependency structure, it can be richer concerning the represented abstract syntactic information, since it directly assigns categories to groups of tokens. On the other hand, not all dependency labels are represented in a standard constituency structure; but they can be incorporated as extensions of the corresponding non-terminal nodes (see, e.g., the PTB label NP-SBJ in App. A.1). To quantify the gain that we get from probing constituency rather than dependency trees, we compare the unlabeled bracketings in the syntactic trees in both formalisms on the PTB (Marcus et al., 1993; de Marneffe et al., 2006), where an unlabeled bracketing is the yield of a subtree. We find that while 97% of the bracketings in a dependency tree are also present in a constituency tree, only 54% of the bracketings in

[1]Code for our experiments is available at https://github.com/davidarps/constptbprobing

the constituency tree are present in the dependency tree. This shows that constituent trees can contain much more fine-grained hierarchical information than dependency trees. A further reason for focusing on constituency structure is that this is the type of structure most linguistic theories use.

We use diagnostic classifiers (Hupkes et al., 2018) and perform model-level, layer-level and neuron-level analyses. Most work on diagnostic classifiers performs mean pool over representations when probing for a relationship between two words (Durrani et al., 2020). We empirically show that mean pool results in lossy representation, and we recommend concatenation of representations as a better way to probe for relations between words.

A difficulty when probing a LM for whether certain categories are learned is that we cannot be sure that the LM does not learn a different category instead that is also predictive for the category we are interested in. More concretely, when probing for syntax, one should make sure that it is not semantics that one finds and considers to be syntax (since semantic relations influence syntactic structure). This point was also observed by Gulordava et al. (2018) and, more recently, Hall Maudslay and Cotterell (2021). Therefore, before probing the LM for syntactic relations, we manipulate our data by replacing a subset of tokens with other tokens that appear in similar syntactic contexts, thereby obtaining nonsensical text that still has a reasonable syntactic structure. We then conduct a series of experiments that show that even for these nonsensical sentences, contextual LMs implicitly represent constituency structure. Lastly, we questioned whether a full syntactic tree can be reconstructed using the linear probe. We achieve a labeled F1 score of 82.6% for RoBERTa when probing on the non-manipulated dataset in comparison to 51.4% with a random representation baseline.

The contributions of our work are as follows:

- We find that constituency structure is linearly separable at various granularity levels: At the model level, we find that four different LMs achieve similar overall performance on our syntactic probing tasks, but make slightly different predictions. At the layer level, the middle layers achieve the best results. At the neuron level, syntax is heavily distributed across neurons.
- We use perturbed data to separate the effect of semantics when probing for syntax, and we find that different sets of neurons capture syntactic and semantic information.
- We show that a simple linear probe is effective in analyzing representations for syntactic properties and we show that a full constituency tree can be linearly separated from LM representations.

The rest of the paper is structured as follows. The next section introduces related work. We define our linguistic probing tasks in Sec. 3. Sec. 4 introduces our experimental methodology. Sec. 5, 6, and 7 discuss our experiments and their results, and Sec. 8 concludes.

2 Related work

Syntactic information in neural LMs An important recent line of research (Adi et al., 2017; Hupkes et al., 2018; Zhang and Bowman, 2018; Blevins et al., 2018; Hewitt and Manning, 2019; Hewitt and Liang, 2019; Reif et al., 2019; Tenney et al., 2019a; Manning et al., 2020; Hall Maudslay et al., 2020; Li et al., 2020; Newman et al., 2021; Hewitt et al., 2021; Belinkov, 2022; Sajjad et al., 2022b; Dalvi et al., 2022) has focused on finding latent hierarchical structures encoded in neural LMs. A common line of work on interpreting models use a probing classifier to gauge the amount of linguistic knowledge learned in the representation (Alain and Bengio, 2016; Belinkov et al., 2020; Conneau et al., 2018).

An ample body of research exists on probing the sub-sentential structure of contextualized word embeddings. Peters et al. (2018) probed neural networks to see to what extent span representations capture phrasal syntax. Tenney et al. (2019b) devised a set of *edge probing* tasks to get new insights on what is encoded by contextualized word embeddings, focusing on the relationship between spans rather than individual words. This enables them to go beyond sequence labeling problems to syntactic constituency, dependencies, entity labels, and semantic role labeling. Their results on syntactic constituency are in line with our findings. The major difference is that we employ simpler probes while achieving similar results. Moreover, we separate the effect of semantics using corrupted data and we reconstruct full constituent trees using our probing setup. Most recently, Wu et al. (2020) propose a parameter-free probing technique to analyze LMs via perturbed masking. Their approach is based on

accessing the impact that one word has on predicting another word within a sequence in the Masked Language Model task. They have also shown that LMs can capture syntactic information with their self-attention layers being capable of surprisingly effective learning.

Hewitt and Manning (2019) demonstrated, by using a structural probe, that it is possible to find a linear transformation of the space of the LM's activation vectors under which the distance between contextualized word vectors corresponds to the distance between the respective words in the dependency tree. In a similar vein, Chen et al. (2021) introduced another structural probe, *Poincaré probe* and have shown that syntactic trees can be better reconstructed from the intermediate representation of BERT in a hyperbolic subspace.

Syntactic and semantic knowledge Gulordava et al. (2018) and Hall Maudslay and Cotterell (2021) recently argued that the work on probing syntax does not fully separate the effect of semantics while probing syntax. Both modify datasets such that the sentences become semantically nonsensical while remaining syntactically well-formed in order to assess, based on this data, whether a LM represents syntactic information. Gulordava et al. (2018) modify treebanks in four languages by replacing content words with other content words that have matching POS and morphological features. They focus on the question if agreement information in the nonce sentences can be recovered from RNN language models trained on regular data. Hall Maudslay and Cotterell (2021) replaced words with pseudowords in a dependency treebank, and quantified how much the pseudowords affect the performance of syntactic dependency probes. We followed a similar setup to separate out the effect of syntax from semantics. In contrast to Hall Maudslay and Cotterell (2021), we replace words with other words (not pseudowords) that occur in a similar syntactic context but are different semantically. This way, the LM has seen most or all words from the semantically nonsensical sentences in pretraining and has learned their syntactic properties.

Fine-grained LM analysis Durrani et al. (2020) used a unified diagnostic classifier approach to perform analyses at various granularity levels, extending Dalvi et al. (2019a). We follow their approach and perform model-level, layer-level and neuron-level analyses (Sajjad et al., 2022a). We additionally extend their approach by proposing an improved way to probe representations of two words. Previous work has mainly used a bilinear probe to investigate syntax. We select a linear model for our experiments. Selecting a weak model ensures that the representations learn the linguistic property, and the probe is not relying on the strength of the classification model used.

3 Diagnostic Tasks and Constituency trees

In this section, we define three classification tasks that are aimed at making different properties of constituency structure explicit. More specifically, the goal of these tasks is to make explicit if and how the LMs encode syntactic categories, such as `S`, `NP`, `VP`, `PP`. The first task, lowest common ancestor prediction, focuses on constituents that span large portions of the sentence. The second task, chunking, focuses on constituents with smaller spans. The third task focuses on complete syntactic trees.

Lowest common ancestor (LCA) prediction Let $s = w_0, \ldots, w_n$ be a sentence. Given combined representations of two tokens w_i, w_j with $j \geq i$, predict the label of their lowest common ancestor in the constituency tree. LCA prediction is a multiclass classification task with 28 target classes (for the PTB). In the example in Fig. 1, *luxury* and *maker* have the LCA `NP`: the lowest node dominating both words has label `NP` (ignoring the function tag `SBJ`). The task also predicts LCA of two identical tokens. In this case, the lowest phrasal node above the token is the target label (for example, `VP` is the target label for *sold*).

Chunking For each token w_i, predict whether it is the beginning of a phrase (`B`), inside a phrase (`I`), the end of a phrase (`E`) or if the token constitutes a single-token phrase (`S`). A token can be part of more than one phrase, and in this case we consider the shortest possible phrase only. For example, *1,214* in Fig. 1 has label `B` because it marks the beginning of a noun phrase. We also propose a version of this task with finer labels that combine `B,I,E,S` with the different phrase labels. In the detailed tagset, $1,214$ receives the label `B-NP`.

Reconstructing full constituent trees Vilares et al. (2020) considered constituency parsing as a multi-label sequence labeling problem. For each token w_i, three labels are predicted: First, the label of the LCA of the token pair (w_i, w_{i+1}). Second, the

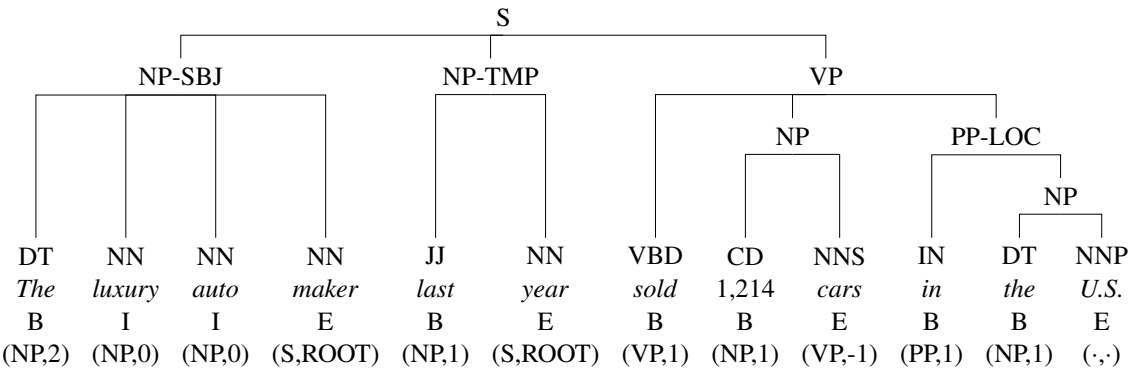


Figure 1: Example tree from the PTB. The line below the text shows gold labels for the simple chunking task. The bottom line shows label pairs from which the complete tree can be reconstructed.

depth of the LCA of (w_i, w_{i+1}) in the tree, relative to the depth of (w_{i-1}, w_i). Third, if the first token is a single-word constituent, and the label of the internal tree node directly above w_i. (Tokens in multiword constituents make up $> 90\%$ of the data and receive a negative label.). For the first two classifications, see Fig. 1. We build separate linear classifiers for each of these tasks and use their predictions to reconstruct full constituent trees.

4 Methods

Diagnostic classification A common method to reveal linguistic representations learned in contextualised embeddings is to train a classifier, a probe, using the activations of the trained LM as features. The classifier performance provides insights into the strength of the linguistic property encoded in contextualised word representations. For all our experimental setups, we employ the NeuroX toolkit (Dalvi et al., 2019b) for diagnostic classification, as it confers several mechanisms to probe neural models on the level of both neurons and layers.

Layer-level probing We probe the activations of individual layers with linear classifiers to measure the linear separability of the syntactic categories at this layer. The performance at each layer serves as a proxy to how much information it encodes with respect to a given syntactic property.

Neuron-level probing Layer-wise probing cannot account for all syntactic abstractions encoded by individual neurons in deep networks. Some groups of neurons that are spread across many layers might robustly respond to a given linguistic property without being exclusively specialized for its detection. By operating also at the level of the neurons, we aim at separating the most salient neurons across the network that learn a given linguistic property. We conduct a *linguistic correlation analysis*, as proposed by Dalvi et al. (2019a). This consists in augmenting the linear classifier with elastic net regularization (Zou and Hastie, 2005). The classifier is then trained by minimizing the loss function in Eq. 1:

$$\begin{aligned}\mathcal{L}(\theta) \quad = \quad & -\sum_j \log P_\theta(t_{s_j}|s_j) \\ & +\lambda_1\|\theta\|_1 + \lambda_2\|\theta\|_2^2\end{aligned} \tag{1}$$

where (θ) are the trained weights of the classifier and $\lambda_1\|\theta\|_1$ and $\lambda_2\|\theta\|_2^2$ correspond to L_1 and L_2 regularization. Elastic net regularization strikes a balance between selecting very focused features (neurons) (L_1) versus distributed features (L_2) shared across many properties. The input neurons to the linear classifier are ranked by saliency with respect to the classification task.

Input Representation We combine the representation vectors $x_i, x_j \in \mathbb{R}^r$ of two tokens in LCA prediction and parse tree reconstruction via concatenation ($concat(x_i, x_j) \in \mathbb{R}^{2r}$). In all experiments, *concat* produced significantly better results than elementwise averaging or a variant of the maximum. We cover the latter methods in Apps. A.3 and A.6.

5 Experimental Setup

5.1 Data

We use data from the English Penn Treebank (PTB, Marcus et al., 1993) for all our experiments. As preprocessing, we remove punctuation and null elements from the trees. The original dataset makes use of fine-grained category labels that consist of the syntactic category and function tags. Function tags indicate grammatical (such as `SBJ` for subject) or adverbial (`LOC` for locative) information.

For chunking, the label distribution is relatively balanced. For LCA prediction, we remove all function tags to keep the number of target labels small. Most of the token pairs have a relatively large distance in the constituent tree, and their LCA is a node very high in the tree (typically with a label for some kind of sentence, such as `S` or `SBAR`). In addition, some phrase labels are less frequent than others (see App. A.4). We train and evaluate on the standard PTB training/development split.

5.1.1 Syntactic and semantic knowledge

To ensure that the probing classifier captures syntactic properties and not semantic properties, we use the original PTB data as well as two modified versions of the PTB with semantically nonsensical sentences that have the same syntactic structure as the original data. The modified versions of the PTB are obtained by making use of the dependency PTB (de Marneffe et al., 2006):

1. Record the dependency context of each token in the dataset. The dependency context consists of (i) the POS tag, (ii) the dependency relation of the token to its head, and (iii) the list of dependency relations of the token to its dependents.
2. Replace a fraction of tokens with other tokens that also appear in the dataset in the same dependency context.

Two versions are created, replacing either a third or two thirds of the tokens. See Table 1 for two examples. When creating manipulated datasets, we separate the training and evaluation data. To create manipulated training data, we look for token replacements in the training split of the PTB (PTB sections 0-18). For manipulated evaluation data, we look for token replacements in the development and test splits of the PTB (sections 19-24). This ensures that the training and evaluation data do not mix, and at the same time, meanings of the newly created sentences are as diverse as possible.

5.2 Probing Classifier Settings

We use linear classifiers trained for 10 epochs with Adam optimization, an initial learning rate of 0.001 and regularization parameters $\lambda_1 = \lambda_2 = 0.001$. The contextualized representations for an input token is created by averaging the representations of its subword tokens of the LM tokenizer.

5.3 Transformer Models

We train structural probes on three 12-layered pretrained transformer models; the base versions of BERT (uncased, Devlin et al., 2019), XLNet (cased, Yang et al., 2019), and RoBERTa (Liu et al., 2019b), and a 6-layered model; DistilBERT (uncased, Sanh et al., 2020). This provides an opportunity to compare the syntactic knowledge learned in models with different hyperparameter settings and pretraining objectives.

5.4 Baselines

We use three baselines to put the results of our probes into context.

Random BERT The first baseline is used in all experiments. It evaluates how much information about linguistic context is accumulated in the LM during pretraining (Belinkov, 2022). The model for this baseline has the same neural architecture, vocabulary and (static) input embeddings as BERT base, but all transformer weights are randomized.

Selectivity To evaluate if the the probe makes linguistic information explicit or just memorizes the tasks, we use the control task proposed by Hewitt and Liang (2019) and described in App. A.2. The difference between control task performance and linguistic task performance is called selectivity. The higher the selectivity, the more one can be sure that the classifier makes linguistic structure inside the representations explicit and does not memorize the task. This baseline is used for the chunking and, in a modified version, the LCA experiments.

Individual tokens This baseline evaluates how much the representation of each token, in contrast to token pairs, contributes to the overall performance. This baseline is used only for the LCA experiments. We train two classifiers using the representation of either the first token in the pair or the second token and evaluate the performance on the diagnostic tasks. The trees in the PTB are right-branching. The left token in most cases is closer to the LCA node than the right token, thus we expect that the classifier trained on only the left token has a better overall performance.

6 Results for LCA prediction and chunking

We experimented using four pretrained models. Due to the limited space and the consistency of results, we reported the analysis for the RoBERTa model only in most of the cases. The complete re-

orig.	Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.
.33	Pierre **Berry**, **5,400** years old, **shall** join the board as a nonexecutive director Nov. 29.
.67	**Mesnil Vitulli**, **9.76 beers state-owned**, **ca succeed either** board as a **cash-rich** director **October 213,000**.
orig.	Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.
.33	Mr. Vinken is **growth without** Elsevier **Hills**, **each** Dutch publishing group.
.67	**Tata Helpern s** chairman **plus** Elsevier **Ohls**, **a** Dutch **snaking** group.

Table 1: Examples for the manipulated data. Replaced words are printed in boldface.

	train/test	RoBERTa task	RoBERTa sel.	RoBERTa Δ_{Random}
LCA *concat*	orig./orig.	**82.8**	58.3	23.4
	.33/orig.	81.1	**62.2**	**27.3**
	.33/.33	79.7	56.8	25.6
	.67/orig.	79.3	61.5	25.3
	.67/.67	77.4	59.9	23.1
chunking simple	orig./orig.	**96.0**	13.7	22.2
	.33/orig.	95.3	**15.0**	**26.2**
	.33/.33	94.4	13.1	25.8
	.67/orig.	94.5	14.2	23.6
	.67/.67	93.3	13.6	23.2
chunking detailed	orig./orig.	**91.2**	9.4	32.3
	.33/orig.	90.5	**10.4**	32.8
	.33/.33/	89.2	10.1	33.1
	.67/orig.	89.3	8.8	**36.5**
	.67/.67	87.1	6.7	36.1

Table 2: Results on different datasets. For each task, there are different setups where the model is trained and evaluated on the unchanged treebank (orig.), and two versions with either a third (0.33) or two thirds (0.67) of the tokens replaced. 'task' shows the performance on test set. 'sel.' shows the selectivity (difference to control task), and Δ_{Random} shows the performance difference to the Random BERT model.

sults of all models are shown in appendix sections A.5 and A.6. In the following, we assess the overall performance of the probing classifiers on both linguistic tasks. Then, we evaluate how changing the semantic structure of the data influences the probing classifiers. Lastly, we show some insights into layer-level and neuron-level experiments.

6.1 Overall performance

Tab. 2 shows the performance of the classifiers trained on non-manipulated data using all neurons of the network (orig./orig.). We observed high performance for each diagnostic task. The differences to the baselines show that the knowledge about the task is indeed learned in the representation.

LCA prediction The best results are achieved when concatenating token representations (82.8% acc.). For other representation methods, see App. A.6. We additionally consider single word representations from the word pair as input. The left token representations are better predictors (66.5% acc. on orig./orig.) than those from the right token (40.8%). The large differences between *concat* and the baselines shows that the probe is not memorizing the task, and that information relevant for predicting LCA is acquired during pretraining.

Chunking Chunking detailed (91.2% acc.) is a harder task than chunking simple (96.0%). Although the classifier for the detailed tagset shows relatively low selectivity in comparison to chunking simple, the overall selectivity is high enough to claim that the knowledge about these probing tasks is learned in the representation. The difference to the random BERT model is higher for chunking detailed than for chunking simple, which shows that fine-grained syntactic knowledge is indeed learnt during pretraining.

6.2 Does the probe learn syntax or semantics?

The high performance of the classifiers serves as a proxy to the amount of syntactic knowledge learned in the representations. But Hall Maudslay and Cotterell (2021) argued that due to the presence of semantic cues in the data, high performance of a syntactic probe may not truly reflect the learning of syntax in the model. To investigate this, we manipulated our diagnostic task data (Sec. 5.1) to separate syntax from semantics, and then trained the probing classifiers on the manipulated data.

The second column in Table 2 shows variations of the manipulated data. The classification performance dropped slightly on the diagnostic tasks at 0.33/orig. Moreover, the classifier performed slightly better when evaluating on original data (∗/orig.) compared to manipulated data (such as .33/.33). There are two possible reasons for this: First, the probing classifiers may still rely on semantic knowledge, even when trained on the manipulated data; second, it is possible that the manipulated data contains syntactically ill-formed sentences. Nonetheless, performance and differences to baselines are reasonably high and give good reason to believe that the classifiers are able to extrapolate syntactic knowledge even from semantically nonsensical data. We now proceed with summa-

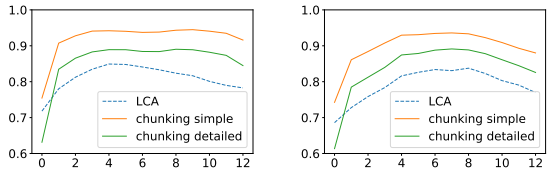


Figure 2: Acc. of layer-wise retraining probing classifiers on RoBERTa (left) and BERT (right) representations for non-manipulated data.

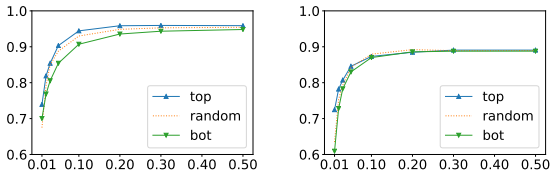


Figure 3: Acc. for simple (left) and detailed (right) chunking tagsets, on top, random and bottom neurons for RoBERTa on non-manipulated data. The horizontal axis plots the fraction of neurons that is selected from all layers.

rizing what our experiments tell about syntactic knowledge in specific layers/neurons of the LM.

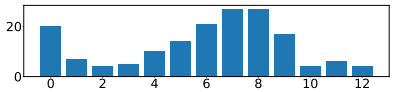
6.3 Layer-wise results

Fig. 2 shows how syntactic knowledge is distributed across all layers. The embedding layer performed worse while middle layers showed the best results, i.e., syntactic information is better represented in the middle layers. The highest layers are more heavily influenced by the pretraining objective, which explains the consistent performance drop across models and tasks on the last layers.

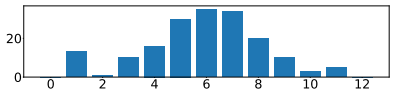
Comparing layer-wise performance with the overall performance, none of the individual layers outperformed the classifier trained on all layers for chunking. In the case of LCA prediction, the performance of layers 4-5 in RoBERTa (6-8 in BERT) are better than the overall performance on all layers. Comparing models, we observed that RoBERTa learns the syntactic knowledge much earlier in the network compared to BERT (see the relatively sharp rise of performance in the lower layers of RoBERTa).

6.4 Neuron-level results

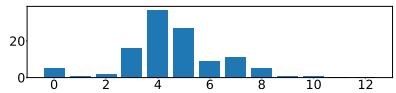
In this section we carry out a more fine-grained neuron-level analysis of the representations. Linguistic correlation analysis (Dalvi et al., 2019a) provides a ranking of neurons with respect to the diagnostic task.



(a) BERT



(b) XLNet



(c) RoBERTa

Figure 4: Spread of neurons relevant for recognizing S in LCA prediction, across layers.

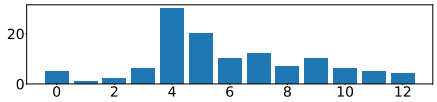


Figure 5: Spread of neurons in RoBERTa across layers that are relevant for identifying NPs in LCA prediction

Minimum Subset of Neurons We evaluated the neuron ranking by training classifiers using top/bottom/random N% neurons. Fig. 3 shows the accuracy curves for the chunking task. The performance margin between different selected neurons is very low. This shows that syntactic information can be extracted from any relatively small subset of neurons i.e. $20-30\%$ of neurons suffice for a probing classifier to perform with the same accuracy as when trained on full representations. Neuron ranking on combined representations does not work well: In some cases, performance on a fraction of randomly selected neurons is worse than performance on the same fraction of neurons ranked as important (see App. A.7).

Distribution of Neurons for LCA prediction Training on subsets of the neurons for LCA prediction is problematic, because the neuron ranking list contains neurons from both token representations. Even though, the distribution of salient neurons across layers yields interesting insights. Fig. 4 presents the spread of top selected neurons for S. As in Sec. 6.3, we found again that top neurons learning syntactic properties come from the middle layers. For 12-layer LMs, we see a trend that neurons from the positional encoding in the embedding layer are utilized to identify distant tokens with LCA S. When comparing the salient neurons selected from each layer, we observe that for identifying S, neurons from the highest layers are less relevant than when identifying NPs (Fig. 5). This

might be due to the comparatively high structural diversity we find in NPs.[2]

Neurons learning Syntax vs. Semantics Comparing the neuron rankings of chunking classifiers trained on the different datasets shows that there is relatively little overlap between the different groups of highest-ranking neurons (see App. A.8). This means that the probing classifiers focus on different neurons when training on manipulated data, compared to the original data. Presumably, the probe focuses more on syntactic and less on semantic information when trained on manipulated data.

7 Reconstructing full Parse Trees

With the insights gained in the previous section, we test if full constituency trees can be linearly separated from LM representations. For this, we train 3 linear classifiers that take as input the concatenated representations of two adjacent tokens and predict the three labels described in Sec. 3. The classifiers for this task take as input not the full LM representation from all layers, but instead the concatenations of every third layer for the 12-layer LMs, and every second layer for the 6-layer LM. This way, the input dimensionality of the classifier is restricted, but the probe can use information from different parts of the LM. The probe is trained (evaluated) on all 38k (5.5k) sentences of the training (development) split of the PTB. We find that the constituency trees reconstructed from different LMs are of high quality (Tab. 3, App. A.9). We achieve a labeled F1 score of 82.6 on the non-manipulated dataset for RoBERTa (80.5 for XLNet, 80.4 for BERT) which is 31 points better than the random BERT baseline. This outperforms the result from Vilares et al. (2020) for BERT by 2.2 points. They also use a linear classifier, but their classifier receives as input only the final layer representation of BERT for the first token in the token pair.

When comparing trees reconstructed from different LMs against each other, we find however that they are quite different. For example, comparing the sentence-level F scores for trees reconstructed from XLNet to those from RoBERTa yields a Pearson correlation of 0.52 only (compared to 0.64 for DistilBERT and BERT, see App. A.10 for the full comparison). This shows that different syntactic properties are linearly separable from the representations of different LMs. These results are not a shortcoming of our probe. We reconstructed parse trees from RoBERTa for the same dataset twice, and a comparison of the two sets of trees gave a labeled F1 score of 96.3. We conclude from this that LMs trained on different data and towards different objectives, such as RoBERTa and XLNet, implicitly make different syntactic generalizations. This insight might have implications for parsing (which is not in the scope of our paper): combining embeddings from both LMs might improve parsing results, compared to using just one LM, as usually done.

	RoBERTa	Δ_{Random}
orig.	**82.6**	31.2
.33/orig.	80.9	31.3
.33/.33	77.8	**31.7**
.67/orig.	78.3	30.4
.67/.67	72.8	28.1

Table 3: Labeled F1 scores for all datasets for reconstructing full parse trees

8 Conclusions

Our experiments have shown that different pretrained LMs encode fine-grained linguistic information that is also present in constituency trees. More specifically, LMs are able to identify properties of different types of constituents, such as S, NP, VP, PP. Good results on the chunking task show that the classifiers are able to combine knowledge about the kind of constituents that a token is part of, and knowledge of the position of a token in the constituent. Using the sequence labeling tasks presented in Vilares et al. (2020), we have shown that full constituency trees are linearly separable from four different pretrained LMs with high quality - even for semantically nonsensical data. In line with Gulordava et al. (2018), we observe a moderate performance drop between performance on the original and nonce dataset. The performance drop is smaller than in Hall Maudslay and Cotterell (2021). who use English pseudowords which the LM has probably never encountered. We use English words whose syntactic and semantic properties are already well-established inside the LM.

In future work, we plan to extend this syntactic probing approach to other languages and other syntactic annotation schemes (for instance Hockenmaier and Steedman, 2007; Evang et al., 2021).

[2] All models are more accurate in LCA prediction when the two tokens are more distant, see App. A.6. Large distance between tokens correlates with LCA nodes close to the root of the syntactic tree, where the LCA often has label S.

Limitations

Our work investigates the question whether syntactic structure is linearly separable from LM representations. However, we make no claim about the question if the syntactic concepts we probe for are actually relevant for LM predictions.

We demonstrate the effectiveness of our method for one high-resource language, namely English. While our methodology is in principle language-agnostic, our study requires high-performing LMs as well as large amounts of annotated data. Both are available for only a relatively small set of languages. More specifically, we found in pilot experiments that supervised probing in general and separating syntactic and semantic knowledge in particular is very data-hungry. While high probe performance on the original data required less than 10k sentences of training data, the performance difference between original and semantically manipulated data shrank with increasing the size of the training data from 10k sentences to 38k sentences. Consequently, in order to have reliable findings, our experiments require large datasets which reflects into the need of sufficient computational resources. For a general discussion of the limitations of supervised probing classifiers, we refer to (Belinkov, 2022).

Acknowledgments

The work presented in this paper was partly funded by the Deutsche Forschungsgemeinschaft (DFG) within the project "Unsupervised Frame Induction: Event Type Hierarchies and Complex Event Types (FInd)". We would like to thank anonymous reviewers for their helpful feedback.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In International Conference on Learning Representations.

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. arXiv preprint arXiv:1610.01644.

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

Yonatan Belinkov, Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and James Glass. 2020. On the linguistic representational power of neural machine translation models. Computational Linguistics, 46(1).

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs encode soft hierarchical syntax. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 14–19, Melbourne, Australia. Association for Computational Linguistics.

Boli Chen, Yao Fu, Guangwei Xu, Pengjun Xie, Chuanqi Tan, Mosha Chen, and Liping Jing. 2021. Probing BERT in hyperbolic spaces.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, D. Anthony Bau, and James Glass. 2019a. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).

Fahim Dalvi, Abdul Rafae Khan, Firoj Alam, Nadir Durrani, Jia Xu, and Hassan Sajjad. 2022. Discovering latent concepts learned in BERT. In International Conference on Learning Representations.

Fahim Dalvi, Avery Nortonsmith, D Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, and James Glass. 2019b. Neurox: A toolkit for analyzing individual neurons in neural networks. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06), Genoa, Italy. European Language Resources Association (ELRA).

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.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. Analyzing individual neurons in pre-trained language models. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

Kilian Evang, Tatiana Bladier, Laura Kallmeyer, and Simon Petitjean. 2021. Bootstrapping Role and Reference Grammar treebanks via Universal Dependencies. In Proceedings of Universal Dependencies Workshop 2021 (UDW 2021). To appear.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1195–1205, New Orleans, Louisiana. 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.

Rowan Hall Maudslay, Josef Valvoda, Tiago Pimentel, Adina Williams, and Ryan Cotterell. 2020. A tale of a probe and a parser. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7389–7395, Online. Association for Computational Linguistics.

John Hewitt, Kawin Ethayarajh, Percy Liang, and Christopher Manning. 2021. Conditional probing: measuring usable information beyond a baseline. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 1626–1639, Online and Punta Cana, Dominican Republic. 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.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. Computational Linguistics, 33(3):355–396.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. Journal of Artificial Intelligence Research, 61:907–926.

Huayang Li, Lemao Liu, Guoping Huang, and Shuming Shi. 2020. On the branching bias of syntax extracted from pre-trained language models. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4473–4478, Online. 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.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. Proceedings of the National Academy of Sciences, 117(48):30046–30054.

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.

Benjamin Newman, Kai-Siang Ang, Julia Gong, and John Hewitt. 2021. Refining targeted syntactic evaluation of language models. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3710–3723, Online. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. In Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc.

Hassan Sajjad, Nadir Durrani, and Fahim Dalvi. 2022a. Neuron-level Interpretation of Deep NLP Models: A Survey. Transactions of the Association for Computational Linguistics.

Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Firoj Alam, Abdul Rafae Khan, and Jia Xu. 2022b. Analyzing encoded concepts in transformer language models. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL '22, Seattle, Washington, USA. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In International Conference on Learning Representations.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. CoRR, abs/1804.07461.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4166–4176, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc.

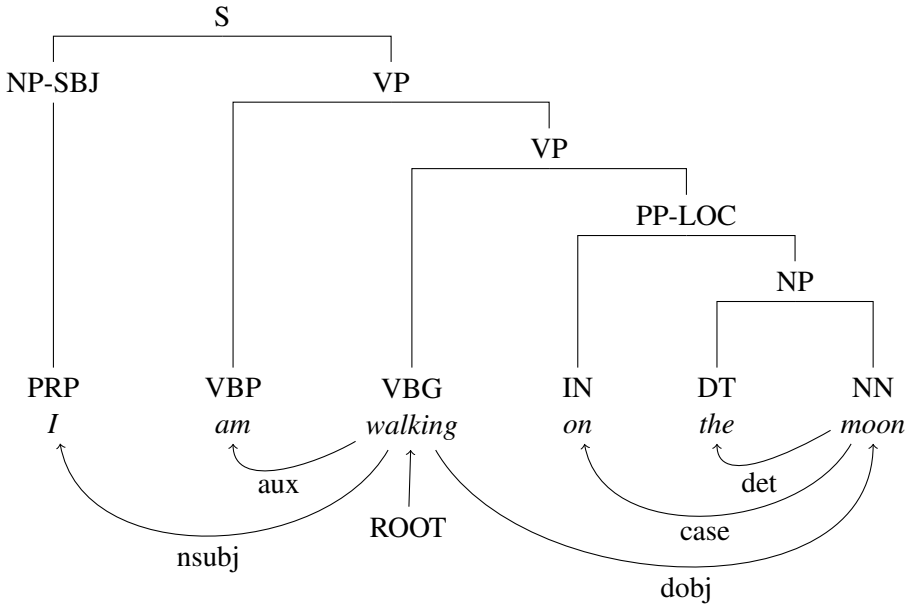
Kelly Zhang and Samuel Bowman. 2018. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 359–361, Brussels, Belgium. Association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

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 67(2):301–320.

A Appendix

A.1 Constituency and Dependency Trees

Here, you see the constituency tree (above the sentence) and the dependency tree (below) for a simple sentence, following the annotation scheme of the PTB and its dependency version. More fine-grained hierarchical structure is encoded in the node labels of the constituency structure. For example, the constituency tree assigns the label VP to the spans *walking on the moon* and *am walking on the moon*. However, these spans are not reflected by particular entities in the dependency tree.



A.2 Control tasks and Selectivity

To ascertain that our structural probe learns complex structural generalization and does not memorize structural information from the task, following Hewitt and Liang (2019), we design two control tasks (CT). This consists in randomizing the labels of syntactic categories, creating a new dataset. In our implementation, the distribution of randomly selected labels approximates the class distribution in the training set. For chunking, a numerical target label is assigned randomly to each word type. We slightly modify this baseline to be able to handle token pairs. For LCA prediction, we assign a random numeric label to each pair of word types. For example, the word pair (*the*, *sold*) in Fig. 1 always receives the label 1, regardless of the context where it occurs. And since the dataset contains more word type pairs than individual word types, the control task setup is inherently more complex for LCA prediction than the one for chunking.

A.3 Token combination methods

For LCA prediction, w conducted experiments using three different combination methods: (i) concatenation ($concat(x_i, x_j) \in \mathbb{R}^{2r}$, see Sec.4); (ii) element-wise average $avg(x_i, x_j) \in \mathbb{R}^r$; and (iii) element-wise signed absolute maximum max_s of two scalars m, n. $max_s(w_i, w_j) \in \mathbb{R}^r$ prefers strong positive and negative neuron activations while keeping the sign of the activation value in the combined vector:

$$max_s(m, n) = \begin{cases} m & \text{if } |m| > |n| \\ n & \text{otherwise} \end{cases} \tag{2}$$

Averaging is the most lossy combination: Large positive or negative neuron activations are canceled out if they are not shared between both vectors. Concatenation is the only lossless combination. The concatenation result $concat(x_i, x_j) \in \mathbb{R}^{2r}$ has a larger dimensionality than the other combination methods.

A.4 Label distributions for the different probing tasks

	B	I	E	S
VP	13.77%	1.24%	0.07%	0.96%
NP	12.53%	8.83%	12.27%	4.05%
PP	9.73%	0.25%	0.00%	0.03%
NP-SBJ	2.96%	1.73%	3.11%	2.46%
SBAR	1.14%	0.10%	0.00%	
ADJP	0.82%	0.21%	0.50%	0.38%
QP	0.51%	0.75%	0.85%	0.00%
ADVP	0.35%	0.04%	0.27%	1.57%
NP-TMP	0.28%	0.07%	0.29%	0.14%
S	0.25%	0.29%		
NP-PRD	0.18%	0.18%	0.18%	0.03%
NP-ADV	0.16%	0.01%	0.16%	0.01%
NP-LGS	0.12%	0.16%	0.13%	0.02%
NP-EXT	0.08%	0.01%	0.03%	0.03%
NX	0.06%	0.06%	0.06%	0.03%
NAC	0.05%	0.04%	0.00%	0.00%
WHPP	0.04%			
CONJP	0.04%	0.02%	0.04%	0.00%
WHNP	0.03%	0.01%	0.04%	0.75%
UCP	0.03%	0.06%	0.02%	
SQ	0.02%	0.01%		
NP-LOC	0.02%	0.00%	0.02%	0.04%
NP-TTL	0.02%	0.01%	0.02%	0.01%
NP-HLN	0.02%	0.02%	0.01%	0.00%
SINV	0.01%	0.00%		
FRAG	0.01%	0.01%	0.00%	
WHADVP	0.01%	0.00%	0.01%	0.18%
LST	0.01%			
WHADJP	0.00%	0.00%	0.00%	
X	0.00%	0.00%	0.00%	0.00%
SBARQ	0.00%	0.00%		
NP-MNR	0.00%	0.00%	0.00%	
NP-CLR	0.00%		0.00%	0.02%
INTJ	0.00%	0.00%	0.00%	0.01%
NP-TPC	0.00%	0.00%	0.00%	0.00%
NP-VOC	0.00%		0.00%	0.00%
NP-DIR	0.00%		0.00%	
ADVP—PRT				0.00%

(a) Label distribution for the detailed chunking tagset. Empty cells indicate that the combination of chunking label and phrase label is not present in the training data. Cells rounded to 0.00% indicate labels that are exceptionally rare in the training data.

S	39.20%
VP	25.37%
NP	22.66%
PP	5.35%
SBAR	2.65%
SINV	2.51%
ADJP	0.68%
ADVP	0.38%
QP	0.37%
FRAG	0.29%
UCP	0.13%
SQ	0.09%
WHNP	0.08%
NX	0.08%
SBARQ	0.05%
PRN	0.02%
WHADVP	0.02%
NAC	0.02%
CONJP	0.01%
WHPP	0.01%
X	0.01%
RRC	0.00%
INTJ	0.00%
LST	0.00%
ADVP—PRT	0.00%
WHADJP	0.00%
PRT—ADVP	0.00%
PRT	0.00%

(b) Label distribution for LCA prediction.

B	I	E	S	PCT
43.3%	14.1%	18.1%	10.7%	13.8%

(c) `B`: beginning, `I`: inside, `E`: end, `S`: Single, `PCT`: punctuation: punctuation is not considered for evaluation

Table 4: Label distributions for the different probing tasks

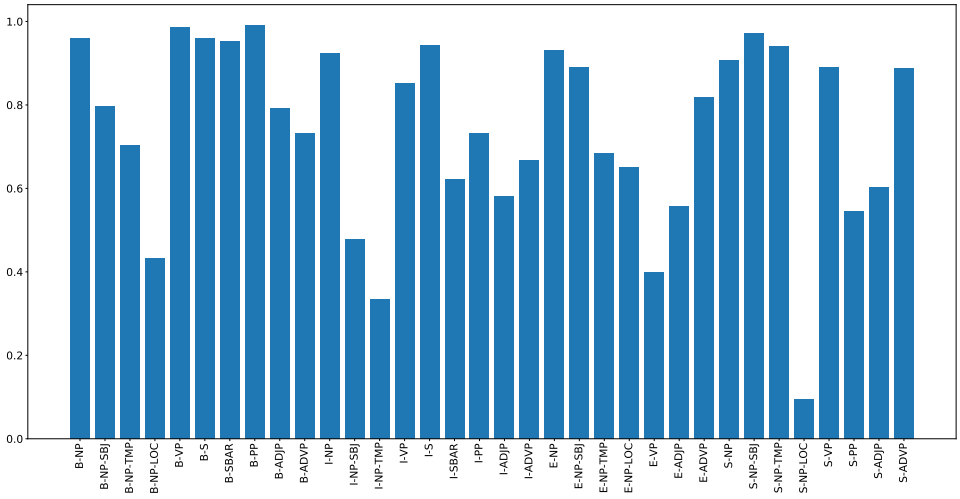
A.5 Chunking results

task	train/test	DistilBERT		BERT		XLNet	
		task	sel.	task	sel.	task	sel.
chunking simple	orig./orig.	95.3	14.1	95.2	13.4	94.3	17.1
	.33/orig.	94.1	14.9	94.4	14.9	93.9	18.7
	.33/.33	93.7	15.2	93.7	14.5	92.8	17.8
	.67/orig.	93.6	14.4	93.4	14.0	92.6	17.8
	.67/.67	92.6	14.7	92.4	16.0	91.3	17.1
chunking detailed	orig./orig.	90.3	8.7	90.6	9.4	91.1	16.5
	.33/orig.	89.3	10.4	89.3	10.7	89.6	15.9
	.33/.33/	88.1	10.6	87.9	10.7	87.8	14.2
	.67/orig.	88.5	9.7	88.1	9.7	87.9	13.8
	.67/.67	86.3	8.5	86.1	8.8	86.4	12.8

(a) Results for chunking experiments with DistilBERT, BERT and XLNet

	B	I	E	S
B	108594	1238	65	1026
I	1683	33345	1504	217
E	119	836	46980	570
S	999	222	662	24597

(b) Confusion matrix for chunking experiments with RoBERTa.



(c) Accuracy per category for the most frequent labels in the detailed tagset. All beginnings of frequent constituents are recognized with high accuracy. The classifier is also able to distinguish between different kinds of NPs, such as `NP` without further specification, subject NPs (`NP-SBJ`) or temporal and local NPs (`NP-TMP`,`NP-LOC`)

Table 5: Results for chunking

A.6 LCA prediction results

task	train/test	RoBERTa task	RoBERTa sel.	DistilBERT task	DistilBERT sel.	BERT task	BERT sel.	XLNet task	XLNet sel.
concat	orig./orig.	82.9	58.3	81.0	55.2	81.4	59.0	83.1	67.8
	.33/orig.	81.1	62.2	79.4	52.9	80.7	60.9	81.2	68.0
	.33/.33	79.7	56.8	78.0	51.8	79.5	56.6	79.7	61.8
	.67/orig.	79.3	61.5	78.6	58.2	79.5	65.2	80.9	59.9
	.67/.67	77.4	59.9	75.9	53.3	77.5	61.5	78.2	61.4
max_s	orig./orig.	69.3	42.7	68.6	40.1	63.4	39.7	70.5	54.5
	.33/orig.	68.1	46.3	69.0	42.0	61.9	41.4	66.0	48.2
	.33/.33	66.5	45.2	66.4	36.2	60.3	39.6	60.1	43.2
	.67/orig.	65.8	45.0	64.5	41.6	61.0	43.6	59.2	37.8
	.67/.67	62.3	43.9	63.3	38.0	59.0	39.4	55.1	32.0
avg	orig./orig.	63.9	36.9	66.8	38.8	62.3	38.4	57.5	42.3
	.33/orig.	64.3	37.2	66.1	38.7	63.9	39.3	58.9	45.0
	.33/.33	62.0	36.8	64.1	31.3	61.8	37.1	54.8	34.5
	.67/orig.	63.4	37.2	63.4	36.5	62.9	41.7	57.3	38.4
	.67/.67	59.6	36.5	61.3	34.0	59.4	37.2	51.5	27.1

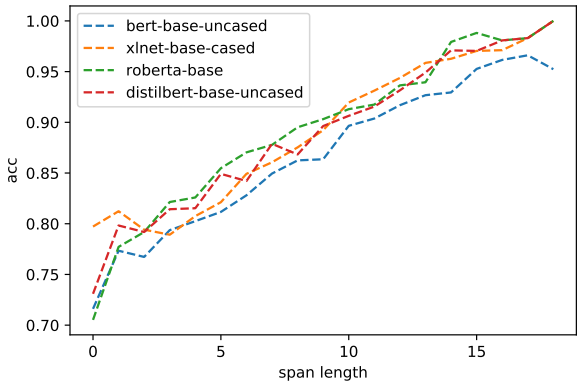
(a) LCA prediction results. The performance gains of *concat* wrt. *avg* and max_s are not matched by higher performance in the control task for *concat*. Thus *concat* shows not only the best task performance but also the highest selectivity for LCA prediction.

	NP	VP	S	SBAR	PP	ADJP	ADVP
NP	87681	4229	7716	508	2299	337	393
VP	7817	99200	8389	1089	2228	121	755
S	5503	3776	159212	1575	691	93	346
SBAR	193	286	168	5849	212	10	4
PP	1754	848	331	189	18974	187	91
ADJP	629	384	59	5	4	1861	153
ADVP	50	104	38	20	0	1	2035

(b) Confusion matrix for LCA prediction for the most frequent constituents labels for RoBERTa when trained and evaluated on non-manipulated data. The columns represent predicted values, rows represent actual values. Some categories are better represented in the probing classifiers than others. For example, prepositional phrases are recognized quite reliably, but adjectival phrases are confused for `VP`s and `NP`s in a number of cases. `NP`s are frequently confused with all other categories. The reason might be that a variety of different phenomena are collected under `NP`, such as appositions and relative clauses.

	left	right
RoBERTa	66.5	40.8
DistilBERT	68.0	43.6
BERT	64.8	40.9
XLNet	62.1	40.7

(c) Results for single-token baseline on LCA prediction



(d) For LCA prediction, all models are more accurate when the distance between two tokens is higher

Figure 6: Results for LCA prediction

A.7 Neuron-level results for LCA prediction

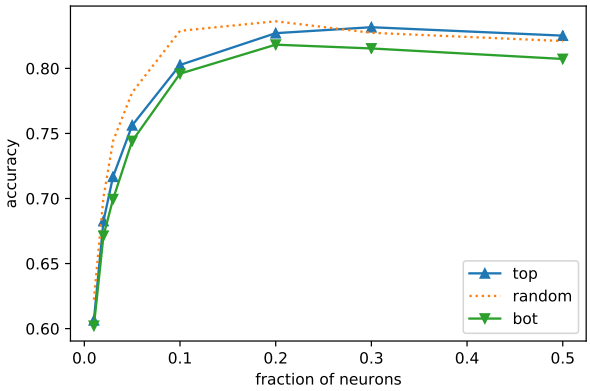


Figure 7: Accuracy for RoBERTa probes when trained on fractions of top, random or bottom neurons identified by linguistic correlation analysis (Durrani et al., 2020). The classifiers are trained on LCA prediction, using concatenated representations. Fractions on the x-axis refer to the dimensions of concatenated representations. Thus, the absolute number of considered neurons is twice as high as in the respective plots for chunking in Fig. 3.

A.8 Neuron orderings for different datasets

% neurons	orig. & .33	orig. & .67	.33 & .67
1%	21.2%	16.2%	15.2%
2%	18.1%	15.6%	23.1%
3%	17.4%	12.7%	20.4%
5%	19.8%	16.6%	21.2%
10%	25.4%	20.7%	26.6%
20%	35.7%	29.6%	34.4%
30%	42.6%	38.1%	43.7%
50%	59.1%	56.4%	59.4%

Table 6: Overlap of fractions of top neurons for chunking (simple tagset) when classifiers are trained on different datasets. For each dataset, a neuron ranking list is obtained. This table shows the size of the fraction of neurons that are ranked among the most salient $x\%$ of neurons for two different datasets. For example, 21.2% of the 1% most salient neurons for the original dataset are also among the 1% most salient neurons for the .33 dataset.

A.9 Results for parse tree reconstructions from all language models

	RoBERTa	BERT	DistilBERT	XLNet	Random BERT
orig./orig.	82.58	80.42	79.88	80.52	51.36
.33/orig.	80.88	77.73	77.65	78.99	49.60
.33/.33	77.84	73.97	74.09	75.23	46.13
.67/orig.	78.30	74.63	74.81	75.37	47.95
.67/.67	72.77	69.63	69.72	69.91	44.71

Table 7: Labeled F1 scores for all datasets and models for reconstructing full parse trees

A.10 Comparing parse trees reconstructed from different models

	RoBERTa vs. XLNet	RoBERTa vs. BERT	BERT vs. XLNet	DistilBERT vs. BERT	DistilBERT vs. RoBERTa	DistilBERT vs. XLNet
orig.	81.83	82.80	80.39	83.17	82.34	80.19
.33/orig.	79.71	80.21	77.60	81.60	80.53	77.88
.33/.33	76.02	76.52	73.99	78.72	76.78	74.30
.67/orig.	76.54	77.08	74.07	78.68	77.78	74.40
.67/.67	71.75	72.83	69.56	75.01	73.45	69.89

Table 8: Labeled F scores for comparing constituent trees predicted by different models against each other. The comparison of trees predicted by DistilBERT and BERT yields the highest F scores, hence trees reconstructed from these models are most similar.

	RoBERTa	BERT	DistilBERT
BERT	0.59		
DistilBERT	0.57	0.64	
XLNet	0.52	0.52	0.51

Table 9: Pearson correlation of sentence-level F scores for different LMs on original data. The correlations between F scores are lowest when comparing XLNet to BERT, RoBERTa and DistilBERT. The correlations are highest between the latter three models.

A.11 Examples for reconstructed parse trees

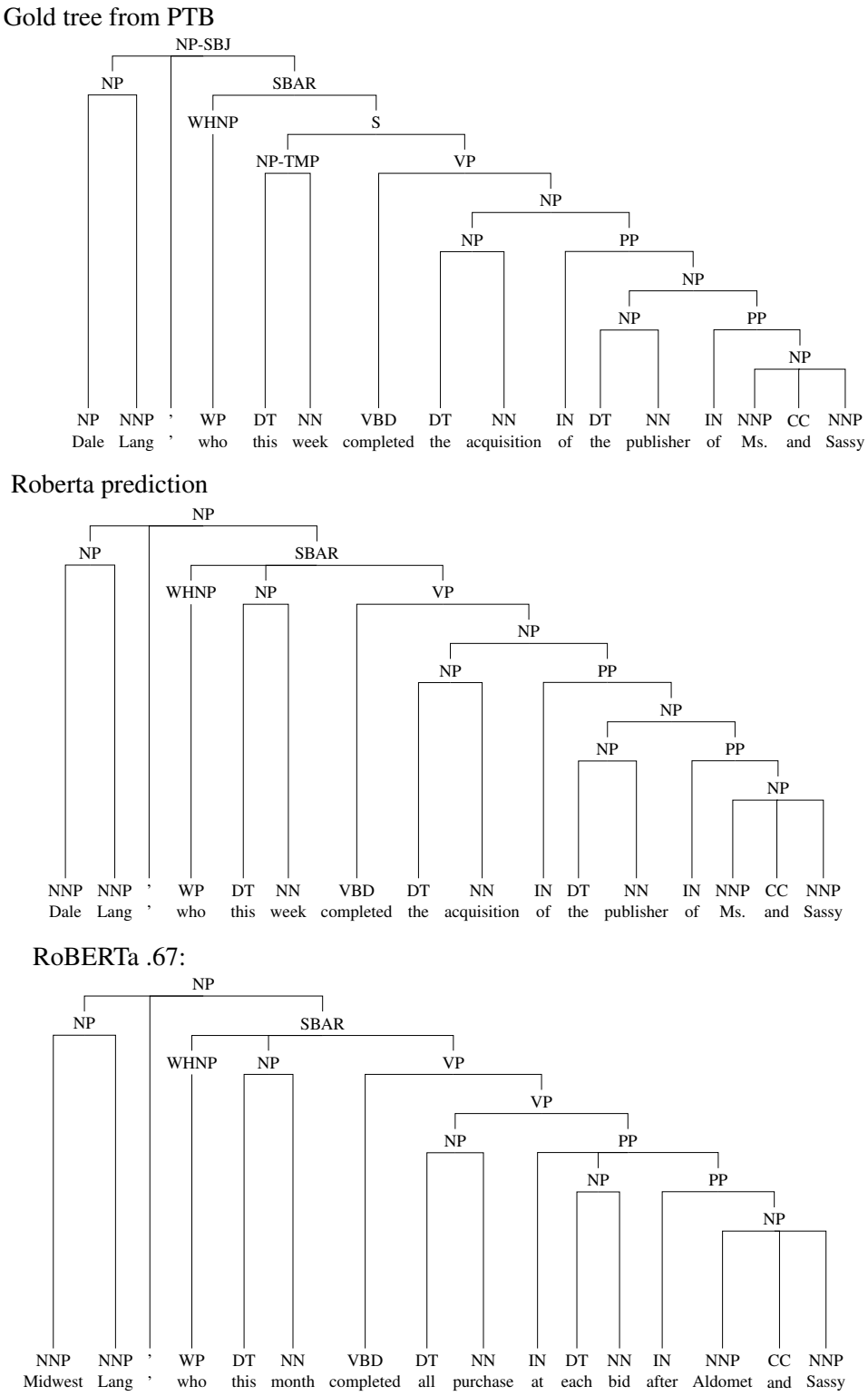
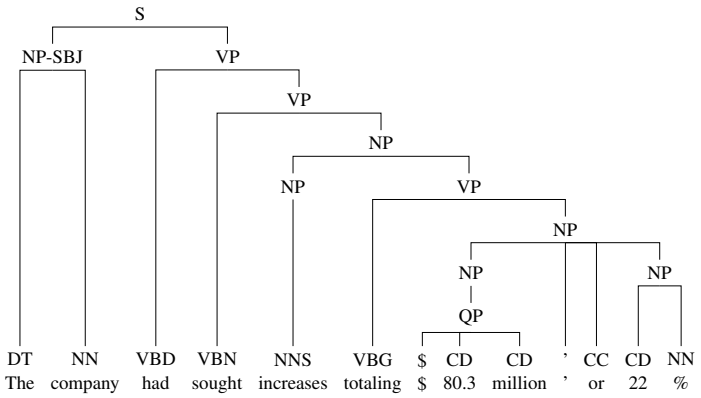
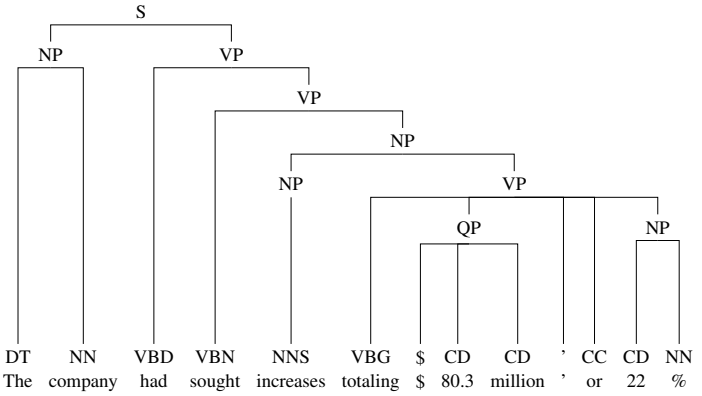


Figure 8: Example for reconstructing parse trees: The subject phrase of sentence 23 from the development set. In the RoBERTa prediction, the SBAR and S nodes of the relative clause are conflated to an SBAR node. In the RoBERTa prediction with .67 of the tokens replaced, the short chunks are correctly recognized, but the VP of the relative clause is structured differently.

Gold tree from PTB



RoBERTa orig:



RoBERTa .67:

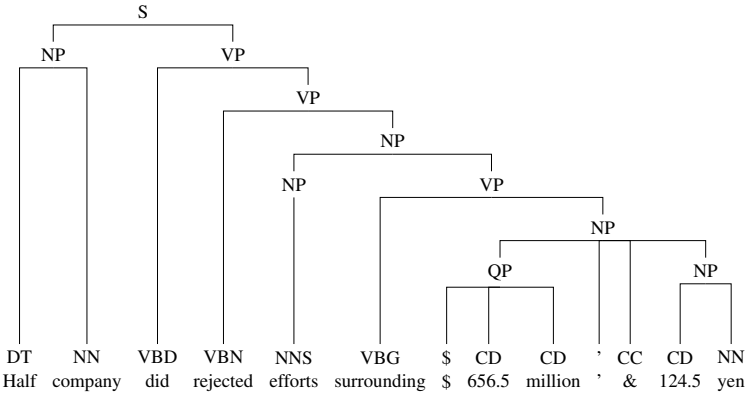


Figure 9: Example for reconstructing parse trees: Sentence 4 from the development set. For the RoBERTa prediction, the unary leaf chain NP $\rightarrow$ QP is not predicted because it cannot be predicted by our version of the sequence labeling algorithm. Apart from that, the *or*-NP is not recognized as a single NP. In the RoBERTa prediction for the sentence with .67 of the tokens replaced, the VP is recognized correctly even though there is an agreement mismatch (*did rejected*). Diverging from the prediction for original data, the *or*-NP is reconstructed correctly

A.12 Computing infrastructure and experiment runtime

All experiments where run on an Nvidia Titan XP GPU. For a 12-layer model, the full set of chunking experiments takes 4 hours. This includes extracting the full neuron activation values for the training and evaluation data, all control task experiments and all experiments on the linguistic task. The full set of LCA prediction experiments for a 12-layer model takes around 15 hours when using all ways of combining input representations ($concat$, avg, max_s). Compared to the chunking experiments, an additional step is combining the activation values of token pairs. The full set of experiments for reconstructing parse trees takes around 3 hours when using a 12-layered language models and all datasets. Experiments with the 6-layer model DistilBERT take half the time. The reported time includes time where the GPU itself is not active, e.g. times where representations are combined and written to harddrive. All experiments were powered with electricity from renewable energy sources.