Do Syntactic Probes Probe Syntax? Experiments with Jabberwocky Probing

Rowan Hall Maudslay[1,2] **Ryan Cotterell**[1,2]
[1]University of Cambridge [2]ETH Zürich
rh635@cam.ac.uk ryan.cotterell@inf.ethz.ch

Abstract

Analysing whether neural language models encode linguistic information has become popular in NLP. One method of doing so, which is frequently cited to support the claim that models like BERT encode syntax, is called probing; probes are small supervised models trained to extract linguistic information from another model's output. If a probe is able to predict a particular structure, it is argued that the model whose output it is trained on must have implicitly learnt to encode it. However, drawing a generalisation about a model's linguistic knowledge about a specific phenomena based on what a probe is able to learn may be problematic: in this work, we show that semantic cues in training data means that syntactic probes do not properly isolate syntax. We generate a new corpus of semantically nonsensical but syntactically well-formed Jabberwocky sentences, which we use to evaluate two probes trained on normal data. We train the probes on several popular language models (BERT, GPT-2, and RoBERTa), and find that in all settings they perform worse when evaluated on these data, for one probe by an average of 15.4 UUAS points absolute. Although in most cases they still outperform the baselines, their lead is reduced substantially, e.g. by 53% in the case of BERT for one probe. This begs the question: what empirical scores constitute knowing syntax?

1 'Twas Brillig, and the Slithy Toves

Recently, unsupervised language models like BERT (Devlin et al., 2019) have become popular within natural language processing (NLP). These pre-trained sentence encoders, known affectionately as BERToids (Rogers et al., 2020), have pushed forward the state of the art in many NLP tasks. Given their impressive performance, a natural question to ask is whether models like these implicitly learn to encode linguistic structures, such as part-of-speech tags or dependency trees.

There are two strains of research that investigate this question. On one hand, **stimuli-analysis** compares the relative probabilities a language model assigns to words which could fill a gap in a cloze-style task. This allows the experimenter to test whether neural models do well at capturing specific linguistic phenomena, such as subject–verb agreement (Linzen et al., 2016; Gulordava et al., 2018) or negative-polarity item licensing (Marvin and Linzen, 2018; Warstadt et al., 2019). Another strain of research directly analyses the neural network's representations; this is called **probing**. Probes are supervised models which attempt to predict a target linguistic structure using a model's representation as its input (e.g. Alain and Bengio, 2017; Conneau et al., 2018; Hupkes and Zuidema, 2018); if the probe is able to perform the task well, then it is argued that the model has learnt to implicitly encode that structure in its representation.[1]

Work from this inchoate probing literature is frequently cited to support the claim that models like BERT encode a large amount of syntactic knowledge. For instance, consider the two excerpts below demonstrating how a couple of syntactic probing papers have been interpreted:[2]

> *[The training objectives of BERT/GPT-2/XLNet] have shown great abilities to capture dependency between words and syntactic structures (Jawahar et al., 2019)* (Tian et al., 2020)

> *Further work has found impressive degrees of syntactic structure in Transformer encodings (Hewitt and Manning, 2019)* (Soulos et al., 2020)

[1]Methods which analyse stimuli are also sometimes termed 'probes' (e.g. Niven and Kao, 2019), but in this paper we use the term to refer specifically to supervised models.

[2]Jawahar et al. (2019) and Hewitt and Manning (2019) are more reserved about their claims; these examples merely show how such work is frequently interpreted, regardless of intent.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 124–131
June 6–11, 2021.

Our position in this paper is simple: we argue that the literature on syntactic probing is methodologically flawed, owing to a conflation of syntax with semantics. We contend that no existing probing work has rigorously tested whether BERT encodes syntax, and *a fortiori* this literature should not be used to support this claim.

To investigate whether syntactic probes actually probe syntax (or instead rely on semantics), we train two probes (§4) on the output representations produced by three pre-trained encoders on normal sentences—BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), and RoBERTa (Liu et al., 2019). We then evaluate these probes on a novel corpus of syntactically well-formed sentences made up of pseudowords (§3), and find that their performance drops substantially in this setting: on one probe, the average BERToid UUAS is reduced by 15.4 points, and on the other the relative advantage that BERT exhibits over a baseline drops by 53%. This suggests that the probes are leveraging statistical patterns in distributional semantics to aide them in the search for syntax. According to one of the probes, GPT-2 falls behind a simple baseline, but in some cases the leads remains substantial, e.g. 20.4 UUAS points in the case of BERT. We use these results not to draw conclusions about any BERToids' syntactic knowledge, but instead to urge caution when drawing conclusions from probing results. In our discussion, we contend that evaluating BERToids' syntactic knowledge requires more nuanced experimentation than simply training a syntactic probe as if it were a parser (Hall Maudslay et al., 2020), and call for the separation of syntax and semantics in future probing work.

2 Syntax and Semantics

When investigating whether a particular model encodes syntax, those who have opted for stimuli-analysis have been careful to isolate syntactic phenomena from semantics (Marvin and Linzen, 2018; Gulordava et al., 2018; Goldberg, 2019), but the same cannot be said of most syntactic probing work, which conflates the two. To see how the two can be separated, consider the famous utterance of Chomsky (1957):

(1) Colourless green ideas sleep furiously

whose dependency parse is give in Figure 1. Chomsky's point is that (1) is semantically nonsensical, but syntactically well formed.

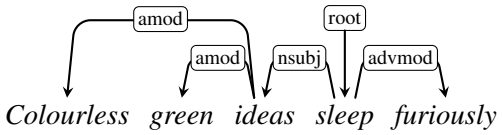


Figure 1: Chomsky's classic, albeit with the spelling corrected.

Syntactic probes are typically evaluated on real-world data, not on Chomsky-style sentences of (1)'s ilk. The same is true for parsers, but from a *machine-learning point of view* this is not problematic, since the goal of a statistical parser is to parse well the data that one may encounter in the real world. The probing literature, however, is inherently making a epistemological claim: whether BERT knows syntax.[3] Indeed, we already know that BERT significantly improves the performance of statistical parsing models on real-world data (Zhou and Zhao, 2019); there is no reason to develop specialist probes to reinforce that claim. As probing consider a scientific qustion, it follows that the probing literature needs to consider syntax from *a linguistic point of view* and, thus, it requires a linguistic definition of syntax. At least in the generative tradition, it taken as definitional that grammaticality, i.e. syntactic well-formedness, is distinct from the meaning of the sentence. It is this distinction that the nascent syntactic probing literature has overlooked.

3 Generating Jabberwocky Sentences

To tease apart syntax and semantics when evaluating probes, we construct a new evaluation corpus of syntactically valid English **Jabberwocky sentences**, so called after Carroll (1871) who wrote verse consisting in large part of pseudowords (see App. A). In written language, a pseudoword is a sequence of letters which looks like a valid word in a particular language (usually determined by acceptability judgments), but which carries with it no lexical meaning.

For our Jabberwocky corpus, we make use of the ARC Nonword Database, which contains 358,534 monosyllabic English pseudowords (Rastle et al., 2002). We use a subset of these which were filtered

[3]This is not an engineering claim because the NLP engineer is unlikely to care whether BERT's representations encode syntactic structure—they just care about building reliable models that perform well on real data. An open question, however, is whether representations *require* a notion of syntax to properly generalise; this is not addressed in our work.

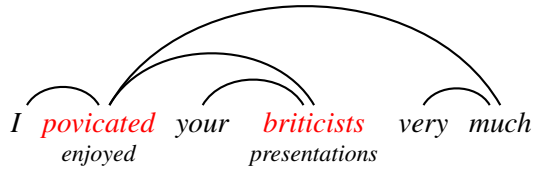


Figure 2: An unlabeled undirected parse from the EWT treebank, with Jabberwocky substitutions in red.

out then manually validated for high plausibility by Kharkwal (2014). We conjugate each of these words using hand-written rules assuming they obey the standard English morphology and graphotactics. This results in 1361 word types—a total of 2377 varieties when we annotate these regular forms with several possible fine-grained part-of-speech realisations.

To build sentences, we take the test portion of the English EWT Universal Dependency (UD; Nivre et al., 2016) treebank and substitute words (randomly) with our pseudowords whenever we have one available with matching fine-grained part-of-speech annotation.[4] Our method closely resembles Kasai and Frank (2019), except they do so to analyse parsers in place of syntactic probes. An example of one of our Jabberwocky sentences is shown in Figure 2, along with its unlabeled undirected parse (used by the probes) which is taken from the vanilla sentence's annotation in the treebank.

4 Two Syntactic Probes

A **syntactic probe** is a supervised model trained to predict the syntactic structure of a sentence using representations produced by another model. The main distinction between syntactic probes and dependency parsers is one of researcher intent—probes are not meant to best the state of the art, but are a visualisation method (Hupkes and Zuidema, 2018). As such, probes are typically minimally parameterised so they do not "dig" for information (but see Pimentel et al., 2020). If a syntactic probe performs well using a model's representations, it is argued that that model implicitly encodes syntax.

[4]More specifically, for nouns we treat elements annotated (in UD notation) with `Number=Sing` or `Number=Plur`; for verbs we treat `VerbForm=Inf`, `VerbForm=Fin | Mood=Ind | Number=Sing | Person=3 | Tense=Pres`, `VerbForm=Fin | Mood=Ind | Tense=Pres`, or `VerbForm=Part | Tense=Pres`; for adjectives and adverbs we treat `Degree=Cmp` or `Degree=Sup`, along with unmarked. These cases cover all regular forms in the EWT treebank.

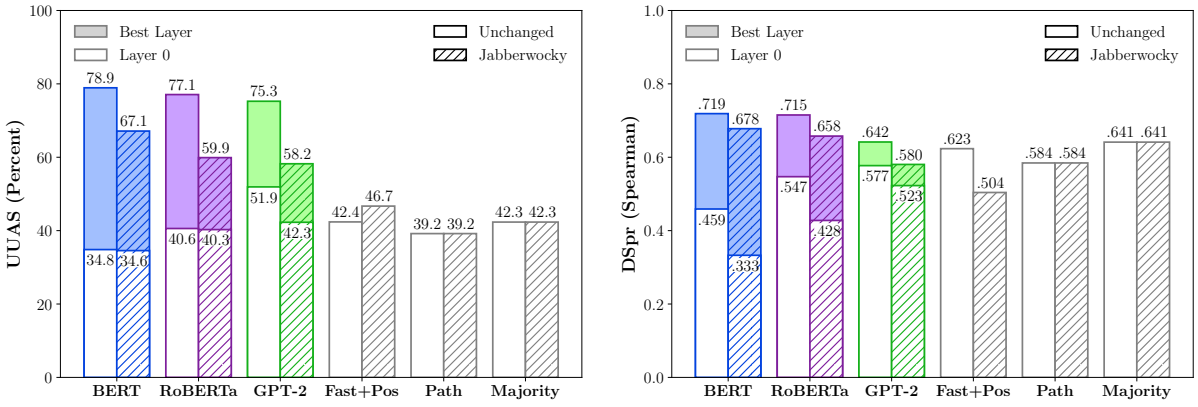
Here we briefly introduce two syntactic probes, each designed to learn the **syntactic distance** between a pair of words in a sentence, which is the number of steps between them in an undirected parse tree (example in Figure 2). Hewitt and Manning (2019) first introduced syntactic distance, and propose the **structural probe** as a means of identifying it; it takes a pair of embeddings and learns to predict the syntactic distance between them. An alternative to the structural probe which learns parameters for the same function is a structured perceptron dependency parser, originally introduced in McDonald et al. (2005), and first applied to probing in Hall Maudslay et al. (2020). Here we call this the **perceptron probe**. Rather than learning syntactic distance directly, the perceptron probe instead learns to predict syntactic distances such that the minimum spanning tree that results from a sentence's predictions matches the gold standard parse tree. The difference between these probes is subtle, but they optimise for different metrics—this is reflected in our evaluation in §5.

5 Hast Thou [Parsed] the Jabberwock?

We train the probes on normal UDs, then evaluate them on Jabberwocky sentences; if the probes are really learning to extract syntax, they should perform just as well in the Jabberwocky setting.

5.1 Experimental Setup

Models to Probe We probe three popular Transformer (Vaswani et al., 2017) models: **BERT** (Devlin et al., 2019), **GPT-2** (Radford et al., 2019), and **RoBERTa** (Liu et al., 2019). For all three we use the 'large' version. We train probes on the representations at multiple layers, and choose whichever layers result in the best performance on the development set. For each Transformer model, we also train probes on the layer 0 embeddings; we can treat these layer 0 embeddings as baselines since they are uncontextualised, with knowledge only of a single word and where it sits in a sentence, but no knowledge of the other words. As an additional baseline representation to probe, we use FastText embeddings (Bojanowski et al., 2017) appended with BERT position embeddings (**Fast+Pos**). We emphasise that none of these baselines can be said to encode anything about syntax (in a linguistic sense), since they are uncontextualised. Training details of these models and baselines can be found in App. B.



(a) UUAS results from the Perceptron Probe

(b) DSpr results from the Structural Probe

Figure 3: How the models fair when the probes are evaluated on unchanged sentences vs. the Jabberwocky.

Additional Simple Baselines In addition to the baseline representations which we probe, we compute two even simpler baselines, which ignore the lexical items completely. The first simply connects each word to the word next to it in a sentence (**Path**). The second returns, for a given sentence length, the tree which contains the edges occurring most frequently in the training data (**Majority**), which is computed as follows: first, we subdivide the training data into bins based on sentence length. For each sentence length n, we create an undirected graph G_n with n nodes, each corresponding to a different position in the sentence. The edges are weighted according to the number of times they occur in the training data bin which contains sentences of length n. The 'majority tree' of sentence length n is then computed by calculating the maximum spanning tree over G_n, which can be done by negating the edges, then running Prim's algorithm. For $n > 40$, we use the Path baseline's predictions, owing to data sparsity.

Metrics As mentioned in §4, the probes we experiment with each optimise for subtly different aspects of syntax; we evaluate them on different metrics which reflect this. We evaluate the structural probe on DSpr, introduced in Hewitt and Manning (2019)—it is the Spearman correlation between the actual and predicted syntactic distances between each pair of words. We evaluate the perceptron probe using the unlabeled undirected attachment score (UUAS), which is the percentage of correctly identified edges. These different metrics reflect differences in the probe designs, which are elaborated in Hall Maudslay et al. (2020).

5.2 Results

Figure 3 shows the performance of the probes we trained, when they are evaluated on normal test data (plain) versus our specially constructed Jabberwocky data (hatched). Recall that the test sets have identical sentence–parse structures, and differ only insofar as words in the Jabberwocky test set have been swapped for pseudowords.[5] For each BERToid, the lower portion of its bars (in white) shows the performance of its layer 0 embeddings, which are uncontextualised and thus function as additional baselines.

All the probes trained on the BERToids perform worse on the Jabberwocky data than on normal data, indicating that the probes rely in part on semantic information to make syntactic predictions. This is most pronounced with the perceptron probe: in this setting, the three BERToids' scores dropped by an average of 15.4 UUAS points. Although they all still outperform the baselines under UUAS, their advantage is less pronounced, but in some cases it remains high, e.g. for BERT the lead is 20.4 points over the Fast+Pos baseline. With the structural probe, BERT's lead over the simple Majority baseline is reduced from 0.078 to 0.037 DSpr, and RoBERTa's from 0.074 to 0.017—reductions of 53% and 77%, respectively. GPT-2 falls behind the baselines, and performs worse than even the simple Path predictions (0.580 compared to 0.584).

[5] This is why the Path and Majority baselines, which do not condition on the lexical items in a sentence, have identical scores on both datasets.

5.3 Discussion

Is BERT still the syntactic wunderkind we had all assumed? Or do these reductions mean that these models can no longer be said to encode syntax? We do not use our results to make either claim. The reductions we have seen here may reflect a weakness of the syntactic probes rather than a weakness of the models themselves, per se. In order to properly give the BERToids their due, one ought train the probes on data which controls for semantic cues (e.g. more Jabberwocky data) in addition to evaluating them on it. Here, we wish only to show that existing probes leverage semantic cues to make their syntactic predictions; since they do not properly *isolate* syntax, they should not be cited to support claims *about* syntax.

The high performance of the baselines (which inherently contain *no* syntax) is reason enough to be cautious about claims of these model's syntactic abilities. In general, single number metrics like these can be misleading: many correctly labeled easy dependencies may well obfuscate the mistakes being made on comparatively few hard ones, which may well be far more revealing (see, for instance, Briscoe and Carroll, 2006).

Even if these syntactic probes achieved impressive results on Jabberwocky data, beating the baselines by some margin, that alone would not be enough to conclude that the models encoded a deep understanding of syntax. Dependency grammarians generally parse sentences into directed graphs with labels; these probes by comparison only identify undirected unlabeled parse trees (compare Figures 1 and 2 for the difference). This much-simplified version of syntax has a vastly reduced space of possible syntactic structures. Consider a sentence with e.g. $n = 5$ words, for which there are only 125 possible unlabeled undirected parse trees (by Cayley's formula, n^{n-2}). As the high performance of the Majority baseline indicates, these are not uniformly distributed (some parse trees are more likely than others); a probe might well use these statistical confounds to advance its syntactic predictions. Although they remain present, biases like these are less easily exploitable in the labeled and directed case, where there are just over one billion possible parse trees to choose from.[6] Syntax is an incredibly rich phenomena—far more so than when it is reduced to syntactic distance.

[6] $n \cdot n^{n-2} \cdot k^{n-1}$ where k is the number of possible labels, and $k = 36$ in the case of UDs (Nivre et al., 2016).

6 O Frabjous Day! Callooh! Callay!

In this work, we trained two syntactic probes on a variety of BERToids, then evaluated them using Jabberwocky sentences, and showed that performance dropped substantially in this setting. This suggests that previous results from the probing literature may have overestimated BERT's syntactic abilities. However, in this context, we do not use the results to make any claims about BERT; we contend that to make such a claim one ought train the probes on Jabberwocky sentences, which would require more psuedowords than we had available. Instead, we advocate for the separation of syntax and semantics in probing. Future work could explore the development of artificial treebanks for use specifically for training syntactic probes, which minimise for any confounding statistical biases in the data. We make our Jabberwocky evaluation data and code publicly available at `https://github.com/rowanhm/jabberwocky-probing`.

References

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. In *The 5th International Conference on Learning Representations*.

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

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 41–48, Sydney, Australia. Association for Computational Linguistics.

Lewis Carroll. 1871. *Through the Looking-Glass, and What Alice Found There*. Macmillan.

Arthur Cayley. 1889. A theorem on trees. *Quarterly Journal of Mathematic's*, 23:376–378.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton and Co., The Hague.

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.

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.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287.

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, 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 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.

Dieuwke Hupkes and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5617–5621. International Joint Conferences on Artificial Intelligence Organization.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Jungo Kasai and Robert Frank. 2019. Jabberwocky parsing: Dependency parsing with lexical noise. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 113–123.

Gaurav Kharkwal. 2014. *Taming the Jabberwocky: Examining Sentence Processing with Novel Words*. Ph.D. thesis, Rutgers University-Graduate School-New Brunswick.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *The 3rd International Conference on Learning Representations*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

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

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

R. C. Prim. 1957. Shortest connection networks and some generalizations. *The Bell Systems Technical Journal*, 36(6):1389–1401.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Kathleen Rastle, Jonathan Harrington, and Max Coltheart. 2002. 358,534 nonwords: The ARC nonword database. *The Quarterly Journal of Experimental Psychology Section A*, 55(4):1339–1362.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Paul Soulos, R. Thomas McCoy, Tal Linzen, and Paul Smolensky. 2020. Discovering the compositional structure of vector representations with role learning networks. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 238–254, Online. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Hao Tian, Can Gao, Xinyan Xiao, Hao Liu, Bolei He, Hua Wu, Haifeng Wang, and Feng Wu. 2020. SKEP: Sentiment knowledge enhanced pre-training for sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4067–4076, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, Sheng-Fu Wang, Jason Phang, Anhad Mohananey, Phu Mon Htut, Paloma Jeretic, and Samuel R. Bowman. 2019. Investigating BERT's knowledge of language: Five analysis methods with NPIs. 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 2877–2887, Hong Kong, China. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

A The Jabberwocky

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!"

He took his vorpal sword in hand:
Long time the manxome foe he sought—
So rested he by the Tumtum tree,
And stood awhile in thought.

And as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!

One, two! One, two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.

"And hast thou slain the Jabberwock?
Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!"
He chortled in his joy.

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

Carroll, 1871

B Probe Training Details

For the Fast+Pos baseline, we use the base model of BERT, whose position embeddings are 768 dimensions, and the pretrained FastText embeddings trained on the Common Crawl (2M word variety with subword information).[7] Combining the position embeddings with the 300 dimensional FastText embeddings yields embeddings with 1068 dimensions for this baseline. By comparison, the 'large' version of the BERToids we train each consist of 24 layers, and produce embeddings which have 1024 dimensions.

[7]The FastText embeddings are avaiable at https://fasttext.cc/docs/en/english-vectors.html

Each BERToid we train uses a different tokenisation scheme. We need tokens which align with the tokens in the UD trees. In the case when one of the schemes does not split a word which is split in the UD trees, we merge nodes in the trees so they align. In the case where one of the schems splits a word which was not split in the UD trees, we use the first token. If the alignment is not easily fixed, we remove the sentence from the treebank. Table 1 shows the data split we are left with after sentences have been removed from the EWT UD treebank.

Dataset	**# Sentences**
Train	9444
Dev	1400
Test	1398

Table 1: Sentences following removals

To find optimimum hyperparameters, we perform a random search with 10 trials per model. When training, we used a batch size of 64 sentences, and as the optimiser we used Adam (Kingma and Ba, 2015). We consider three hyperparameters: the learning rate, the rank of the probe, and Dropout (Srivastava et al., 2014), over the ranges $[5 \times 10^{-5}], 5 \times 10^{-3}]$, $[1, d]$, and $[0.1, 0.8]$ respectively, where d is the dimensionality of the input representation. Along with the Fast+Pos baseline, we also perform the search on BERT, RoBERTa and GPT-2 at every fourth layer (so a total of 7 varieties each), and choose the best layer based on loss on the development set. For each trial, we train for a maximum of 20 epochs, and use early stopping if the loss does not decrease for 15 consecutive steps.