A Tale of a Probe and a Parser

Rowan Hall Maudslay[ð] **Josef Valvoda**[ð] **Tiago Pimentel**[ð]
Adina Williams[ə] **Ryan Cotterell**[ð,ſ]
[ð]University of Cambridge [ə]Facebook AI Research [ſ]ETH Zürich
`rh635@cam.ac.uk`, `jv406@cam.ac.uk`, `tp472@cam.ac.uk`,
`adinawilliams@fb.com`, `ryan.cotterell@inf.ethz.ch`

Abstract

Measuring what linguistic information is encoded in neural models of language has become popular in NLP. Researchers approach this enterprise by training "probes"—supervised models designed to extract linguistic structure from another model's output. One such probe is the *structural probe* (Hewitt and Manning, 2019), designed to quantify the extent to which syntactic information is encoded in contextualised word representations. The structural probe has a novel design, unattested in the parsing literature, the precise benefit of which is not immediately obvious. To explore whether syntactic probes would do better to make use of existing techniques, we compare the structural probe to a more traditional parser with an *identical* lightweight parameterisation. The parser outperforms structural probe on UUAS in seven of nine analysed languages, often by a substantial amount (e.g. by 11.1 points in English). Under a second less common metric, however, there is the opposite trend—the structural probe outperforms the parser. This begs the question: which metric should we prefer?

1 Introduction

Recently, unsupervised sentence encoders such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) have become popular within NLP. These pre-trained models boast impressive performance when used in many language-related tasks, but this gain has come at the cost of interpretability. A natural question to ask, then, is whether these models encode the traditional linguistic structures one might expect, such as part-of-speech tags or syntactic trees. To this end, researchers have invested in the design of diagnostic tools commonly referred to as **probes** (Alain and Bengio, 2017; Conneau et al., 2018; Hupkes et al., 2018; Poliak et al., 2018; Marvin and Linzen, 2018; Niven

and Kao, 2019). Probes are supervised models designed to extract a target linguistic structure from the output representation learned by another model.

Based on the authors' reading of the probing literature, there is little consensus on where to draw the line between probes and models for performing a target task (e.g. a part-of-speech tagger versus a probe for identifying part-of-speech). The main distinction appears to be one of researcher intent: probes are, in essence, a visualisation method (Hupkes et al., 2018). Their goal is not to best the state-of-the-art, but rather to indicate whether certain information is readily available in a model—probes should not "dig" for information, they should just expose what is already present. Indeed, a sufficiently expressive probe with enough training data could learn *any* task (Hewitt and Liang, 2019), but this tells us nothing about a representation, so it is beside the point. For this reason, probes are made "simple" (Liu et al., 2019), which usually means they are minimally parameterised.[1]

Syntactic probes, then, are designed to measure the extent to which a target model encodes syntax. A popular example is the **structural probe** (Hewitt and Manning, 2019), used to compare the syntax that is decodable from different contextualised word embeddings. Rather than adopting methodology from the parsing literature, this probe utilises a novel approach for syntax extraction. However, the precise motivation for this novel approach is not immediately clear, since it has nothing to do with model complexity, and appears orthogonal to the goal of a probe. Probes are designed to help researchers understand what information exists in a model, and unfamiliar ways of measuring this information may obscure whether we are actually gaining an insight about the representation we wish to measure, or the tool of measurement itself.

---

[1]An information-theoretic take on probe complexity is the subject of concurrent work; see Pimentel et al. (2020).
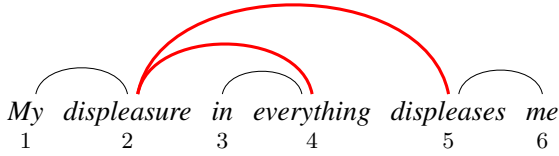
Figure 1: Example of an undirected dependency tree. We observe that the syntactic distance between *displeases* and *everything* is 2 (the red path).

Using the structural probe as a case study, we explore whether there is merit in designing models specifically for the purpose of probing—whether we should distinguish between the fundamental *design* of probes and models for performing an equivalent task, as opposed to just comparing their simplicity. We pit the structural probe against a simple parser that has the exact same lightweight parameterisation, but instead employs a standard loss function for parsing. Experimenting on multiligual BERT (Devlin et al., 2019), we find that in seven of nine typologically diverse languages studied (Arabic, Basque, Czech, English, Finnish, Japanese, Korean, Tamil, and Turkish), the parser boosts UUAS dramatically (by, for example, 11.1 points in English).

In addition to using UUAS, Hewitt and Manning (2019) also introduce a new metric—correlation of pairwise distance predictions with the gold standard. We find that the structural probe outperforms the more traditional parser substantially in terms of this new metric, but it is unclear why this metric matters more than UUAS. In our discussion, we contend that, unless a convincing argument to the contrary is provided, traditional metrics are preferable. Justifying metric choice is of central importance for probing, lest we muddy the waters with a preponderance of ill-understood metrics.

2   Syntactic Probing Using Distance

Here we introduce **syntactic distance**, which we will later train a probe to approximate.

Syntactic Distance   The syntactic distance between two words in a sentence is, informally, the number of steps between them in an undirected parse tree. Let $\mathbf{w} = w_1 \cdots w_n$ be a sentence of length n. A parse tree $\mathbf{t}$ belonging to the sentence $\mathbf{w}$ is an *undirected* spanning tree of n vertices, each representing a word in the sentence $\mathbf{w}$. The syntactic distance between two words w_i and w_j, denoted $\Delta_{\mathbf{t}}(w_i, w_j)$, is defined as the shortest path from

w_i to w_j in the tree $\mathbf{t}$ where each edge has weight 1. Note that $\Delta_{\mathbf{t}}(\cdot, \cdot)$ is a distance in the technical sense of the word: it is non-negative, symmetric, and satisfies the triangle inequality.

Tree Extraction   Converting from syntactic distance to a syntactic tree representation (or vice versa) is trivial and deterministic:

Proposition 1. *There is a bijection between syntactic distance and undirected spanning trees.*

Proof. Suppose we have the syntactic distances $\Delta_{\mathbf{t}}(w_i, w_j)$ for an unknown, undirected spanning tree $\mathbf{t}$. We may uniquely recover that tree by constructing a graph with an edge between w_i and w_j iff $\Delta_{\mathbf{t}}(w_i, w_j) = 1$. (This analysis also holds if we have access to only the ordering of the distances between all $|\mathbf{w}|^2$ pairs of words, rather than the perfect distance calculations—if that were the case, the minimum spanning tree could be computed e.g. with Prim's.) On the other hand, if we have an undirected spanning tree $\mathbf{t}$ and wish to recover the syntactic distances, we only need to compute the shortest path between each pair of words, with e.g. Floyd–Warshall, to yield $\Delta_{\mathbf{t}}(\cdot, \cdot)$ uniquely.   □

3   Probe, Meet Parser

In this section, we introduce a popular syntactic probe and a more traditional parser.

3.1   The Structural Probe

Hewitt and Manning (2019) introduce a novel method for approximating the syntactic distance $\Delta_{\mathbf{t}}(\cdot, \cdot)$ between any two words in a sentence. They christen their method the *structural probe*, since it is intended to uncover latent syntactic structure in contextual embeddings.[2] To do this, they define a parameterised distance function whose parameters are to be learned from data. For a word w_i, let $\mathbf{h}_i \in \mathbb{R}^d$ denote its contextual embedding, where d is the dimensionality of the embeddings from the model we wish to probe, such as BERT. Hewitt and Manning (2019) define the parameterised distance function

$$d_B(w_i, w_j) = \frac{}{\sqrt{(\mathbf{h}_i - \mathbf{h}_j)^\top B^\top B (\mathbf{h}_i - \mathbf{h}_j)}} \quad (1)$$

---

[2]In actual fact, the structural probe consists of two probes, one used to estimate the syntactic distance between words (which recovers an undirected graph) and another to calculate their depth in the tree (which is used to recover ordering). In this work, we focus exclusively on the former.

where $B \in \mathbb{R}^{r \times d}$ is to be learned from data, and r is a user-defined hyperparameter. The matrix $B^\top B$ is positive semi-definite and has rank at most r.[3]

The goal of the loss function, then, is to find B such that the distance function $d_B(\cdot, \cdot)$ best approximates $\Delta(\cdot, \cdot)$. If we are to organise our training data into pairs, each consisting of a gold tree $\mathbf{t}$ and the sentence it spans $\mathbf{w}$, we can then define the local loss function as

$$\ell(B, \langle \mathbf{t}, \mathbf{w} \rangle) = \hspace{3cm} (2)$$
$$\sum_{i=1}^{|\mathbf{w}|} \sum_{j=i+1}^{|\mathbf{w}|} \left| \Delta_{\mathbf{t}}(w_i, w_j) - d_B(w_i, w_j) \right|$$

which is then averaged over the entire training set $\mathcal{D} = \{ \langle \mathbf{t}^{(k)}, \mathbf{w}^{(k)} \rangle \}_{k=1}^N$ to create the following global loss function

$$\mathcal{L}(B) = \sum_{k=1}^{N} \frac{1}{|\mathbf{w}^{(k)}|^2} \ell \left(B, \langle \mathbf{t}^{(k)}, \mathbf{w}^{(k)} \rangle \right) \quad (3)$$

Dividing the contribution of each local loss by the square of the length of its sentence (the $|\mathbf{w}^{(k)}|^2$ factor in the denominator) ensures that each sentence makes an equal contribution to the overall loss, to avoid a bias towards the effect of longer sentences. This global loss can be minimised computationally using stochastic gradient descent.[4]

3.2 A Structured Perceptron Parser

Given that probe simplicity seemingly refers to parameterisation rather than the design of loss function, we infer that swapping the loss function should not be understood as increasing model complexity. With that in mind, here we describe an alternative to the structural probe which learns parameters for the same function d_B—a structured perceptron dependency parser, originally introduced in McDonald et al. (2005).

This parser's loss function works not by predicting every pairwise distance, but instead by predicting the tree based on the current estimation of the distances between each pair of words, then comparing the total weight of that tree to the total weight of the gold tree (based on the current distance predictions). The local perceptron loss is defined as

$$\ell(B, \langle \mathbf{t}, \mathbf{w} \rangle) = \sum_{(i,j) \in \mathbf{t}} d_B(w_i, w_j) \hspace{1cm} (4)$$
$$- \underbrace{\min_{\mathbf{t}' \in \mathcal{T}(\mathbf{w})} \sum_{(i',j') \in \mathbf{t}'} d_B(w_{i'}, w_{j'})}_{\text{computed with Prim's algorihtm}}$$

When the predicted minimum spanning tree $\mathbf{t}'$ perfectly matches the gold tree $\mathbf{t}$, each edge will cancel and this loss will equal zero. Otherwise, it will be positive, since the sum of the predicted distances for the edges in the gold tree will necessarily exceed the sum in the minimum spanning tree. Local loss is then trivially summed into a global loss as

$$\mathcal{L}(B) = \sum_{k=1}^{N} \ell \left(B, \langle \mathbf{t}^{(k)}, \mathbf{w}^{(k)} \rangle \right) \hspace{1cm} (5)$$

This quantity can also be minimised with a (sub)gradient-based method.

Though both the structural probe and the structured perceptron parser may seem equivalent under Prop. 1, there is a subtle but important difference. To minimise the loss in eq. (2), the structural probe needs to encode (in d_B) the rank-ordering of the distances between each pair of words within a sentence. This is not necessarily the case for the structured perceptron. It could minimise the loss in eq. (4) by just encoding each pair of words as "near" or "far"—and Prim's algorithm will do the rest.[5]

4 Experimental Setup

4.1 Processing Results

Embeddings and Data We experimented on the contextual embeddings in the final hidden layer of the pre-trained multilingual release of BERT (Devlin et al., 2019), and trained the models on the Universal Dependency (Nivre et al., 2016) treebands (v2.4). This allowed our analysis to be multilingual. More specifically, we consider eight typologically diverse languages (Arabic, Basque, Czech, Finnish, Japanese, Korean, Tamil, and Turkish), plus English.

---

[3] To see this, let $\mathbf{x} \in \mathbb{R}^d$ be a vector. Then, we have that $\mathbf{x}^\top B^\top B \mathbf{x} = (B\mathbf{x})^\top (B\mathbf{x}) = ||B\mathbf{x}||_2^2 \geq 0$.

[4] Hewitt and Manning found that replacing $d_B(\cdot, \cdot)$ in eq. (2) with $d_B(\cdot, \cdot)^2$ yielded better empirical results, so we do the same. For a discussion of this, refer to App. A.1 in Hewitt and Manning. Coenen et al. (2019) later offer a theoretical motivation, based on embedding trees in Euclidean space.

[5] One reviewer argued that, by injecting the tree constraint into the model in this manner, we lose the ability to answer the question of whether a probe discovered trees organically. While we believe this is valid, we do not see why the same criticism cannot be levelled against the structural probe—after all, it is trained on the same trees, just processed into pairwise distance matrices. The trees have been obfuscated, to be sure, but they remain in the data.
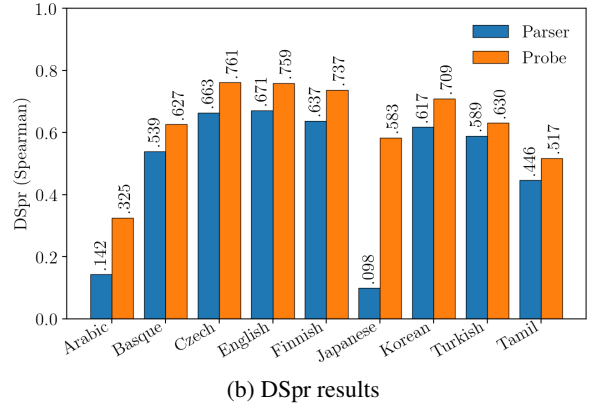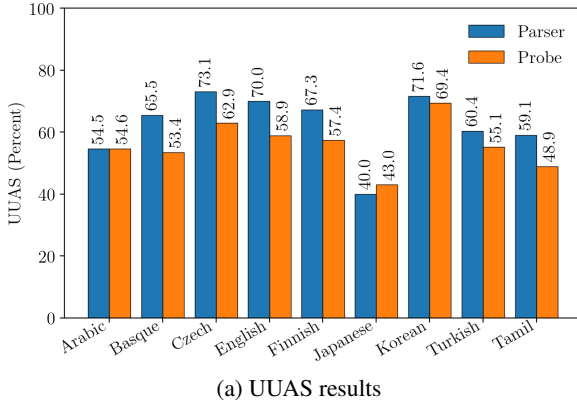
(a) UUAS results

(b) DSpr results

Figure 2: Results for the metrics in Hewitt and Manning (2019): different metrics, opposite trends.

Decoding the Predicted Trees Having trained a model to find a $d_B(\cdot, \cdot)$ that approximates $\Delta_{\mathbf{t}}(\cdot, \cdot)$, it is trivial to decode test sentences into trees (see Prop. 1). For an unseen sentence $\mathbf{w} = w_1 \cdots w_n$, we compute the $n \times n$ pairwise distance matrix D:

$$D_{uv} = \begin{cases} d_B(w_i, w_v) & \textbf{if } v > u \\ 0 & \textbf{otherwise} \end{cases} \quad (6)$$

We can then compute the predicted tree $\mathbf{t}$ from D using Prim's algorithm, which returns the minimum spanning tree from the predicted distances.

4.2 Experiments

To compare the performance of the models, we used both metrics from Hewitt and Manning (2019), plus a new variant of the second.

UUAS The *undirected unlabeled attachment score* (UUAS) is a standard metric in the parsing literature, which reports the percentage of correctly identified edges in the predicted tree.

DSpr The second metric is the Spearman rank-order correlation between the predicted distances, which are output from d_B, and the gold-standard distances (computable from the gold tree using the Floyd–Warshall algorithm). Hewitt and Manning term this metric *distance Spearman* (DSpr). While UUAS measures whether the model captures edges in the tree, DSpr considers pairwise distances between *all leaves* in the tree—even those which are not connected in a single step.

DSpr$_{P+FW}$ As a final experiment, we run DSpr again, but first pass each pairwise distance matrix D through Prim's (to recover the predicted tree) then through Floyd–Warshall (to recover a new distance matrix, with distances calculated *based on the predicted tree*). This post-processing would convert

a "near"–"far" matrix encoding to a precise rank-order one. This should positively affect the results, in particular for the parser, since that is trained to predict trees which result from the pairwise distance matrix, not the pairwise distance matrix itself.

5 Results

This section presents results for the structural probe and structured perceptron parser.

UUAS Results Figure 2a presents UUAS results for both models. The parser is the highest performing model on seven of the nine languages. In many of these the difference is substantial—in English, for instance, the parser outperforms the structural probe by 11.1 UUAS points.[6]

DSpr Results The DSpr results (Figure 2b) show the opposite trend: the structural probe outperforms the parser on all languages. The parser performs particularly poorly on Japanese and Arabic, which is surprising, given that these had the second and third largest sets of training data for BERT respectively (refer to Table 1 in the appendices). We speculate that this may be because in the treebanks used, Japanese and Arabic have a longer average sentence length than other languages.

DSpr$_{P+FW}$ Results Following the post-processing step, the difference in DSpr (shown in Figure 3) is far less stark than previously suggested—the mean difference between the two across all nine languages is just 0.0006 (in favour of the parser). Notice in particular the improvement

---

[6]We used the UD treebanks rather than the Penn-Treebank (Marcus et al., 1993), and experimented on the final hidden layer of multilingual BERT using a subset of 12,000 sentences from the larger treebanks. This renders our numbers incomparable to those found in Hewitt and Manning (2019).
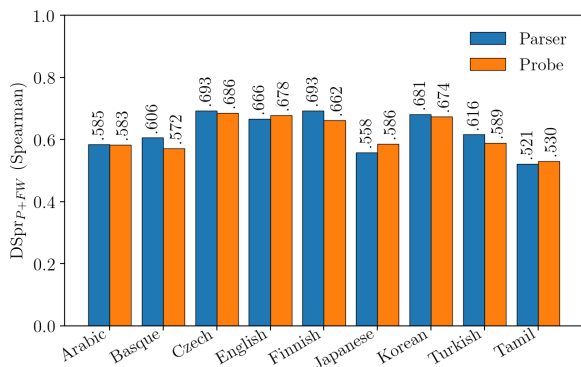
Figure 3: DSpr$_{P+FW}$ results—DSpr following the application of [Prim's](#) then [Floyd–Warshall](#) to D.

for both Arabic and Japanese—where previously (in the vanilla DSpr) the structured perceptron vastly underperformed, the post-processing step closes the gap almost entirely. Though Prop. 1 implies that we do not need to consider the full pairwise output of d_B to account for global properties of tree, this is not totally borne out in our empirical findings, since we do not see the same trend in DSpr$_{P+FW}$ as we do in UUAS. If we recover the gold tree, we will have a perfect correlation with the true syntactic distance—but we do not always recover the gold tree (the UUAS is less than 100%), and therefore the errors the parser makes are pronounced.

6   Discussion: Probe v. Parser

Although we agree that probes should be somehow more constrained in their complexity than models designed to perform well on tasks, we see no reason why being a "probe" should necessitate fundamentally different design choices. It seems clear from our results that how you design a probe has a notable effect on the conclusions one might draw about a representation. Our parser was trained to recover trees (so it is more attuned to UUAS), whilst the structural probe was trained to recover pairwise distances (so it is more attuned to DSpr)—viewed this way, our results are not surprising in the least.

The fundamental question for probe designers, then, is which metric best captures a linguistic structure believed to be a property of a given representation—in this case, syntactic dependency. We suggest that probing research should focus more explicitly on this question—on the development and justification of probing metrics. Once a metric is established and well motivated, a lightweight probe can be developed to determine whether that structure is present in a model.

If proposing a new metric, however, the burden of proof lies with the researcher to articulate and demonstrate why it is worthwhile. Moreover, this process of exploring which details a new metric is sensitive to (and comparing with existing metrics) ought not be conflated with an analysis of a particular model (e.g. BERT)—it should be clear whether the results enable us to draw conclusions about a model, or about a means of analysing one.

For syntactic probing, there is certainly no *a priori* reason why one should prefer DSpr to UUAS. If anything, we tentatively recommend UUAS, pending further investigation. The DSpr$_{P+FW}$ results show no clear difference between the models, whereas UUAS exhibits a clear trend in favour of the parser, suggesting that it may be easier to recover pairwise distances from a good estimate of the tree than vice versa. UUAS also has the advantage that it is well described in the literature (and, in turn, well understood by the research community).

According to UUAS, existing methods were able to identify more syntax in BERT than the structural probe. In this context, though, we use these results not to give kudos to BERT, but to argue that the perceptron-based parser is a better tool for syntactic probing. Excluding differences in parameterisation, the line between what constitutes a probe or a model designed for a particular task is awfully thin, and when it comes to syntactic probing, a powerful probe seems to look a lot like a traditional parser.

7   Conclusion

We advocate for the position that, beyond some notion of model complexity, there should be no inherent difference between the design of a probe and a model designed for a corresponding task. We analysed the structural probe ([Hewitt and Manning](#), 2019), and showed that a simple parser with an identical lightweight parameterisation was able to identify more syntax in BERT in seven of nine compared languages under UUAS. However, the structural probe outperformed the parser on a novel metric proposed in [Hewitt and Manning (2019)](#), bringing to attention a broader question: how should one choose metrics for probing? In our discussion, we argued that if one is to propose a new metric, they should clearly justify its usage.

References

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg. 2019. Visualizing and measuring the geometry of BERT. *arXiv preprint arXiv:1906.02715*.

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.

Robert W. Floyd. 1962. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345.

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.

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.

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

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. 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.

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.

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 2016)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

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*. Association for Computational Linguistics.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81, Brussels, Belgium. Association for Computational Linguistics.

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

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.

Stephen Warshall. 1962. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ÅĄukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

A Training Details

For all models (separately for each language), we considered three hyperparameters: r (full rank when $r = 768$, since this is the dimensionality of the BERT embeddings), the learning rate, and the dropout (Srivastava et al., 2014). To optimise these, we performed a random search, selecting values as judged by loss on the development set. When training, we used a batch size of 64 sentences, and employed early stopping after five steps based on loss reduction. As the optimiser, we used Adam (Kingma and Ba, 2015).

For each language, we used the largest available Universal Dependency 2.4 treebank. One-word sentences and sentences of over 50 words were discarded, and the larger treebanks were pruned to 12,000 sentences (in an 8:1:1 data split).

We used the BERT implementation made by Wolf et al. (2019). Since BERT accepts WordPiece units (Wu et al., 2016) rather than words, where necessary we averaged the output to get word-level embeddings. This is clearly a naïve composition method; improving it would likely strengthen the results for both the probe and the parser.

B Multilingual BERT Details

The multilingual BERT has 12 layers, 768 hidden states, and a total of 110M parameters. It was trained on the complete Wikipedia dumps for the 104 languages with the largest Wikipedias. Table 1 reports the size of the Wikipedias for the languages considered in this paper.[7] Further details of the training can be found on Google Research's GitHub.[8]

Language	Articles
Arabic	1,016,152
Basque	342,426
Czech	439,467
English	5,986,229
Finnish	473,729
Japanese	1,178,594
Korean	476,068
Tamil	125,031
Turkish	336,380

Table 1: The number of articles in the Wikipedias of the languages considered.

[7]According to https://en.wikipedia.org/wiki/List_of_Wikipedias, sampled 24/10/19.

[8]https://github.com/google-research/bert/blob/master/multilingual.md