Low-Complexity Probing via Finding Subnetworks

Steven Cao[1,2] **Victor Sanh**[2] **Alexander M. Rush**[2]

[1]University of California, Berkeley
[2]Hugging Face
stevencao@berkeley.edu
{victor,sasha}@huggingface.co

Abstract

The dominant approach in probing neural networks for linguistic properties is to train a new shallow multi-layer perceptron (MLP) on top of the model's internal representations. This approach can detect properties encoded in the model, but at the cost of adding new parameters that may learn the task directly. We instead propose a subtractive pruning-based probe, where we find an existing subnetwork that performs the linguistic task of interest. Compared to an MLP, the subnetwork probe achieves both higher accuracy on pre-trained models and lower accuracy on random models, so it is both better at finding properties of interest and worse at learning on its own. Next, by varying the complexity of each probe, we show that subnetwork probing Pareto-dominates MLP probing in that it achieves higher accuracy given any budget of probe complexity. Finally, we analyze the resulting subnetworks across various tasks to locate where each task is encoded, and we find that lower-level tasks are captured in lower layers, reproducing similar findings in past work.

1 Introduction

While pre-training has produced large gains for natural language tasks, it is unclear what a model learns during pre-training. Research in *probing* investigates this question by training a shallow classifier on top of the pre-trained model's internal representations to predict some linguistic property (Adi et al., 2016; Shi et al., 2016; Tenney et al., 2019, *inter alia*). The resulting accuracy is then roughly indicative of the model encoding that property.

However, it is unclear how much is learned by the probe versus already captured in the model representations. This question has been the subject of much recent debate (Hewitt and Liang, 2019; Voita and Titov, 2020; Pimentel et al., 2020b, *inter alia*). We would like the probe to find only and all properties captured by a model, leading to a tradeoff between accuracy and complexity: a linear probe is insufficient to find the non-linear patterns in neural models, but a deeper multi-layer perceptron (MLP) is complex enough to learn the task on its own.

Motivated by this tradeoff and the goal of low-complexity probes, we consider a different approach based on pruning. Specifically, we search for a subnetwork — a version of the model with a subset of the weights set to zero — that performs the task of interest. As our search procedure, we build upon past work in pruning and perform gradient descent on a continuous relaxation of the search problem (Louizos et al., 2017; Mallya et al., 2018; Sanh et al., 2020). The resulting probe has many fewer free parameters than MLP probes.

Our experiments evaluate the accuracy-complexity tradeoff compared to MLP probes on an array of linguistic tasks. First, we find that the neuron subnetwork probe has both higher accuracy on pre-trained models and lower accuracy on random models, so it is both better at finding properties of interest and less able to learn the tasks on its own. Next, we measure complexity as the bits needed to transmit the probe parameters (Pimentel et al., 2020a; Voita and Titov, 2020). Varying the complexity of each probe, we find that subnetwork probing Pareto-dominates MLP probing in that it achieves higher accuracy given any desired complexity. Finally, we analyze the resulting subnetworks across various tasks and find that lower-level tasks are captured in lower layers, reproducing similar findings in past work (Tenney et al., 2019). These results suggest that subnetwork probing is an effective new direction for improving our understanding of pre-training.

2 Related Work

Probing. Probing investigates whether a model captures some hypothesized property and typically

The code is available at https://github.com/stevenxcao/subnetwork-probing.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 960–966
June 6–11, 2021.

involves learning a shallow classifier on top of the model's frozen internal representations (Adi et al., 2016; Shi et al., 2016; Conneau et al., 2018). Recent work has primarily applied this technique to pre-trained models.[1] Clark et al. (2019), Hewitt and Manning (2019), and Manning et al. (2020) found that BERT captures various properties of syntax. Tenney et al. (2019) probed the layers of BERT for an array of tasks, and they found that their localization mirrored the classical NLP pipeline (part-of-speech, parsing, named entity recognition, semantic roles, coreference) in that lower-level tasks were captured in the lower layers.

However, these results are difficult to interpret due to the use of a learned classifier. One line of work suggests comparing the probe accuracy to random baselines, e.g. random models (Zhang and Bowman, 2018) or random control tasks (Hewitt and Liang, 2019). Other works take an information-theoretic view: Voita and Titov (2020) measure the complexity of the probe in terms of the bits needed to transmit its parameters, while Pimentel et al. (2020b) argue that probing should measure mutual information between the representation and the property. Pimentel et al. (2020a) propose a Pareto approach where they plot accuracy versus probe complexity, unifying several of these goals. We use these proposed metrics to compare our probing method to standard probing approaches.

Subnetworks. While pruning is widely used for model compression, some works have explored pruning as a technique for learning as well. Mallya et al. (2018) found that a model trained on ImageNet could be used for new tasks by learning a binary mask over the weights. More recently, Radiya-Dixit and Wang (2020) and Zhao et al. (2020) showed the analogous result in NLP that weight pruning can be used as an alternative to fine-tuning for pre-trained models. Our paper seeks to use pruning to reveal what the model already captures, rather than learn new tasks.

3 Subnetwork Probing

Given a task and a pre-trained encoder model with a classification head, our goal is to find a subnetwork with high accuracy on that task, where a subnetwork is the model with a subset of the encoder weights masked, i.e. set to zero. We search for this subnetwork via supervised gradient descent on the head and a continuous relaxation of the mask. We also mask at several levels of granularity, including pruning weights, neurons, or layers.

To learn the masks, we follow Louizos et al. (2017). Letting $\phi \in \mathbb{R}^d$ denote the model weights, we associate the ith weight ϕ_i with a real-valued parameter θ_i, which parameterizes a random variable $Z_i \in [0, 1]$ representing the mask. Z_i follows the hard concrete distribution HardConcrete(β, θ_i) with temperature β and location θ_i,

$$
\begin{aligned}
U_i &\sim \text{Unif}[0, 1] \\
S_i &= \sigma\left(\frac{1}{\beta}\left(\log\frac{U_i}{1-U_i} + \theta_i\right)\right) \\
Z_i &= \min\left(1, \max\left(0, S_i(\zeta - \gamma) + \gamma\right)\right),
\end{aligned}
$$

where σ denotes the sigmoid and $\gamma = -0.1$, $\zeta = 1.1$ are constants. This random variable can be thought of as a soft version of the Bernoulli. S_i follows the concrete (or Gumbel-Softmax) distribution with temperature β (Maddison et al., 2016; Jang et al., 2016). To put non-zero mass on 0 and 1, the distribution is stretched to the interval $(\gamma = -0.1, \zeta = 1.1)$ and clamped back to $[0, 1]$.

We will denote the mask as $Z_i = z(U_i, \theta_i)$ and the masked weights as $\phi * Z$. We can then optimize the mask parameters θ via gradient descent. Specifically, let $f(x; \phi)$ denote the model. Then, given a data point (x, y) and a loss function L, we can minimize the expectation of the loss, or

$$\mathcal{L}(x, y, \theta) = \mathbb{E}_{U_i \sim \text{Unif}[0,1]} L(f(x; \phi * z(U, \theta)), y).$$

We estimate the expectation via sampling: we sample a single U and take the gradient $\nabla_\theta L(f(x; \phi * z(U, \theta))$. To encourage sparsity, we penalize the mask based on the probability it is non-zero, or

$$R(\theta) = \mathbb{E}\|\theta\|_0 = \frac{1}{d}\sum_{i=1}^{d} \sigma\left(\theta_i - \beta \log\frac{-\gamma}{\zeta}\right).$$

Letting λ denote regularization strength, our objective becomes $\frac{1}{|D|}\sum_{(x,y)\in D} \mathcal{L}(x, y, \theta) + \lambda R(\theta)$.[2]

4 Probe Evaluation

To evaluate the accuracy-complexity tradeoff of a probe, we adapt methodology from recent work.

[1] While probing is also used in other domains (e.g. neural decoding), we focus on understanding neural models. Therefore, one source of strength for our probe is that we exploit the entire model, rather than only operating on representations.

[2] Departing from past work, we schedule λ linearly to improve search: it stays fixed at 0 for the first 25% of training, linearly increases to λ_{max} for the next 50%, and then stays fixed. We set $\lambda_{\text{max}} = 1$ in our evaluation experiments.

First, we consider the non-parametric test of probing a random model (Zhang and Bowman, 2018). We check probe accuracy on the pre-trained model, the model with the encoder randomly reset (*reset encoder*), and the model with the encoder and embeddings reset (*reset all*). An ideal probe should achieve high accuracy on the pre-trained model and low accuracy on the reset models.[3]

Next, we consider a parametric test based on probe complexity. We first vary the complexity of each probe, where for subnetwork probing we associate multiple encoder weights with a single mask,[4] and for the MLP probe we restrict the rank of the hidden layer. We then plot the resulting accuracy-complexity curve (Pimentel et al., 2020a).

To plot this curve, we need a measure of complexity that can compare probes of different types. Therefore, we measure complexity as the number of bits needed to transmit the probe parameters (Voita and Titov, 2020), where for simplicity we use a uniform encoding. In the subnetwork case, this encoding corresponds to using a single bit for each mask parameter. In the case of an MLP probe, each parameter is a real number, so the number of bits per parameter depends on its range and precision. For example, if each parameter lies in $[a, b]$ and requires ϵ precision, then we need $\log(\frac{b-a}{\epsilon})$ bits per parameter. To avoid having the choice of precision impact results, we plot lower and upper bounds of 1 and 32 bits per parameter.

5 Experimental Setup

We probe `bert-base-uncased` (Devlin et al., 2019; Wolf et al., 2020) for the following tasks:

(1) **Part-of-speech Tagging:** We use the part-of-speech tags in the universal dependencies dataset (Zeman et al., 2017). As our classification head, we use dropout with probability $p = 0.1$, followed by a linear layer and softmax projecting from the BERT dimension to the number of tags.

[3]The reset encoder model contains some non-contextual information from its word embeddings, but no modeling of context; therefore, we would expect it to have better probe accuracy on tasks based mainly on word type (e.g. part-of-speech tagging).

[4]For subnetworks, the pre-trained model has 72 matrices of size 768×768; see https://github.com/huggingface/transformers/blob/v3.4.0/src/transformers/modeling_bert.py. For each matrix, let n_r and n_c denote the number of rows and columns per mask. Then, we set (n_r, n_c) to $(768, 768)$, $(768, 192)$, $(768, 24)$, $(768, 6)$, $(768, 1)$, $(192, 1)$, $(24, 1)$, $(6, 1)$, and $(1, 1)$. $(768, 768)$ corresponds to masking entire matrices, $(768, 1)$ to masking neurons, and $(1, 1)$ to masking weights.

	Pre-trained ↑	Reset encoder ↓	Reset all ↓
Part-of-speech Tagging			
Subnetworks	**93.39**	87.53	71.53
MLP-1	90.25	86.53	**69.16**
Fine-tuning	95.69	86.47	84.42
Dependency Parsing			
Subnetworks	**86.86**	54.31	**39.84**
MLP-1	76.65	54.09	42.81
Fine-tuning	89.93	79.10	74.48
Named Entity Recognition			
Subnetworks	**87.94**	**68.09**	**30.83**
MLP-1	84.80	69.35	53.25
Fine-tuning	93.68	81.80	70.08

Table 1: Probe accuracy for `bert-base-uncased` (Pre-trained), the model with the encoder reset but the embeddings preserved (Reset encoder), and the model completely reset (Reset all). The ↑ and ↓ denote whether higher or lower is better (substantially better numbers are bolded). For reference, we also include fine-tuning (training all model parameters rather than probing). Compared to MLP-1, neuron subnetwork probing achieves higher accuracy for the pre-trained model and lower accuracy for the random models.

(2) **Dependency Parsing**: We use the universal dependencies dataset (Zeman et al., 2017) and the biaffine head for classification (Dozat and Manning, 2016). We report macro-averaged labeled attachment score.

(3) **Named Entity Recognition (NER)**: We use the data from the CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003) and the same classification head as for part-of-speech tagging. We report F1 using the CoNLL 2003 script.

Our primary probing baseline is the MLP probe with one hidden layer (MLP-1):

$$\texttt{MLP-1}(x) = \texttt{ReLU}(\texttt{LayerNorm}(UV^T x)),$$

with $U, V \in \mathbb{R}^{d \times r}$. The choice of r restricts the rank of the hidden layer and thus its complexity.[5] Then, if $g(x; \phi)$ is our pre-trained encoder and `cls` is the classification head, our two probes are $f_{\text{Subnetwork}}(x) = \texttt{cls}(g(x; \phi * Z))$ and $f_{\text{MLP-1}}(x) = \texttt{cls}(\texttt{MLP-1}(g(x; \phi)))$.

While we vary the complexity of each probe to produce the accuracy-complexity plot, we default to neuron subnetwork probing and full rank MLP-1 probing in all other experiments.

[5]We set the rank to 1, 2, 5, 10, 25, 50, 125, 250, and 768.

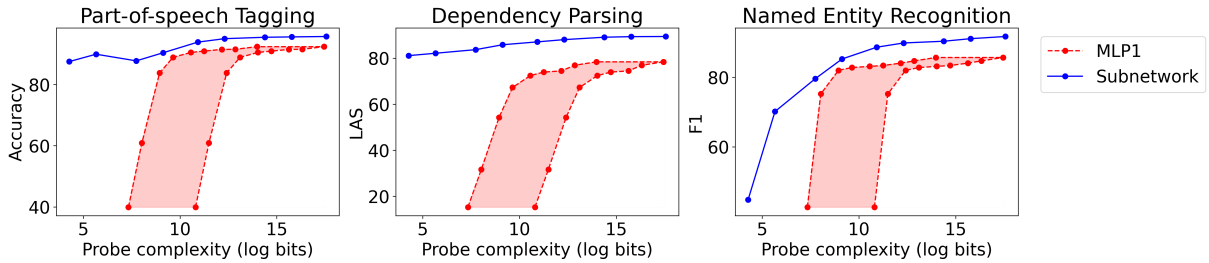

Figure 1: Subnetwork probe and MLP-1 probe accuracy on the pre-trained model plotted versus probe complexity, measured in $\ln(\text{bits})$. For the MLP-1 probe, we plot lower and upper bounds on complexity of 1 and 32 bits per parameter. The subnetwork probe Pareto-dominates the MLP-1 probe in that it achieves higher accuracy for any desired complexity, even if we assume the optimistic lower bound on MLP-1 complexity of 1 bit per parameter.

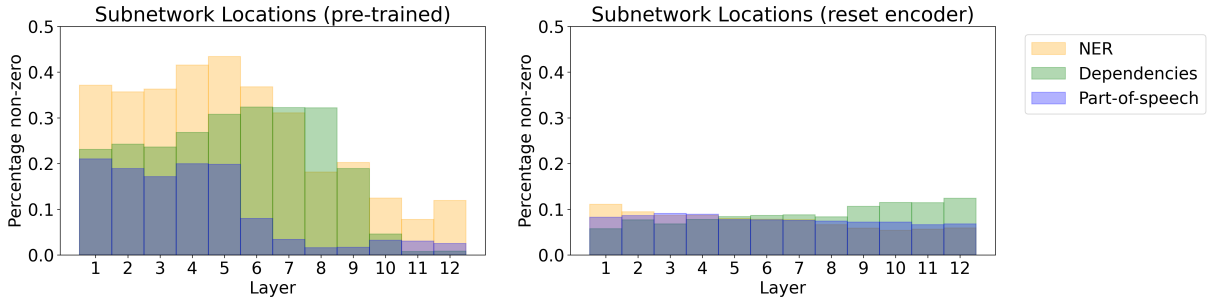

Figure 2: The percentage of non-zero weights in each layer for subnetworks of the pre-trained model and the reset encoder model. While the reset encoder model's subnetworks are distributed uniformly across the layers, the pre-trained model's subnetworks are localized, with the order part-of-speech → dependencies → NER.

6 Results

Accuracy-Complexity Tradeoff. Table 1 shows the results from the non-parametric experiments. When probing the pre-trained model, the subnetwork probe has much higher accuracy than the MLP-1 probe across all tasks. Furthermore, when probing the random models, the subnetwork probe has much lower accuracy for dependency parsing and NER, suggesting that the probe is less able to learn the task on its own. Overall, these numbers suggest that the subnetwork probe is a more faithful probe in that it finds properties when they are present, and does not find them in a random model.

Figure 1 plots the results from the parametric experiments, where we vary the complexity of each probe, apply it to the pre-trained model, and plot the resulting accuracy-complexity curve. We find that the subnetwork probe Pareto-dominates the MLP-1 probe in that it achieves higher accuracy for any complexity, even if we assume an overly optimistic MLP-1 lower bound of 1 bit per parameter. In particular, for part-of-speech and dependency parsing, the subnetwork probe achieves high accuracy even when given only 72 bits, while the MLP-1 probe falls off heavily at ∼20K bits.

Subnetwork Analysis. An auxiliary benefit of subnetwork probing is that we can examine the subnetworks produced by the procedure. One possibility is to look at the locations of the subnetworks, and one way to examine location is to count the number of unmasked weights in each layer. Figure 2 shows locations of the remaining parameters in the subnetworks extracted from the pre-trained model and the random encoder model. To prune as many parameters as possible, we set λ_{max} to be the largest out of $(1, 5, 25, 125)$ such that accuracy is within 10% of fine-tuning accuracy (see the Appendix for more details). We then examine the sparsity levels of the attention heads for each layer. While reset encoder model's subnetworks are uniformly distributed across the layers, the pre-trained model's subnetworks are localized and follow the order part-of-speech → dependencies → NER, reproducing the order found in Tenney et al. (2019). While Tenney et al. (2019) derived layer importance by training classifiers at each layer, we find location directly via pruning. This experiment strengthens their result and represents one example where subnetwork probing reveals additional insights into the model beyond accuracy.

7 Conclusion

Together, these results show that subnetwork probing is more faithful to the model and offers richer analysis than existing probing approaches. While this work explores accuracy and location-based analysis, there are other possible directions, e.g., applying neuron explainability techniques. Therefore, we see subnetwork probing as a fruitful new direction for understanding pre-training.

8 Ethical Considerations

While pre-trained models have improved performance for many NLP tasks, they exhibit biases present in the pre-training corpora (Manzini et al., 2019; Tan and Celis, 2019; Kurita et al., 2019, *inter alia*). As a result, deploying pre-trained models runs the risk of reinforcing social biases. Probing gives us a tool to better understand and hopefully mitigate these biases. As one example of such a study, Vig et al. (2020) analyze how neurons and attention heads contribute to gender bias in pre-trained transformers. Therefore, while we analyze linguistic tasks in our paper, our method could also provide insights into model bias, e.g. by analyzing subnetworks for bias detection tasks like CrowS-Pairs (Nangia et al., 2020) or StereoSet (Nadeem et al., 2020).

9 Acknowledgements

We would like to thank Eric Wallace, Kevin Yang, Ruiqi Zhong, Dan Klein, and Yacine Jernite for their useful comments and feedback. This work was done during an internship at Hugging Face.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing.

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.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax.

Keita Kurita, Nidhi Vyas, Ayush Pareek, Alan W Black, and Yulia Tsvetkov. 2019. Measuring bias in contextualized word representations. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 166–172, Florence, Italy. Association for Computational Linguistics.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2017. Learning sparse neural networks through l_0 regularization.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights.

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 of the United States of America*.

Thomas Manzini, Lim Yao Chong, Alan W Black, and Yulia Tsvetkov. 2019. Black is to criminal as caucasian is to police: Detecting and removing multiclass bias in word embeddings. 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 615–621, Minneapolis, Minnesota. Association for Computational Linguistics.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2020. Stereoset: Measuring stereotypical bias in pretrained language models.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

Tiago Pimentel, Naomi Saphra, Adina Williams, and Ryan Cotterell. 2020a. Pareto probing: Trading off accuracy for complexity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3138–3153, Online. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020b. 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.

Evani Radiya-Dixit and Xin Wang. 2020. How fine can fine-tuning be? learning efficient language models. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2435–2443, Online. PMLR.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Yi Chern Tan and L. Elisa Celis. 2019. Assessing social and intersectional biases in contextualized word representations. In *Advances in Neural Information Processing Systems*, volume 32, pages 13230–13241. Curran Associates, Inc.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. 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.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *NeuRIPS*.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, Remi Louf, Patrick von Platen, Tim Rault, Yacine Jernite, Teven Le Scao, Sylvain Gugger, Julien Plu, Clara Ma, Canwei Shen, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

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.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online. Association for Computational Linguistics.

A Appendix

A.1 Hyperparameters

The mask parameters are optimized using Adam with $\beta = (0.9, 0.999)$, $\epsilon = 1 \times 10^{-8}$, and learning rate 0.2 with linear warmup for the first 10% of the data. The classification head parameters are also optimized using Adam with the same hyperparameters and warmup, except with learning rate 5×10^{-5}. The MLP-1 and fine-tuning baselines are also optimized using Adam with the same hyperparameters, warmup, and learning rate 5×10^{-5}. We train for 30 epochs for all tasks.

A.2 Varying Regularization Strength

Table 2 shows probing accuracies for $\lambda_{\text{max}} \in (1, 5, 25, 125)$. Our method is consistently more selective than MLP-1 across the various values of λ_{max}, except for $\lambda_{\text{max}} = 125$, which seems to require too much sparsity.

A.3 Reproducibility Checklist

Experiments were run in Google Colab using a single 12GB NVIDIA Tesla K80 GPU. For each task, one run of fine-tuning took about half an hour. We used the transformers implementation of the `bert-base-uncased` model (Wolf et al., 2020; Devlin et al., 2019), which has 12 layers, 768 hidden dimension, 12 heads, and 110M parameters. As data, we used the dev (2002 examples) and train (12541 examples) splits of the English universal dependencies dataset (Zeman et al., 2017), and the test (3235 examples) and train (13862 examples) splits of the CoNLL 2003 NER shared task (Tjong Kim Sang and De Meulder, 2003).

	Reset all	Reset encoder	Pre-trained
Part-of-speech Tagging			
Subnetworks			
$\lambda_{\text{max}} = 1$	71.53	87.53	93.39
$\lambda_{\text{max}} = 5$	70.45	87.20	92.41
$\lambda_{\text{max}} = 25$	68.65	86.23	90.66
$\lambda_{\text{max}} = 125$	66.39	86.10	84.86
MLP-1	69.16	86.53	90.25
Fine-tuning	84.42	86.47	95.69
Dependency Parsing			
Subnetworks			
$\lambda_{\text{max}} = 1$	39.84	54.31	86.86
$\lambda_{\text{max}} = 5$	43.36	54.93	85.99
$\lambda_{\text{max}} = 25$	41.43	54.41	83.12
$\lambda_{\text{max}} = 125$	43.07	53.70	74.49
MLP-1	42.81	54.09	76.65
Fine-tuning	74.48	79.10	89.93
Named Entity Recognition (NER)			
Subnetworks			
$\lambda_{\text{max}} = 1$	30.83	68.09	87.94
$\lambda_{\text{max}} = 5$	22.92	65.18	84.48
$\lambda_{\text{max}} = 25$	3.41	57.82	72.26
$\lambda_{\text{max}} = 125$	2.04	57.56	50.67
MLP-1	53.25	69.35	84.80
Fine-tuning	70.08	81.80	93.68

Table 2: Subnetwork probing accuracies while varying regularization strength.