A Edge Probes on the Top Layer of BERT

In Section 4, we present the edge probing results based on the *mix* probes. We take as input a learned scalar mixing of the representations from various layers and use these to evaluate whether linguistic information is available *somewhere* within the core model. Here, we repeat the edge probing results but use the *top* probe, which measures the amount of linguistic information provided in the output (top) layer. This provides an alternate perspective from the mix probe and measures the linguistic information that is available immediately before the output.

The results of this test are presented in Figure 2. The top probes provide an interesting perspective on fine-tuning, showing that while still present in earlier layers, some linguistic features *are* discarded at the top of the model. For the model fine-tuned on Dep, we see that the syntactic information such as Constituents and Dependencies is well-preserved. In contrast, semantic-based information such as entities or relations appears to be degraded by the last layer. With respect to the models fine-tuned on MNLI and SQuAD, these models generally show the opposite pattern, discarding syntactic information but retaining semantic information.

We note that it is difficult to draw any *causal* conclusions from this result: the absence of a feature at the top of the model does not necessarily mean it was irrelevant to the model's predictions. It is possible that the model reasons over such features from earlier layers, synthesizing the result into more concise, task-specific information at the top-most layers.

B Minimum Description Length Probes

One concern with supervised probing–such as edge probing or structural probes–is that the capacity of the probing model may lead to strong performance even from weak or spurious features, such as random encoders. We address this partially in Section 4 by comparing to lexical and randomized baselines, but recent techniques provide a more robust way of countering this effect. In this section, we turn to the Minimum Description Length (MDL) framework of Voita and Titov (2020) in order to analyze the 'amount of effort' required to learn any given probe. Readily available linguistic structures within the representations could be learned from just a few examples, whereas many more examples would be required to memorize random labels. Thus, by measuring model performance as a function of number of examples, we attain a measure for the difficulty of extracting a particular linguistic structure from a set of representations.

This coincides with the Online variant of the MDL probes. See (Voita and Titov, 2020; Rissanen, 1984; Yogatama et al., 2019) for an information-theoretic motivation, which is related to the information required to transmit the data in a two-player game where both agents have already agreed upon a model, random seed, and the learning algorithm. The associated metric, the Online Codelength, effectively approximates the area under the learning curve as a function of the number of examples and is computed as follows.

For a dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, we consider the effect of training on increasingly larger sets of data of size $0 < t_1 < t_2 < \cdots < t_S = n$[11]. Let θ_{t_i} represent the model's parameters after training with t_i examples. We then use $x_{t_i+1:t_{i+1}}, y_{t_i+1:t_{i+1}}$ to evaluate θ_{t_i} and measure performance in terms of compression. For example, when trained with no datapoints, the best possible compression for a probe with K labels is $\log_2 K$, corresponding to the uniform distribution. In total, the Online Minimum Description Length is therefore given by:

$$L^{online} = t_1 \log_2 K + \sum_{i=1}^{S-1} \log_2 p_{\theta_{t_i}}(y_{t_s+1:t_{s+1}} \mid x_{t_s+1:t_{s+1}})$$

We repeat this Minimum Description Length probe using the *mix* version of the edge probe, as done in the core body of the paper. The results from this test are presented in Figure 6. Since codelength varies significantly based on task, we present the ratio of the value for the BERT base model compared to the fine-tuned model, such that larger values correspond to greater probe performance with fewer examples. Focusing first on dependency parsing, we see that significant improvement in the corresponding Dependencies edge probing task and limited degradation in performance on the higher order semantic tasks such

[11] We follow Voita and Titov (2020) in our choice on t_i and other relevant hyper-parameters.

		Δ for Baselines		Δ for Fine-tuned Models		
Task	**BERT Base**	**Lexical**	**Randomized**	**MNLI**	**SQuAD**	**Dep**
POS	96.8	-8.3	-12.9	-2.8	-0.8	-2.5
Constituents	84.3	-12.8	-24.0	-5.8	-3.6	3.7
Dependencies	93.0	-13.1	-15.7	-2.5	-3.9	2.4
Entities	95.9	-6.3	-9.7	-0.7	-1.5	-2.5
SRL	90.5	-11.4	-12.6	-1.4	-2.4	-1.4
Coreference	95.2	-5.3	-5.7	-0.9	-1.0	-1.7
SPR	84.2	-6.2	-11.8	-1.0	-0.4	-2.1
Relations	78.8	-20.0	-39.8	-2.0	-0.8	-4.7

Table 2: Comparison of F1 performance on the edge probing tasks, using the *top* version of the edge probe. The BERT Base performance is consistent with (Tenney et al., 2019b), whereas there appear to be larger drops for fine-tuned models.

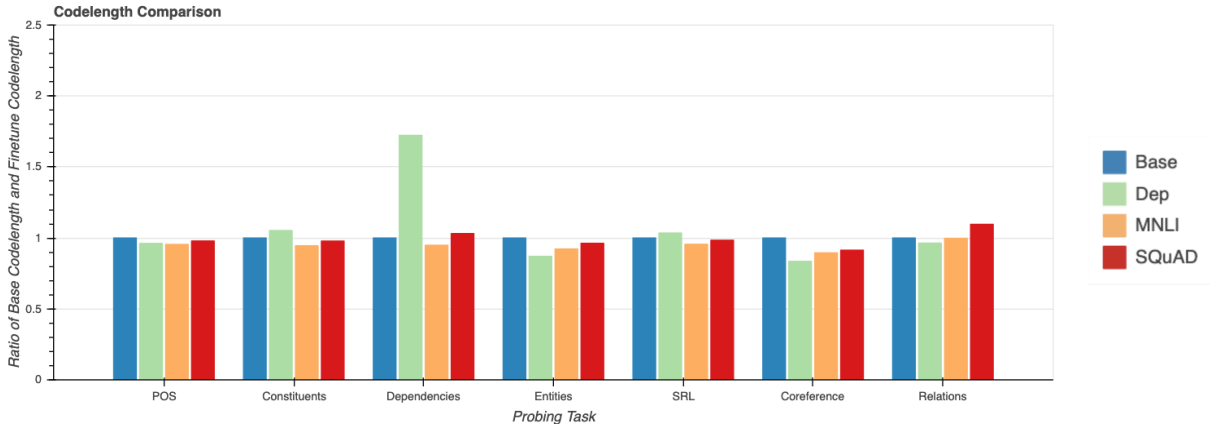

Figure 6: Comparison of the Online Codelength from the BERT Base models and the fine-tuned equivalents, using the Minimum Description Length probes. Since the codelength depends on the number of examples in addition to model efficiency, we show the relative change in codelength compared to probing the Base model (leftmost bar in each group). Using this metric, higher values corresponds to a structure that is easier to learn.

as coreference. This is interesting as it shows that fine-tuning is surfacing the linguistic structure when the downstream task is directly related. That all other tasks remain approximately constant reflects the finding from the other probing experiments in Section 4, that linguistic information is available just not being altered significantly. This hold true for both the MNLI and SQuAD models where there is limited change in the codelengths.

C Additional Experimental Details

For the sake of reproducible results, in this section, we provide additional experimental details and setup.

Computing Infrastructure All BERT models are pre-trained and fine-tuned on Cloud TPUs, using 32 and 8 workers at a time respectively. For these models, we build off of the public version of the BERT (Devlin et al., 2019) code, available at `https://github.com/google-research/bert`. In contrast, all of the probing experiments (edge probing, structural-probing, and RSA) were conducted using GPUs.

Average Runtimes To get a sense of the overall compute used in our experiments, pre-training takes on the order of 4-5 days for the BERT Base models. Both the fine-tuning steps and the probing models (edge and structural) take 1-6 hours on GPUs, depending on the task and number of encoder layers used. Representational Similarity Analysis takes only a few minutes since it requires only a single forward pass through the model for each examples, and no additional training.

Datasets Due to the extensive number of datasets used in this paper, we refer to prior work for full details. For example, BERT pre-training is conducted on the BooksCorpus (Zhu et al., 2015) (800M words) and English Wikipedia

(2500M words). Our fine-tuned models make use of popular benchmark datasets such as MNLI (Williams et al., 2018) (433k sentence pairs),[12] SQuAD (Rajpurkar et al., 2016) (100k+ question-answer pairs),[13] and the CoNLL 2017 Shared Task (Zeman et al., 2017)[14]. We use the standard splits for each of the provided tasks in order to create our train/test splits. For edge probing, we replicate the experiments and datasets by Tenney et al. (2019b,a). Dataset statistics for each of the tasks is provided at https://github.com/nyu-mll/jiant/tree/master/probing. We modify the dependency labeling task from the English Web Treebank (Silveira et al., 2014) to extract tree depths and distances for use in the structural probing task.

Implementation We use a TensorFlow (Abadi et al., 2015) re-implementation of original Jiant (Pruksachatkun et al., 2020) edge probing codebase. It is generally difficult to exactly reproduce a PyTorch result in TensorFlow and vice versa, due to differences in optimizers, training libraries, and randomization. However, our numbers generally agree with (Tenney et al., 2019b,a) up to the variance observed across training runs. The structural probes were re-implemented within TensorFlow as well and give performance comparable to (Hewitt and Manning, 2019) on the Base models.

[12] Available at https://cims.nyu.edu/~sbowman/multinli/.

[13] Available at https://rajpurkar.github.io/SQuAD-explorer/.

[14] Available at http://universaldependencies.org/conll17/.