Does My Representation Capture X? Probe-Ably

Deborah Ferreira[*]🐝, **Julia Rozanova**[*]🐝, **Mokanarangan Thayaparan**[*]🐝,
Marco Valentino[*]🐝, **André Freitas**🐝🏔️

Department of Computer Science, University of Manchester, United Kingdom🐝

Idiap Research Institute, Switzerland 🏔️

{firstname.surname} @manchester.ac.uk

Abstract

Probing (or *diagnostic classification*) has become a popular strategy for investigating whether a given set of intermediate features is present in the representations of neural models. Probing studies may have misleading results, but various recent works have suggested more reliable methodologies that compensate for the possible pitfalls of probing. However, these best practices are numerous and fast-evolving. To simplify the process of running a set of probing experiments in line with suggested methodologies, we introduce **Probe-Ably:** an extendable probing framework which supports and automates the application of probing methods to the user's inputs.

1 Introduction

Recent interest in investigating the intermediate features present in neural models' representations has led to the use of structural analysis methods such as *probing*.

At its simplest, **probing**[1] is the training of an external classifier model (a "probe") to determine the extent to which a set of auxiliary target feature labels can be predicted from the internal model representations. For example, probing studies have been carried out to determine whether word and sentence representations generated by models such as BERT (Devlin et al., 2019) capture intermediate syntactic and semantic features such as parts of speech and dependency labels (Hewitt and Manning, 2019b; Tenney et al., 2019b) and lexical relations (Vulić et al., 2020).

Various problems can arise when performing probing experiments (Hewitt and Liang, 2019),

such as achieving a high probing accuracy without being due to a high mutual information between the representation and the auxiliary task labels. This has prompted much recent work on establishing more reliable methodologies for probing (Hewitt and Liang, 2019; Voita and Titov, 2020; Pimentel et al., 2020b,a).

These approaches introduce various steps such as controlling and varying model complexity and structure, including randomized control tasks and incorporating more informative metrics such as selectivity (Hewitt and Liang, 2019) and minimum description length (Voita and Titov, 2020).

To make these methods more accessible and quick to implement for any user wishing to probe the representations of their neural models in line with the evolving suggested methodologies, we introduce **Probe-Ably:** an extendable probing framework which supports and automates the application of suggested best practices for probing studies.

2 Probe-Ably

Probe-Ably[2] is a framework designed for PyTorch[3] to support researchers in the implementation of probes for neural representations in a flexible and extendable way.

The core facility provided by Probe-Ably is the encapsulation of the end-to-end experimental probing pipeline. Specifically, Probe-Ably provides a complete implementation of the core tasks necessary for probing neural representations, starting from the configuration and training of heterogeneous probe models, to the calculation and visualization of metrics for the evaluation.

The probing pipeline and the core tasks operate on a set of abstract classes, making the whole

---

[*]Equal contribution, presented in alphabetical order.

[1]The term "probing" has also been used describe stress-test style analyses, but we mean "probing" in the sense of *diagnostic classification* as in (Alain and Bengio, 2018; Pimentel et al., 2020b).

[2]Video demonstration:
https://youtu.be/lE3O_BENBxk
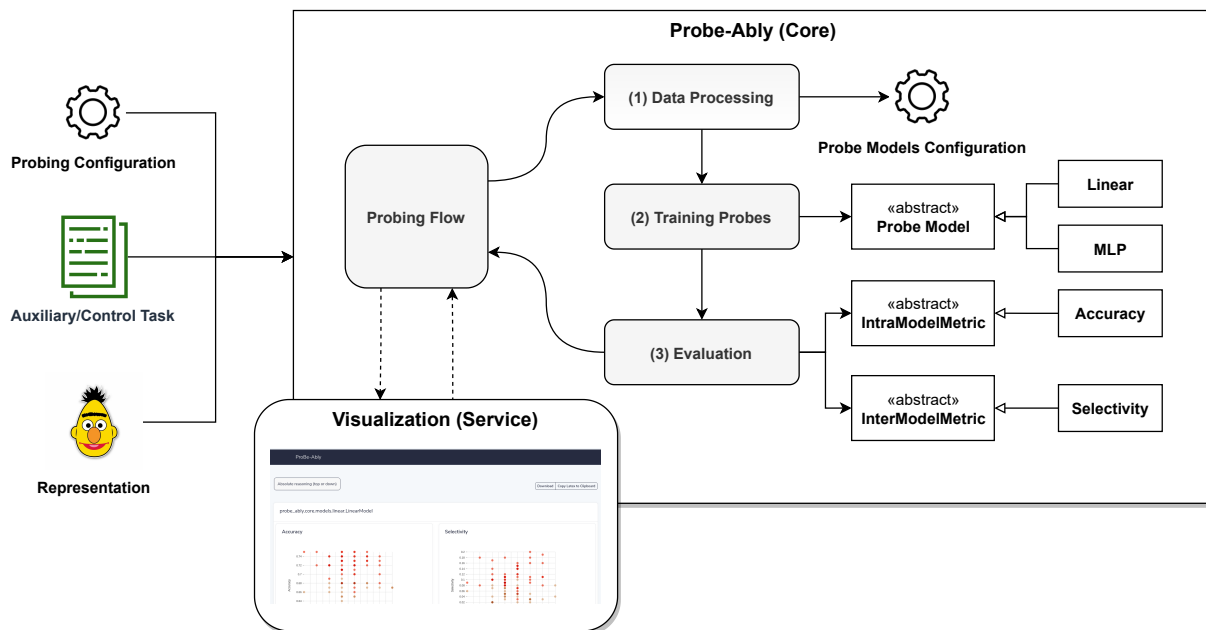[3]https://pytorch.org/

Figure 1: An overview of Probe-Ably. The core facility provided by Probe-Ably is the encapsulation of an end-to-end experimental probing pipeline. The framework offers a complete implementation and orchestration of the main tasks required for probing, together with a suite of standard probe models and evaluation metrics.

framework agnostic to the specific representation, auxiliary task, probe model, and metrics used in the concrete experiments (see Fig 1). This architectural design allows the user to:

1. Configure and run probing experiments on different representations and auxiliary tasks in parallel;

2. Automatically generate control tasks for the probing, allowing the computation of inter-model metrics such as *selectivity*;

3. Extend the suite of probes with new models without the need to change the core probing pipeline;

4. Customize, implement and adopt novel evaluation metrics for the experiments.

2.1 Probing Pipeline

In this section we describe the core components implemented in Probe-Ably.

A probing pipeline is typically composed of the following sub-tasks:

1. **Data Processing:** This task consists in data preparation and configuration of the probe models for the subsequent training task. For each representation to be probed and each auxiliary task, a requirement in this stage is

the generation of a *control task* (Hewitt and Liang, 2019), along with the selection of distinct hyperparameter configurations for the probe models. Generally, the control task can be either designed by researchers or automatically constructed by randomly assigning labels to the examples in the auxiliary task. On the other hand, the hyperparameter selection is crucial for the interpretation of the probing results, and has to guarantee a large coverage of the configuration space to allow for a significant comparison of the representations under investigation. Common methods for hyperparameter selection adopt a combination of grid search and random sampling techniques.

2. **Training Probes:** This task consists in training a set Φ of probe models. In particular, for each representation and each auxiliary task, researchers need to train probe models of different types (e.g., *linear models*, *multi-layer perceptrons*) and distinct hyperparameter configurations (e.g., hidden size, number of layers). Therefore, the number of probe models to be trained can rapidly increase with the number of representations, auxiliary tasks, and possible configurations. Let n be the number of representations to be probed, m the number of auxiliary tasks, z the number of probe models, and k the number of selected

hyperparameter configurations for each probe. The total cardinality of Φ is generally equal to $|\Phi| = n \times m \times z \times k$. Thus, because of the potentially large space of models and configurations, the training task typically represents the most demanding and time-consuming stage in the overall probing pipeline.

3. **Evaluation:** The evaluation stage consists in calculating a set of metrics for assessing the performance and quality of the probes on the auxiliary tasks. The most common metrics adopted for probing evaluation are *accuracy* and *selectivity*. Generally, these quantities are plotted against the complexity of the probe models and are used to compare the trend in the performance of different neural representations on a given auxiliary task.

Probe-Ably provides a complete implementation and orchestration of the aforementioned tasks, which are integrated by a component named *Probing Flow* (see Fig. 1).

The Probing Flow is ready to use for configuring and running standard probing experiments including hyperparameters selection via grid search. Moreover, the flow can be flexibly adapted to new models and metrics if necessary by extending the appropriate abstract classes and configuration files (additional details are described in section 3). We provide a pre-implemented suite of probe models and metrics whose details are described in sections 2.2 and 2.3.

In order to configure and run a new probing experiment, the user has to provide the following input:

- **Probing Configuration:** a JSON file describing the components and parameters for the probing experiments. This file allows specifying the concrete probe models to train on each auxiliary task, along with pre-defined training parameters such as batch-size, number of epochs and number of different hyperparameter configurations to test. Additionally, the probing configuration file can be used to indicate the metrics to use for the final evaluation.

- **Auxiliary Task:** a TSV file containing the data and labels composing the auxiliary task. Probe-Ably allows the user to configure experiments that run on more than one auxiliary task in parallel.

- **Control Task (Optional):** a TSV file containing the labels composing a control task. The control tasks are automatically generated for each auxiliary task during the data processing stage. If not provided, we assign random labels to the example in the auxiliary tasksfor.

- **Representation:** a TSV file containing the pre-trained embeddings for each example in the auxiliary task (e.g. BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019)). Similarly to the auxiliary tasks, Probe-Ably can run experiments on more than one representation in parallel.

2.2 Available Models

A common theme in probing studies is the use of structurally *simple* classifiers: two common choices are *linear* models and *multi layer perceptrons*[4].

Following works such as (Hewitt and Manning, 2019a) and (Pimentel et al., 2020a), each instantiated model comes with some approximate appropriate *complexity*. This is varied in a controlled way in order to include results for a range of model complexities: this mitigates the possible confounding effect of *overly expressive probes* which might be "memorizing" the task (Hewitt and Liang, 2019; Pimentel et al., 2020a).

For linear models $\hat{y} = W\mathbf{x} + \mathbf{b}$, we mimic (Pimentel et al., 2020a) in using the nuclear norm

$$||\mathbf{W}||_* = \sum_{i=1}^{min(|\mathcal{T}|,d)} \sigma_i(\mathbf{W}).$$

of the matrix W as the approximate measure of complexity. The rationale here is that the nuclear norm approximates the rank of the transformation matrix. The rank may be used instead in situations where there is a large number of class labels, but as it is limited by this number the nuclear norm presents a wider range of values. The nuclear norm is included in the loss (weighted by a parameter λ)

$$-\sum_{i=1}^{n} \log p(t^{(i)} \mid \mathbf{h}^{(i)}) + \lambda \cdot ||\mathbf{W}||_*$$

and is thus regulated in the training loop.

Multi-layer perceptrons are the only non-linear models currently included. Their flexibility and

---

[4]The hyperparameters of all implemented models are configurable, but we use the same default hyperparameter ranges as (Pimentel et al., 2020b).

Figure 2: Probe-Ably is integrated with a front-end visualization service, which supports researchers in consulting and plotting the results of their experiments.

simplicity has made them popular choices in probing studies. We use the *number of parameters* as an estimation of model complexity. Since sufficiently large MLP models could be prone to "fitting" noise in the data, it is especially important to monitor the *selectivity* when using this class of probes.

2.3 Available Metrics

Certain probing metrics are not tied to the output of a specific probe, but to two or more probes or training runs. As such, we have chosen to distinguish between *intra-model* and *inter-model* metrics.

Intra-Model Metrics. Individual model results and losses fall into this category. This includes the usual suspects such as *cross-entropy loss* and *accuracy*. Intra-model metrics can be used for training, model-selection and reporting purposes.

Inter-Model Metrics. An important component of assessing the reliability of a probe's result is the *selectivity* metric (Hewitt and Liang, 2019): for a fixed probe architecture and hyperparameter configuration, the auxiliary task accuracy is compared to the accuracy on a *control task*, hence incorporating the results of two trained models. This is our primary example of an inter-model metric, but this format could be useful for other probing metrics such as minimum description length (online code version) (Voita and Titov, 2020) or pareto hypervolume (Pimentel et al., 2020a), which incorporate the results of multiple models or training runs. These

are only used for *reporting* purposes, as they are external to each model's training loop.

2.4 Front-end Visualization

Probe-Ably is integrated with a front-end visualization service. The front-end is used to plot the results of each probing experiment in a user-friendly way. The service is designed to be accessible via standard web browsers, and support researchers in analysing and comparing the probing performance of each representation on different auxiliary tasks.

An example of plots included in the front-end visualization is shown in Figure 2. Each plot can be downloaded in a pdf format to be stored locally or integrated in a LaTeX project.

3 Customized Probing Experiments

Probe-Ably can be flexibly adapted and extended to run experiments on different representations, novel probe models and evaluation metrics. The following sections provide an overview of how researchers and users can customize their experiments via configuration files or implementation of new concrete classes.

For a complete guide on how to extend and customize Probe-Ably, please consult the documentation[5][6].

---

[5]Documentation:
https://ai-systems.github.io/Probe-Ably/
[6]Repository: https://github.com/ai-systems/Probe-Ably/

3.1 Configuration

Although default configurations are ready to use to run a basic set of experiments, the details of the latter can be customized according to specific needs, using the apposite probing configuration file. This pertains to aspects such as probe model choice, number of experiments, auxiliary tasks labels, input representations and custom control labels.

Therefore, the settings can be modified by providing or editing the values of the attributes in the configuration file which specifies details about auxiliary tasks, probing model/s and training regime, including paths to any custom metrics or models.

The structure of the probing configuration file is as follows:

- tasks (list)
    - task_name (attr)
    - representations (list)
        * representation_name (attr)
        * file_location (attr)
        * control_location (attr)

- probing_setup (dict)
    - train_size (attr)
    - dev_size (attr)
    - test_size (attr)
    - intra_metric (attr)
    - inter_metric (attr)
    - probing_models (list)
        * probing_model_name (attr)
        * batch_size (attr)
        * epochs (attr)
        * number_of_models (attr)

3.2 Adding a Probe Model

Custom probe models can be introduced by extending the abstract `ProbeModel` class (Fig. 1). This class inherits the methods and attributes of a `nn.Module` in PyTorch. To extend Probe-Ably with a new probe model, the user needs to implement two methods, namely `forward` and `get_complexity`.

The `forward` method is inherited from PyTorch and is adopted to compute the predictions of the probe models along with their loss function. On the other hand, the `get_complexity` method has to return a complexity measure for the model (e.g., nuclear norm, number of parameters). This method is internally used by the Probing Flow for setting up and executing the probing pipeline, and creating the right visualization for the results.

In order to make a customized probe model available for new experiments, the user needs to specify a model configuration file (JSON format) containing the path to the concrete class, together with the parameters required for its instantiation. The model configuration file is organized as follows:

- model_class (attr)

- params (list)
    - name (attr)
    - type (attr)
   - options (attr)

3.3 Adding an Evaluation Metric

Similarly to probe models, it is possible to extend Probe-Ably with new evaluation metrics. In order to add a new metric, the user can extend one of the available abstract classes (i.e., `IntraModelMetric` or `InterModelMetric`).

In this case, it is not necessary to specify a configuration file for the metrics, and the user only needs to implement the apposite function, `calculate_metrics`, that performs the appropriate computation. Subsequently, the user can adopt the new metric in a probing experiment by editing the apposite attribute in the probing configuration file.

4 Interpreting Results

We provide the following list of guidelines for interpreting results:

- Regions of low selectivity indicates a less trustworthy auxiliary task accuracy result. As accuracy increases with model complexity, keep an eye on the selectivity value: if it starts to drop again, this indicates that the probe is expressive enough to fit the randomized control task (and thus high expressivity and overfitting may be responsible for a high auxiliary task accuracy).

- We recommend a focus on *comparison of trends* between models/representations rather than probe performance on any fixed set of representations.

- These comparisons are more convincing if they are consistent across a range of probe complexities.
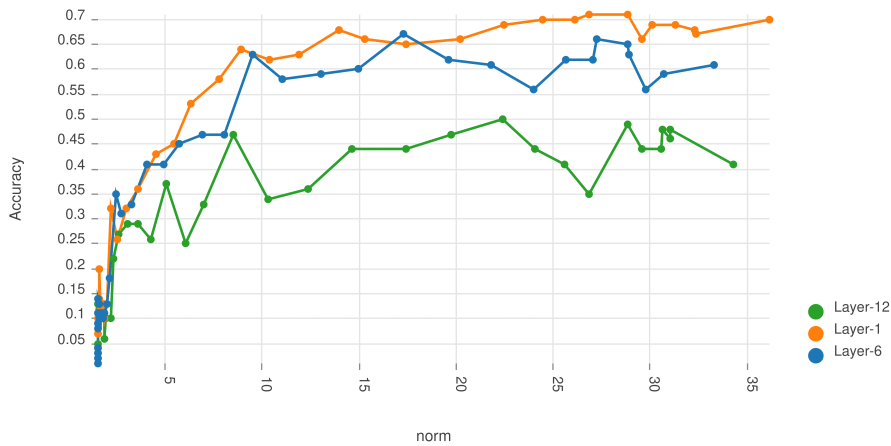
(a) Linear Model Accuracy.
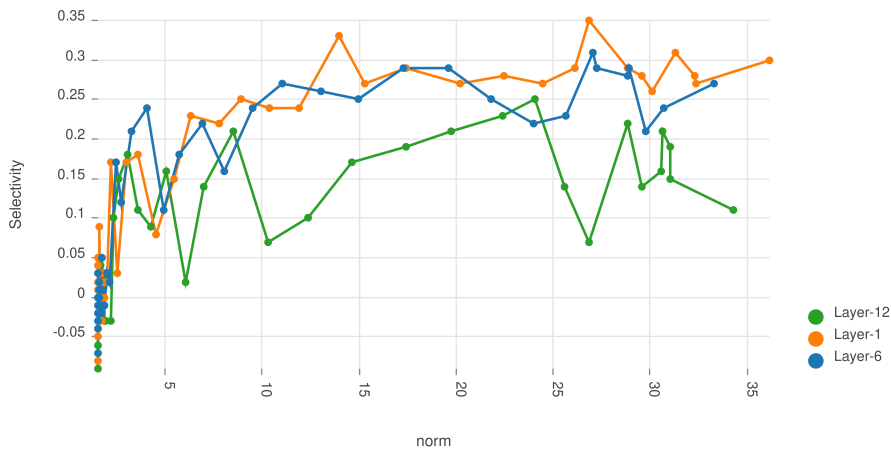
(b) Linear Model Selectivity.

Figure 3: Probing results for different layers of BERT on the *Part-Of-Speech* task using the control task presented in (Hewitt and Liang, 2019), implemented and executed through Probe-Ably (see Section 5). The results are consistent with observations in (Tenney et al., 2019a), which note that syntactic features (such as part of speech tags) are more prevalent in earlier layers of BERT.

- Note that any given probe architecture imposes a structural assumption. For example, *linear* probes may only attain a high accuracy if the representation-target relationship is linear. We recommend that these assumptions/probe model choices be guided by prior visualizations and hypothesized relationships.

- As far as possible, stick to comparing representations of the same sizes. Lower-dimensional representations may reach their maximum accuracy at lower probe complexity values; as such they may give the "appearance" of superior probe accuracy scores to larger representations. For this reason, it is also im-

portant that you investigate a sufficiently large range of model complexities.

5 Case Study

To demonstrate the Probe-Ably system, we include an implementation of a Part-Of-Speech tagging auxiliary task based on the Penn Treebank corpus (Marcus et al., 1993). It has been used multiple times in works on probing methodology (Hewitt and Liang, 2019; Voita and Titov, 2020; Pimentel et al., 2020b). We use the custom control task from (Hewitt and Liang, 2019). Using linear models as probes, we compare the probing results for different layers of BERT (bert-base-uncased)

pre-trained on the masked language modelling task (Devlin et al., 2019), across 50 probing runs. The results are consistent with observations in (Tenney et al., 2019a), which note that syntactic features (such as part of speech tags) are more prevalent in earlier layers of BERT. This case study is available as a ready-to-run example.

6 Related Work

Previous interpretability tools for neural models have focused on gradient-based methods (Wallace et al., 2019), the visualization of attention weights (Vig, 2019) and other tools focusing on NLP model explainability and interpretability (Wexler et al., 2020; Tenney et al., 2020).

The ongoing discussion on probing, auxiliary tasks and the surrounding best practices can be traced back to the early definitions in (Alain and Bengio, 2018), where it was first described as *diagnostic classification*. Early probing studies in NLP include (Zhang and Bowman, 2018) and (Tenney et al., 2019c), the former being an early example of the importance of comparing with randomized representations or labels. Further discussion has introduced control tasks and the selectivity metric (Hewitt and Liang, 2019), formalized notions of *ease of extraction* (Voita and Titov, 2020) and described other strategies for taking model complexity into account (Pimentel et al., 2020a).

7 Conclusion

While probing can be used to explore hypotheses about linguistic (or general) features present in model representations, there are various pitfalls that can lead to premature or incorrect claims. Much progress has been made in establishing better practices for probing studies, but these involve running large systematic sets of experiments employing recently-developed metrics and correctly interpreting results. **Probe-Ably** is designed to simplify and encourage the use of emerging methodological developments in probing studies, serving as a task-agnostic and model-agnostic platform for auxiliary diagnostic classification for high-dimensional vector representations.

Acknowledgements

References

Guillaume Alain and Yoshua Bengio. 2018. Understanding intermediate layers using linear classifier probes.

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.

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. 2019a. 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.

John Hewitt and Christopher D Manning. 2019b. 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.

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. *arXiv preprint arXiv:1907.11692*.

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.

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

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.

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

Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. 2020. The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models.

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

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

Jesse Vig. 2019. Visualizing attention in transformer-based language representation models. *CoRR*, abs/1904.02679.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7222–7240.

Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. AllenNLP Interpret: A framework for explaining predictions of NLP models. In *Empirical Methods in Natural Language Processing*.

J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. 2020. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65.

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.