Implications of Annotation Artifacts in Edge Probing Test Datasets

Sagnik Ray Choudhury[1*], **Jushaan Kalra**[2*]
[1]University of Michigan, [2]Wadhwani AI
`sagnikrayc@gmail.com, jushaan18@gmail.com`

Abstract

Edge probing tests are classification tasks that test for grammatical knowledge encoded in token representations coming from contextual encoders such as large language models (LLMs). Many LLM encoders have shown high performance in EP tests, leading to conjectures about their ability to encode linguistic knowledge. However, a large body of research claims that the tests necessarily do not measure the LLM's capacity to encode knowledge, but rather reflect the classifiers' ability to learn the problem. Much of this criticism stems from the fact that often the classifiers have very similar accuracy when an LLM vs a random encoder is used. Consequently, several modifications to the tests have been suggested, including information theoretic probes. We show that commonly used edge probing test datasets have various biases including memorization. When these biases are removed, the LLM encoders do show a significant difference from the random ones, even with the simple non-information theoretic probes [1].

1 Introduction

Word embeddings generated from large corpora can be expected to encode knowledge about syntax and semantics (Manning et al., 2020). This is certainly truer for the contextual ones from large language models such as `Elmo` (Peters et al., 2018), `BERT` (Devlin et al., 2019) or `RoBERTa`(Liu et al., 2019b). Edge probing (EP) tests (Liu et al., 2019a; Tenney et al., 2019a) are standard classification tasks to probe for such knowledge.

Consider the sentence "The Met is closing soon", the word "Met" functions as a noun, referring to a museum rather than the past form of the verb "meet". To determine its part of speech, humans rely on the context words "the" and "is". If a classifier predicts this token as a noun using *only* the representation from a contextual LLM encoder such as `BERT` (i.e., without using the entire sentence), it is implied that these contextual signals are encoded within the token representation itself. EP tests aim to uncover such syntactic and semantic knowledge encoded (§2).

EP tests are however *indirect* measures of such knowledge. A high accuracy of an encoder in an EP test for a grammatical property in itself does not necessarily guarantee that the said knowledge is encoded. Instead, the score should be *significantly higher* than the same from a baseline, which is typically set as static embedding encoders (Belinkov and Glass, 2017) or contextual encoders with random weights (Zhang and Bowman, 2018; Tenney et al., 2019a; Liu et al., 2019a).

NLP tasks are typically modeled by datasets, albeit imperfectly (Ravichander et al., 2021), and consequently, the performance of the encoders in the EP tests are confounded by the choice of the test dataset and its inherent biases. Despite a long history of research in edge probing tests, this problem has not been studied well (Belinkov, 2022).

To bridge this research gap, we propose three research questions.

RQ1: Are there "annotation artifacts" in the EP test datasets? Many standard NLP datasets have data points that can be solved by superficial cues, i.e., reasoning strategies unrelated to the expected causal mechanism of the task at hand (Kaushik et al., 2020). For example, Gururangan et al. (2018) show that a negation operator in the premise is a strong predictor of the "contradiction" class in the SNLI (Bowman et al., 2015) dataset. Sen and Saffari (2020) show that in popular extractive machine reading comprehension (MRC) datasets such as SQuAD (Rajpurkar et al., 2016) or HotpotQA (Yang et al., 2018), in many cases the answer phrase can be found in the first sentence of

[*]The authors contributed equally.
[1] The code is available at `https://github.com/Josh1108/EPtest.git`

the context. We analyze 17 EP test datasets across 10 tasks and find different biases in multiple of them.

RQ2: Do the EP models use heuristics? Existence of annotation artifacts in the data does not necessarily imply that the models will learn to use the related heuristics, eg., predict “contradiction” whenever the premise contains a negation. We can a) remove the biased test data (McCoy et al., 2019) or b) adversarially perturb it (Jia and Liang, 2017) and observe the performance degradation (if any) of a model. A significant degradation will indicate that the model does depend on the heuristic. Using this technique, we show that the EP classifiers trained with random encoders do indeed learn to use the heuristics to a large extent, whereas the same ones trained with pre-trained encoders do not in the same capacity.

RQ3: Do the pre-trained encoders encode grammatical knowledge better than the random encoders? A strong criticism of EP tests is that often the performances of the pre-trained and the random encoders are not **significantly different** (Zhang and Bowman, 2018). This is often attributed to the “classifier knowledge” problem, i.e., the EP classifier learns the task itself and does not necessarily depend on the encoder representations. Various information theoretic probes (Pimentel et al., 2020) have been proposed to solve this, including a popular one based on the Minimum Description Length (MDL) principle (Grünwald, 2000). In this MDL probe (Voita and Titov, 2020), a combined measure defined on the EP classifier model complexity and its performance is minimized. The MDL codelengths of contextualized representations such as `Elmo` are shown to be much lower than the corresponding random ones even when their EP test accuracies are very similar. However, we show this is not strictly necessary, and the similar performance of a pre-trained and random encoder can largely be attributed to the EP test dataset biases, as in when the “biased” data points are removed, a simple linear or MLP classifier shows a significant difference in the pre-trained vs random encoder. We investigate this further and show that Bayesian classifiers such as MDL probes are not “inherently better” in testing an encoder’s ability to encode grammatical knowledge.

2 Edge Probing

2.1 Formulation

We base our experiments on the model architecture (Figure 1) and edge probing tasks proposed by Tenney et al. (2019a) and Liu et al. (2019a), two cotemporaneous works that introduced the idea of EP tests on contextual encoders.

Given a sentence $S = [T_1, ...T_n]$ of n tokens, a span $s_k = [T_i, ...T_j]$ is defined as a contiguous sequence of tokens i to j. Depending on the task, an individual or a pair of spans is assigned a label. For example, in the Named Entity Recognition EP test, the label of the span “Barack Obama” would be PERSON. In the EP test for Coreference Resolution, a pair of spans would be labeled true or false depending on whether they were co-referent to each other in a sentence or not.

The input to the EP classifier is an embedding $e_i \in \mathcal{R}^d$ for a (pair of) span(s) and its goal is to predict its label. Token representations can be generated from the top layer (Tenney et al., 2019c) or the intermediate layers (Liu et al., 2019a) of an encoder, which is typically a large language model (LLM) such as `BERT`, `RoBERTa`, or `Elmo`. For our EP tests, we consider the top-layer representations.[2] Following Liu et al. (2019a), we generate e_i by taking an average of all token embeddings in the span, which is further averaged over the spans in the two-span tasks.

The final embedding is passed to an EP classifier (also referred to as a probe), which is either a) **MLP**: A multilayer perceptron with **one** hidden layer (1024 dim) and a RELU activation, or b) **Linear**: A linear layer without any non-linearity. For all models, the dropout (Srivastava et al., 2014) is kept at `1e-1`.

Liu et al. (2019a) used a linear layer classifier, and so did Tenney et al. (2019c), who also used a single hidden layer MLP. Follow-up work by Hewitt and Liang (2019) and Voita and Titov (2020) both used single or multiple hidden layer MLPs, but we didn’t find much difference in our experiments by increasing the number of layers. Specifically, Hewitt and Liang (2019) suggested using probes with high “selectivity”, i.e., they should have a high accuracy on an EP task, but a low score when the

[2]Tenney et al. (2019c) uses both the top layer and a mixed representation from all layers, and Hewitt and Manning (2019) uses the top layer. As there is not a significant difference in the mixed vs top layer representations in Tenney et al. (2019c), we leave the mixed representations for future work.

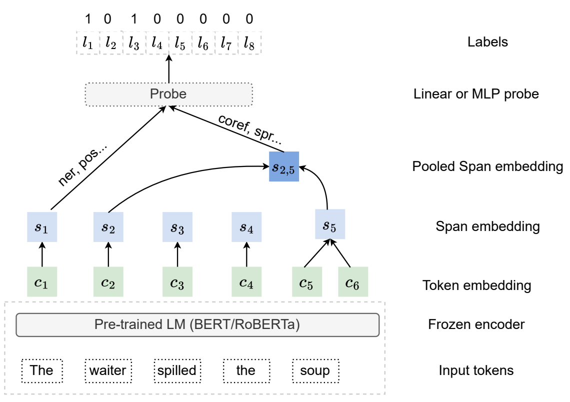
1 0 1 0 0 0 0 0
l_1 l_2 l_3 l_4 l_5 l_6 l_7 l_8
Labels
Probe
Linear or MLP probe
ner, pos...
coref, spr...
$s_{2,5}$
Pooled Span embedding
s_1 s_2 s_3 s_4 s_5
Span embedding
c_1 c_2 c_3 c_4 c_5 c_6
Token embedding
Pre-trained LM (BERT/RoBERTa)
Frozen encoder
The waiter spilled the soup
Input tokens

Figure 1: The architecture for edge probing tasks.

labels of the same task are randomized (control tasks). They concluded that simpler, i.e., lower depth probes showed higher selectivity, which is another reason for our probe choice.

Crucially, during training, *only the parameters of the probe are changed* and the encoder below is kept frozen. If the LLM encoder truly encodes certain types of syntactic (eg., identifying constituent types) or semantic (coreference relation between phrases) knowledge, we can expect it to have a significantly higher performance in the related EP test than an encoder of the same architecture but with random weights.

2.2 Edge Probing Tasks and Datasets

To ensure wide coverage, we experiment with 17 EP datasets involving 10 different NLP tasks that have been used before in Tenney et al. (2019c) and Liu et al. (2019a). The tasks are described below, the dataset statistics are presented in Table 1.

Part of Speech Tagging. POS tagging is a syntactic task, where each token is assigned one of the possible part-of-speech tags. e.g. "$[Napoleon]_{NNP}$ Bonaparte was the emperor of France", where NNP stands for "Proper Noun, Singular". We use 3 different datasets for this task: the OntoNotes corpus (Weischedel, Ralph et al., 2013), the Penn Treebank (PTB) corpus (Marcus et al., 1993) and the Universal Dependencies English Web Treebank (EWT) corpus (Silveira et al., 2014).

Named Entity Recognition. NER is a task to predict the pre-defined semantic category of a span such as persons, organizations, date, and quantity, e.g. - "$[Napoleon\ Bonaparte]_{PERSON}$ was the emperor of France." We use the OntoNotes corpus and the CoNLL 2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003).

Dataset	#Points in the EP test data		
	Train	Test	Dev
Part of Speech Tagging			
EWT-PoS[2]	204,607	25,097	25,150
PTB-PoS[2]	950,028	56,684	40,117
OntoNotes-PoS[1]	2,070,382	212121	290,013
Named Entity Recognition			
CoNLL-2003-NER[2]	203,621	46,435	51,362
OntoNotes-NER[1]	128,738	12,586	255,133
Coreference Resolution			
DPR[1]	1,787	949	379
OntoNotes-Coref[1]	207,830	27,800	26,333
Syntactic Dependency Classification			
EWT-Syn-Dep-Cls[2]	203,919	25,049	25,110
PTB-Syn-Dep-Cls[2]	910,196	54,268	38,417
Syntactic Dependency Prediction			
EWT-Syn-Dep-Pred[1,2]	383,462	45,901	46,155
PTB-Syn-Dep-Pred[2]	1,820,225	108,529	76,820
Semantic Proto-Role Labeling			
SPR-1[1]	7,611	1,055	1,071
SPR-2[1]	4,925	582	630
One Task Datasets			
CoNLL-Chunking[2]	211,727	47,377	-
OntoNotes-Const[1]	1,851,590	190,535	255,133
OntoNotes-SRL[1]	598,983	61,716	83,362
Semeval-Rel-Cls[1]	8,000	2,717	-

Table 1: Statistics for the EP datasets used in this paper, with the tasks and in which paper they were used in: Tenney et al. (2019c)[1] or Liu et al. (2019a)[2].

Constituency Labeling. The goal of this task is to recover the constituency parse tree of a sentence, eg., "$[Napoleon\ Bonaparte]_{NP}$ was the emperor of France.", where NP stands for "Noun Phrase". We use the OntoNotes corpus for this task.

POS, NER, and Constituency Labeling are usually modeled as token-level *tagging* tasks using the standard BIO format (Pradhan et al., 2013) but in the EP tests, they are classification problems. The classifier predicts the label for a token or a span, which can be one of the pre-defined ones, eg., "ADJ" for Part of Speech, "PER" for NER, or "PP" for Constituency Labeling or "None" if the input can not be assigned a label. Importantly, the classifier has access to only the token representations and not the whole sentence.

Coreference Resolution. Coreference resolution is the task of finding anaphoric relations between spans in a text: e.g. "$[Barack\ Obama]_1$ is an ex-US president, $[He]_2$ lives in DC with his wife Michelle." In the EP tests, this reduces to a binary classification task: given two spans, predict whether they refer to each other ("Barck Obama", "he": true) or not ("Michelle", "he": false). We use the OntoNotes corpus as well as the Definite Pronoun resolution (DPR) dataset (Rahman and

Ng, 2012), which is considered more challenging.

Semantic Role Labeling. In the SRL task, the goal is to understand *semantic* roles (who did what to whom and when) between spans (argument) in a sentence and a verb (predicate): eg., "$[The\ waiter]_{AGENT}$ $[spilled]_{VERB}$ $[the\ soup]_{THEME}$. In the EP tests, this is modeled as a two-span multi-class classification task for which the OntoNotes corpus is used.

Chunking. While a constituency parse of a sentence is a hierarchical structure, chunking (Abney, 1992) divides the text into syntactically related non-overlapping groups of words. We use the CoNLL-2000-Chunking corpus (Tjong Kim Sang and Buchholz, 2000). For the EP tests, this is a one-span multi-class classification problem.

Semantic Proto-Role Labeling. Proposed by Reisinger et al. (2015), this is a task of annotating detailed, non-exclusive semantic attributes, such as change of state or awareness, over predicate-argument pairs as in SRL. Similar to the SRL EP test, this is modeled as a two-span classification problem, but as there can be more than one potential attribute of the predicate-argument relation, this is a multi-label task. We used two datasets, SPR-1 (Teichert et al., 2017), and SPR-2 (Rudinger et al., 2018), derived from the Penn Treebank and the English Web Treebank respectively.

Relation Classification. Initially proposed by (Girju et al., 2009), Relation Classification is the task of predicting the relation that holds between two nominals, from a given knowledge base. We use the SemEval dataset from (Hendrickx et al., 2010). For the EP tests, this reduces to a two-span multi-class classification task.

Syntactic Dependency Classification. Given representations of two tokens from a sentence, $[head]$ and $[mod]$, the task is to predict the syntactic relationship between the two. We use the Penn Treebank (Marcus et al., 1993) and English Web Treebank (Silveira et al., 2014) datasets. For EP tests, this boils down to a two-span multi-class classification task.

Syntactic Dependency Prediction. The goal of this task is to find whether a dependency arc exists between two tokens in their syntactic structure. We use the Penn Treebank and the English Web Treebank, the same as in the classification variant. This is a two-span binary classification task for EP tests.

Where development data was not available from the source, 10% of the data from the training set was reserved for validation. In a few other cases, the testing set had labels not present in the training set, these data points were discarded. The final datasets (bar the licensed ones) will be made available.

3 Annotation Artifacts in EP Test Datasets

Our analysis indicates that almost all EP test datasets have a significant repetition bias: many samples in the training data are repeated in the test. However, their labels may always not be the same, for example, in the NER EP test, the span "Google" might have the label "ORG" or "O" depending on whether the span refers to the company or the search engine developed by it.

We ask two questions. In a test dataset, in what percentage of cases a test data point is in the training data and has only one label? For example, in the NER datasets, if the span "Google" appears in both the training and the test dataset with the *only* label "Org", the EP classifier can successfully classify it by memorization. We call it the **Mem-Exact** heuristic.

Even if the training data contains multiple labels for a span (eg., both "ORG" and "O"), the EP classifier might be able to successfully classify it in the test data by simply learning the label distribution for the span and not the inherent contextual relationships. In the **Mem-Freq** heuristic we find the percentage of test data points that are present in the training data and can be classified correctly using the training label distribution. We also consider a baseline: the **Mem-Uniform** heuristic where instead of the true label distribution the class labels can be predicted by sampling from a uniform distribution.

Table 2 shows that a large percentage of data points indeed can be classified heuristically, i.e., the dataset has significant biases. Importantly, if an EP classifier does adopt a heuristic, it would need no specific representation for the spans, let alone from a pre-trained or a random one.

4 Do the EP Models Use Heuristics?

Based on the dataset biases discovered in §3, we hypothesize that the EP classifiers can use heuristic algorithms, but there will be a difference in the random vs pre-trained encoders. Specifically, *EP test classifiers with random encoders will learn to use various heuristics* as the input representations

Dataset	Mem-Exact	Mem-Freq	Mem-Uniform
EWT-PoS	89.73	42.85	48.03
PTB-PoS	97.11	42.03	52.62
OntoNotes-PoS	98.06	65.40	35.26
CoNLL-2003-NER	86.87	28.15	67.93
OntoNotes-NER	70.53	23.40	55.58
DPR	28.98	0.21	15.81
OntoNotes-Coref	36.55	16.64	26.51
EWT-Syn-Dep-Cls	37.98	4.56	34.30
PTB-Syn-Dep-Cls	62.17	12.50	51.81
EWT-Syn-Dep-Pred	42.44	13.38	31.99
PTB-Syn-Dep-Pred	68.04	17.37	47.75
SPR-1	5.2	0.47	4.55
SPR-2	7.2	0.42	1.37
CoNLL-Chunking	89.89	57.72	33.88
OntoNotes-Const	45	17.79	33.57
OntoNotes-SRL	32.07	6.98	26.76
Semeval-Rel-Cls	3.35	0.11	3.3

Table 2: Accuracy (in %) of the heuristic algorithms.

themselves do not provide much information. On the other hand, the same classifier models with pre-trained encoders will tend to not make use of such heuristic mechanisms. If the hypothesis is true, we will see a *significant drop in the performance with the random encoders compared to the pre-trained encoders* when the "heuristically classifiable" data points are removed from the test data.

4.1 Experimental Setup

We use 4 encoders - `BERT` (the base-cased version), `RoBERTa` (the base version), and their randomized versions. Following Tenney et al. (2019c), the random encoders are the same LLM models randomly initialized (Glorot and Bengio, 2010) as it is done before pre-training.

For each encoder and EP classifier model (Linear and MLP, see §2) we train 3 models.[3] The models showed little variance on the test data (within 0.1% of the average), therefore, we chose the best model for the subsequent experiments.

4.2 Results and Analysis

For each heuristic algorithm in §3, we create a "filtered dataset" consisting of the points that can not be classified using the said algorithm. For each "EP model" (an encoder + EP classifier), we calculate the accuracy score on the original and the filtered datasets and report the "drop", i.e., the relative reduction percentage: $(acc_{\text{original}} - acc_{\text{filtered}}) * 100/acc_{\text{original}})$. A *negative* drop indicates that the EP model performed *better* on the original dataset vs the filtered one.

Tables 3 and 4 show the results. Firstly, there is an accuracy drop in both pre-trained (base) and random encoders with **all** "Mem-Exact" datasets, indicating these datasets are more difficult in general and both these encoders use the exact memorization heuristic (Augenstein et al., 2017) to some extent. On the other hand, they do not use the baseline "Mem-Uniform" heuristic as expected, as evidenced by the increased accuracy in the filtered dataset.

More importantly, in a large number of EP datasets (11 out of 17), the accuracy drop in the random encoder is higher (indicated by **bold**) than that in the pre-trained encoders. Also, this pre-trained-v-random accuracy drop difference in the filtered datasets is **significant**, i.e., $> 100\%$, in 8 out of 11 cases. On the other hand, when the random encoders show a lower drop than the pre-trained encoders, the difference is almost always negligible (eg., EWT-Syn-Dep-Pred). In 4 of the remaining 6 datasets where we do not see a higher drop in the random encoders - Semeval-Rel-Cls, SPR-1, SPR-2, and Definite Pronoun Resolution, the filtered version of the datasets do not differ much from the original: as only a small percentage of the data points can be solved by the **Mem-Exact** heuristic.

The accuracy drops are consistent across the encoder types and EP classifiers. For example, on the EWT-PoS dataset, the `BERT`-base and the `RoBERTa`-base encoders have similar drops both with the Linear and the MLP EP classifiers as do the random versions of these encoders among themselves. A surprising finding is that the drop pattern is task-dependent. Among the tasks with multiple datasets (Table 4), in all POS, NER, and Syntactic Dependency Classification datasets, the random encoders show a higher drop but in the Syntactic Dependency Prediction and Semantic Proto-Role Labeling tasks, the opposite is true for all datasets. This is not correlated with either the dataset size or the number of labels: both Syntactic Dependency Prediction and Classification tasks have a similar number of training data points, and the Classification task has ≈ 40 labels whereas the Prediction one has only 2.

The OntoNotes-Coref dataset presents an interesting case as the accuracy scores **increase** in the

[3]Each model was trained for 3 epochs with a batch size of 16 using the AdamW optimizer (Kingma and Ba, 2015), a learning rate of `1e-3` and a linear warmup learning rate scheduler (Howard and Ruder, 2018).

Dataset	Encoder	Version	Linear			MLP		
			$\%\Delta_{\text{Mem-Ex}}$	$\%\Delta_{\text{Mem-Freq}}$	$\%\Delta_{\text{Mem-Unif}}$	$\%\Delta_{\text{Mem-Ex}}$	$\%\Delta_{\text{Mem-Freq}}$	$\%\Delta_{\text{Mem-Unif}}$
CoNLL-Chunking	`BERT`	base	12.03	6.36	0.95	10.36	4.65	0.99
		random	**25.47**	**27.61**	*0.44*	**32.4**	**34.6**	**9.73**
	`RoBERTa`	base	10.55	5.34	1.02	9.18	3.86	0.88
		random	**23.5**	**26.36**	*0.21*	**31.69**	**34.45**	**10.65**
OntoNotes-Const	`BERT`	base	13.34	4.38	6.91	10.75	3.33	5.62
		random	**17.42**	**7.66**	**7.36**	**18.91**	**7.02**	**8.95**
	`RoBERTa`	base	14.08	4.34	7.32	10.77	3.27	5.69
		random	**18.55**	**8.04**	**7.97**	**18.44**	**6.94**	**8.68**
OntoNotes-SRL	`BERT`	base	7.47	1.9	6.27	5.15	0.87	4.27
		random	**12.77**	**2.73**	**10.36**	**14.63**	**3.01**	**9.92**
	`RoBERTa`	base	7.83	1.25	6.52	5.12	0.83	4.29
		random	**12.7**	**2.49**	**10.37**	**13.83**	**2.99**	**9.21**
Semeval-Rel-Cls	`BERT`	base	1.9	0.09	1.04	0.72	0.05	0.67
		random	*0.75*	*-0.13*	*0.84*	*0.19*	*0.04*	*0.27*
	`RoBERTa`	base	1.4	0.04	1.3	0.78	0.02	0.72
		random	*0.53*	*0*	*0.33*	*1.36*	*-0.04*	*1.17*

Table 3: The effect of heuristic algorithms on EP tasks where each task has *only one* dataset. Each model is tested with the original test data and three *filtered* test datasets. The $\%\Delta_{\text{Mem-Ex}}$ shows the *percentage drop in the accuracy score* from the original test dataset when the models are tested on the dataset filtered by "Mem-Exact" (the others follow the same nomenclature). **Bold** (*Italicized*) indicates that the random encoder shows a much higher (lower) $\%$ drop on the filtered dataset than the base encoder.

filtered datasets. This binary classification dataset has a significant label imbalance: 78.33% of the test data has a negative label. If the dataset is re-sampled to make the distribution balanced, a) the accuracy score decreases as expected; b) the accuracy drops in the random encoders become higher by 19.28 and 7.08 points than the `BERT` and `RoBERTa` encoders respectively when using the MLP classifier. With the Linear classifier, these numbers are 3.45 and 8.7.

Overall, it is clear that in many EP test datasets, the random encoders perform significantly worse than the pre-trained encoders on the set of data points that are **not** heuristically classifiable (specifically, by the **Mem-Exact** heuristic). In other words, they resort to the heuristics more than the pre-trained ones. This proves our hypothesis.

5 EP Test Results: Random vs Pre-Trained Encoders

Previously, we have shown that the random encoders show a significant memorization bias compared to the pre-trained ones. How does that affect the EP test results? Table 5 and Table 6 show the EP test results for the pre-trained and random encoders on the "Mem-Exact" filtered datasets - except for the OntoNotes-Coref one, where we use the *balanced* dataset. As expected, in almost all cases the pre-trained encoders have a **significantly higher** accuracy than the random ones. Compare this with Voita and Titov (2020) where in 4 out of 7 datasets that is not the case.

MDL Probe. Voita and Titov (2020) show that for many EP datasets, a contextual encoder (`ElMo`) has the same performance as a random encoder. This leads to the conclusion that the EP tests, in reality, measure the classifiers' ability to learn the EP task and do not reflect the knowledge encoded in the representations themselves. To solve this, a minimum description length (MDL) probe is proposed. We have already seen that the pre-trained vs random issue is mitigated in the filtered datasets, but had we used the MDL probes, would our conclusions have changed? More importantly, are the MDL probes necessary in the EP test datasets with a large number of samples (Table 1)?

In its original formulation, the Minimum Description Length (MDL) principle is a Bayesian model selection technique. A model class $\mathbf{M}$ is a set of models M_i, for example, $\mathbf{M}$ can be "all polynomials of degree 3" and one M_i can be $5x^3$. Between two model classes $\mathbf{M_a}$ and $\mathbf{M_b}$, the better model class is the one with the lower *stochastic complexity*.

Given a supervised classification dataset D with data points $d_i = \langle x_i, y_i \rangle$, a model M defines a probability distribution $P(y_i|x_i)$. From the Kraft-Mcmillan inequality, there exists a code C for D with the code length $L_C(D) = -logP(D) = \sum_{i=1}^{n} -logP(d_i)$. Naturally, a better model fit cor-

Dataset	Encoder	Version	Linear			MLP		
			$\%\Delta_{\text{Mem-Ex}}$	$\%\Delta_{\text{Mem-Freq}}$	$\%\Delta_{\text{Mem-Unif}}$	$\%\Delta_{\text{Mem-Ex}}$	$\%\Delta_{\text{Mem-Freq}}$	$\%\Delta_{\text{Mem-Unif}}$
EWT-PoS	BERT	base	15.51	2.45	1.5	15.76	1.95	1.75
		random	**59.15**	**17.68**	*-2.9*	**57.34**	**15.67**	*0.69*
	RoBERTa	base	12.35	1.95	1.46	12.39	1.67	1.38
		random	**60.4**	**17.54**	*-2.88*	**60.01**	**16.1**	*0*
PTB-PoS	BERT	base	13.88	0.51	1.26	13.17	0.59	1.18
		random	**62.46**	**9.68**	*-2.86*	**41.06**	**5.84**	**1.92**
	RoBERTa	base	13.94	0.66	1.06	12.89	0.54	1.17
		random	**69.64**	**10.63**	*-3.71*	**41.34**	**5.98**	**2.03**
OntoNotes-PoS	BERT	base	15.45	2.77	0.13	14.75	2.25	0.3
		random	**71.91**	**38.31**	*-9.48*	**65**	**24.5**	*-2.94*
	RoBERTa	base	14.9	2.55	0.21	13.63	2.45	0.15
		random	**71.73**	**40.59**	*-10.35*	**47.75**	**27.98**	*-3.67*
CoNLL-2003-NER	BERT	base	9.16	0.63	3.56	10.67	0.6	4.07
		random	**34.47**	**2.43**	**13.12**	**32.56**	**2.81**	**9.54**
	RoBERTa	base	8.62	0.58	3.47	8.23	0.48	3.19
		random	**34.1**	**2.39**	**12.98**	**31.36**	**2.38**	**9.7**
OntoNotes-NER	BERT	base	5.35	0.32	3.94	5.35	-0.51	4.43
		random	**29.3**	**14.98**	**5.15**	**35.47**	**13.31**	**8.37**
	RoBERTa	base	5.41	0.52	3.65	4.55	-1.04	4.43
		random	**29.24**	**14.57**	**5.28**	**29.26**	**9.43**	**7.03**
EWT-Syn-Dep-Cls	BERT	base	8.7	0.13	8.36	6.72	0.46	6.09
		random	**29.36**	**0.69**	**27.95**	**26.98**	**2.01**	**24.13**
	RoBERTa	base	8.12	-0.04	7.91	6.74	0.48	6.04
		random	**30.48**	**0.86**	**28.57**	**26.94**	**1.88**	**24.39**
PTB-Syn-Dep-Cls	BERT	base	9.23	0.14	7.92	6.76	0.48	5.2
		random	**36.12**	**0.97**	**29**	**32.28**	**2.35**	**23.96**
	RoBERTa	base	9.44	0.29	8.03	6.72	0.51	5.19
		random	**36.76**	**0.73**	**29.8**	**32.17**	**2.25**	**24.05**
EWT-Syn-Dep-Pred	BERT	base	4.52	0	4.59	5.27	1.29	4.14
		random	**5.66**	**0.74**	**4.95**	*4.71*	*1.14*	*3.46*
	RoBERTa	base	6.64	0.91	5.57	5.05	1.18	3.97
		random	*5.58*	*0.86*	*4.66*	*4.95*	*1.04*	*3.66*
PTB-Syn-Dep-Pred	BERT	base	6.49	0.45	5.43	3.72	1.51	2.5
		random	*4.77*	*0.3*	*3.68*	*2.67*	*1.17*	*1.93*
	RoBERTa	base	7.47	0.96	5.73	4.58	2	2.91
		random	*3.57*	*-0.09*	*3.12*	*2.55*	*1.04*	*1.69*
SPR-1	BERT	base	0.38	0.04	0.31	0.35	0.05	0.29
		random	*0.08*	*0*	*0.1*	**0.66**	*-0.02*	**0.73**
	RoBERTa	base	0	0	-0.01	0.39	0.04	0.33
		random	*0.07*	0	**0.08**	*0.36*	**0.05**	**0.36**
SPR-2	BERT	base	1.96	0	0.31	1.08	0	0.22
		random	*1.37*	0	*0.17*	**1.8**	0	**0.29**
	RoBERTa	base	1.55	0	0.28	1.65	0	0.34
		random	1.55	0	*0.24*	*1.5*	0	*0.26*
DPR	BERT	base	4.37	0	1.52	0.73	-0.22	1.13
		random	*0.92*	*-0.2*	*2.91*	*0.92*	*0.22*	*0.2*
	RoBERTa	base	1.26	0.2	1.93	0.81	0.19	3.34
		random	*0.38*	*0.22*	*0.84*	*-0.5*	*-0.22*	*-1.26*
OntoNotes-Coref	BERT	base	-2.92	-1.61	-0.99	0.84	0.58	0.88
		random	*-7.42*	*-4.59*	*-3.18*	**0.78**	*1.55*	*0.72*
	RoBERTa	base	-3.47	-1.66	-1.45	1.05	0.87	0.86
		random	*-6.76*	*-4.84*	*-2.46*	*0.4*	**1.15**	*0.58*

Table 4: The effect of heuristic algorithms on EP tasks where each task has *multiple* datasets. The structure follows Table 3.

responds to higher probability values and lower code lengths.

The stochastic complexity of the dataset D with respect to the model class $\mathbf{M}$ is the shortest code length of D when D is encoded with the help of class $\mathbf{M}$. Given M and D, one can find the

Dataset	BERT pre-trained	BERT random	RoBERTa pre-trained	RoBERTa random
EWT-PoS	79.74	**25.93**	83.71	**24.61**
PTB-PoS	83.33	**24.79**	83.52	**19.90**
OntoNotes-PoS	81.62	**17.43**	83.30	**17.29**
CoNLL-2003-NER	87.96	**54.26**	88.65	**54.42**
OntoNotes-NER	87.95	**35.83**	88.83	**35.11**
DPR	48.37	*49.70*	50.15	49.55
OntoNotes-Coref	70.53	**60.41**	72.9	**58.44**
EWT-Syn-Dep-Cls	69.35	**32.60**	71.78	**31.36**
PTB-Syn-Dep-Cls	78.58	**33.59**	79.19	**32.96**
EWT-Syn-Dep-Pred	66.54	**62.66**	67.72	**63.63**
PTB-Syn-Dep-Pred	64.45	63.14	63.93	*64.20*
SPR-1*	70.68	**60.66**	67.21	**61.38**
SPR-2*	75.12	**69.88**	76.61	**70.41**
CoNLL-Chunking	81.43	**50.29**	84.40	**50.81**
OntoNotes-Const	62.17	**38.15**	62.81	**38.21**
OntoNotes-SRL	67.79	**44.45**	68.71	**44.96**
Semeval-Rel-Cls	55.10	**22.39**	50.88	**24.35**

Table 5: Accuracy scores (Micro f1 for *) on the filtered EP test dataset, with the **Linear** classifier. **Bold** indicates where the random encoders have a *significantly lower* score than the pre-trained ones, and *Italicized* indicates they have a higher score.

Dataset	BERT pre-trained	BERT random	RoBERTa pre-trained	RoBERTa random
EWT-PoS	79.93	**31.48**	84.33	**28.83**
PTB-PoS	84.31	**48.84**	84.86	**47.99**
OntoNotes-PoS	82.98	**27.98**	84.17	**40.82**
CoNLL-2003-NER	86.6	**57.34**	89.44	**58.14**
OntoNotes-NER	84.55	**38.53**	87.38	**41.77**
DPR	59.94	**49.7**	51.63	50.3
OntoNotes-Coref	85.91	**73.09**	87.4	**73.12**
EWT-Syn-Dep-Cls	80.57	**42.43**	81.63	**42.35**
PTB-Syn-Dep-Cls	86.85	**44.12**	87.42	**44**
EWT-Syn-Dep-Pred	79.26	**72.65**	81.38	**73.41**
PTB-Syn-Dep-Pred	86.72	**80.05**	86.19	**81.17**
SPR-1*	81.97	**63.68**	83.7	**63.5**
SPR-2*	77.91	**72.06**	77.31	**71.5**
CoNLL-Chunking	84.88	**50.15**	86.97	**50.54**
OntoNotes-Const	70.55	**49**	71.05	**49.44**
OntoNotes-SRL	80.26	**51.06**	80.86	**51.34**
Semeval-Rel-Cls	65.04	**26.01**	63.8	**26.03**

Table 6: Accuracy scores (Micro f1 for *) on the filtered EP test dataset, random vs pre-trained encoders with the **MLP** classifier. **Bold** indicates where the random encoders have a *significantly lower* score than the pre-trained ones.

M_i (with parameters θ_i) through maximum likelihood estimation that leads to the maximum P, hence the minimum code length $L(D|\hat{\theta}(D)) = -logP(D|\hat{\theta}(D))$.

Crucially, we are not allowed to fit a different θ and build a new code C' with each new dataset D'. Ideally, we would like to have a single code C^* that can yield the minimum length for *all* datasets but that is not possible if $\mathbf{M}$ contains more than one model. Nevertheless, it is possible to construct C^* such that: (Grünwald, 2000)

$$L_{C^*}(D) = L(D|\hat{\theta}(D)) + K^* \quad (1)$$

Equation 1 is a combination of the "goodness of model fit" (better estimate of $\hat{\theta} \implies$ smaller code length) and the model complexity (K^*). K^* can be approximated for a *regular* model class $\mathbf{M}$ containing models with p parameters as:

$$K^* \approx \frac{p}{2} logn + C_k \quad (2)$$

where n is the length of the dataset D and C_k is negligible for large n (Grünwald, 2000).

Voita and Titov (2020) calculate the code lengths of two EP classifiers with random and pre-trained encoders and show that the second one has a lower code length. This is one of the reasons for using the minimization of codelengths (which is termed "MDL probe") as an alternative to normal classifiers. In the implementation, these two encoders are frozen and hence provide two *datasets*, so the model selection problem is essentially inverted: there is one model class (say, the class of Linear models) and two datasets (token encodings from random and pre-trained encoders): what would two different code lengths mean?

Voita and Titov (2020) follows Blier and Ollivier (2018) in determining code lengths for DNN models because the approximation in eq. (2) is not correct for complex DNNs. But the EP classifiers are not DNNs, they are simple linear models whose code lengths should be approximable by eq. (2). But as eq. (2) shows, the code lengths are not dependent on the datasets as long as the number of data points is large, which is true for most EP datasets (see Table 1). This raises the question of whether the MDL probe is an inherently better choice for comparing the encoding of information in the encoders.

6 Related Work

Previous research has primarily focused on studying different aspects of pre-trained language models (LMs), such as linguistic knowledge (Liu et al., 2019a) and attention patterns (Clark et al., 2019).

The paradigm of classifier-based probing tasks is well-researched (Ettinger et al., 2016) and has gained popularity with the introduction of benchmark EP datasets that we utilize here (Tenney et al., 2019a). Typically, internal layers of large language or machine translation models are used as features for auxiliary prediction tasks related to syntactic properties, such as part-of-speech (Shi et al., 2016; Blevins et al., 2018; Tenney et al., 2019b), tense (Shi et al., 2016; Tenney et al., 2019b), or subject-verb agreement (Tran et al., 2018; Linzen et al., 2016). For a comprehensive survey, refer to Belinkov and Glass (2019).

EP tests are not direct evaluations of models since they use another model (called probe) to extract and evaluate the linguistic features within an encoding. Because of this, it is not clear if the results reflect the quality of encoding or the probe's ability to learn the task (Hewitt and Liang, 2019; Voita and Titov, 2020; Pimentel et al., 2020). We delve into this topic further in Section 5. Additional details can be found in Belinkov (2022).

7 Conclusion

EP tests are classification tasks to measure an LLM's ability to encode syntactic and semantic knowledge. However, in many EP datasets, there is not a significant difference between the random vs pre-trained encoders, which raises questions about the validity of the tests (the "classifier knowledge" problem). We analyze 17 datasets across 10 datasets to find various biases and show that the EP classifiers are more prone to use heuristic mechanisms when random encoders are used instead of the pre-trained ones. When the dataset biases are removed, the pre-trained encoders do show a significant difference from the random ones as expected. Information-theoretic probes have been proposed before to solve the "classifier knowledge" problem, we show why they might not be necessary. Future work would extend the findings of this study to fine-tuned models.

Limitations

There are two important limitations of this study: 1. We analyze a large number of standardized EP test datasets that have been extensively used before, but the paradigm of diagnostic classifiers is quite broad and our findings should not be automatically extended to datasets not used in this study. Also, we do not propose an automated way to remove biases from the existing or newly created datasets. 2. While we argue the popular MDL probe might not be necessary for all EP test datasets (particularly, the ones with a large number of data points), this paper should not be construed as a general criticism of the MDL probes or the area of information-theoretic probing.

Acknowledgement

We thank the anonymous reviewers for their helpful suggestions.

References

Steven P Abney. 1992. Parsing by chunks. *Principle-based parsing: Computation and Psycholinguistics*, pages 257–278. 4

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in named entity recognition: A quantitative analysis. *Comput. Speech Lang.*, 44:61–83. 5

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219. 1, 9

Yonatan Belinkov and James R. Glass. 2017. Analyzing hidden representations in end-to-end automatic speech recognition systems. *CoRR*, abs/1709.04482. 1

Yonatan Belinkov and James R. Glass. 2019. Analysis methods in neural language processing: A survey. *TACL*, 7:49–72. 9

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep rnns encode soft hierarchical syntax. In *ACL*, pages 14–19. Association for Computational Linguistics. 9

Léonard Blier and Yann Ollivier. 2018. The description length of deep learning models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2220–2230. 8

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics. 1

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. 8

Jacob Devlin, Ming - Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics. 1

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, RepEval@ACL 2016, Berlin, Germany, August 2016*, pages 134–139. Association for Computational Linguistics. 9

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2009. Classification of semantic relations between nominals. *Language Resources and Evaluation*, 43(2):105–121. 4

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org. 5

Peter Grünwald. 2000. Model selection based on minimum description length. *Journal of mathematical psychology*, 44(1):133–152. 2, 8

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL-HLT*, pages 107–112. Association for Computational Linguistics. 1

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden. Association for Computational Linguistics. 4

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 2019, Hong Kong, China, November 3-7, 2019*, pages 2733–2743. Association for Computational Linguistics. 2, 9

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. 2

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics. 5

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics. 2

Divyansh Kaushik, Eduard H. Hovy, and Zachary Chase Lipton. 2020. Learning the difference that makes A difference with counterfactually-augmented data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. 1

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 5

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *TACL*, 4:521–535. 9

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. 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. 1, 2, 3, 8

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692. 1

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*, 117(48):30046–30054. 1

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. 3, 4

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics. 2

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. 1

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics. 2, 9

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics. 3

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The Winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789, Jeju Island, Korea. Association for Computational Linguistics. 3

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392. The Association for Computational Linguistics. 1

Abhilasha Ravichander, Yonatan Belinkov, and Eduard H. Hovy. 2021. Probing the probing paradigm: Does probing accuracy entail task relevance? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 3363–3377. Association for Computational Linguistics. 1

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the Association for Computational Linguistics*, 3:475–488. 4

Rachel Rudinger, Adam Teichert, Ryan Culkin, Sheng Zhang, and Benjamin Van Durme. 2018. Neural-Davidsonian semantic proto-role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 944–955, Brussels, Belgium. Association for Computational Linguistics. 4

Priyanka Sen and Amir Saffari. 2020. What do models learn from question answering datasets? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2429–2438. Association for Computational Linguistics. 1

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *EMNLP*, pages 1526–1534. The Association for Computational Linguistics. 9

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA). 3, 4

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958. 2

Adam R. Teichert, Adam Poliak, Benjamin Van Durme, and Matthew R. Gormley. 2017. Semantic proto-role labeling. In *AAAI Conference on Artificial Intelligence*. 4

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, Florence, Italy. Association for Computational Linguistics. 1, 2, 9

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019b. BERT rediscovers the classical NLP pipeline. In *ACL*, pages 4593–4601. Association for Computational Linguistics. 9

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 *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. 2, 3, 5

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*. 4

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. 3

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of Being Recurrent for Modeling Hierarchical Structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4731–4736. Association for Computational Linguistics. 9

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 2020, Online, November 16-20, 2020*, pages 183–196. Association for Computational Linguistics. 2, 6, 8, 9

Weischedel, Ralph, Palmer, Martha, Marcus, Mitchell, Hovy, Eduard, Pradhan, Sameer, Ramshaw, Lance, Xue, Nianwen, Taylor, Ann, Kaufman, Jeff, Franchini, Michelle, El-Bachouti, Mohammed, Belvin, Robert, and Houston, Ann. 2013. Ontonotes release 5.0. 3

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380. Association for Computational Linguistics. 1

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. 1, 2