On the Paradox of Learning to Reason from Data

Honghua Zhang, **Liunian Harold Li**, **Tao Meng**, **Kai-Wei Chang** and **Guy Van den Broeck**
University of California, Los Angeles
{hzhang19, liunian.harold.li, tmeng, kwchang, guyvdb}@cs.ucla.edu

Abstract

Logical reasoning is needed in a wide range of NLP tasks. Can a BERT model be trained end-to-end to solve logical reasoning problems presented in natural language? We attempt to answer this question in a *confined problem space* where there exists a set of parameters that *perfectly simulates* logical reasoning. We make observations that seem to contradict each other: BERT attains near-perfect accuracy on in-distribution test examples while failing to generalize to other data distributions over the exact *same* problem space. Our study provides an explanation for this paradox: instead of learning to emulate the correct reasoning function, BERT has, in fact, learned *statistical features* that inherently exist in logical reasoning problems. We also show that it is infeasible to jointly remove statistical features from data, illustrating the difficulty of learning to reason in general. Our result naturally extends to other neural models (e.g. T5) and unveils the fundamental difference between learning to reason and learning to achieve high performance on NLP benchmarks using statistical features.

1 Introduction

Logical reasoning is needed in a wide range of NLP tasks, including natural language inference (NLI) [Williams *et al.*, 2018; Bowman *et al.*, 2015], question answering (QA) [Rajpurkar *et al.*, 2016; Yang *et al.*, 2018] and common-sense reasoning [Zellers *et al.*, 2018; Talmor *et al.*, 2019]. The ability to draw conclusions based on given facts and rules is essential to solving these tasks.[1] Though NLP models, empowered by the Transformer neural architecture [Vaswani *et al.*, 2017], can achieve high performance on task-specific datasets, it is unclear whether they are "reasoning" following the rules of logic. A research question naturally arises: *can neural models be trained end-to-end to conduct logical reasoning in natural language?*

Following prior work, we attempt to answer this question by training and testing a neural model (e.g. BERT [Devlin *et al.*, 2019]) on a **confined problem space** (see Fig. 1 and Sec. 2) consisting of logical reasoning problems written in English [Johnson *et al.*, 2017; Sinha *et al.*, 2019]. Yet, we observe evidences that seemingly lead to a contradiction.

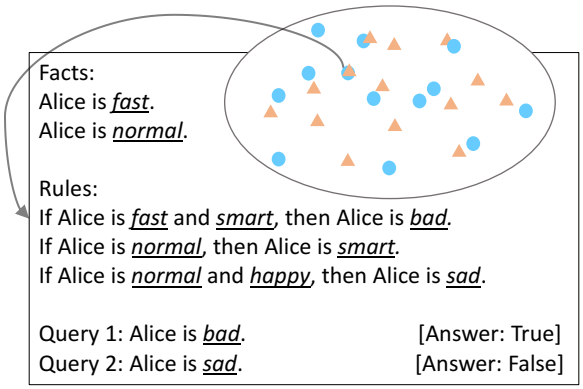


Figure 1: A confined problem space (SimpleLogic) consisting of exponentially many ($\approx 10^{360}$) logical reasoning problems; dots and triangles denote examples sampled from two different distributions over the same problem space.

On the one hand, echoing the findings of prior work [Clark *et al.*, 2020; Talmor *et al.*, 2020], we observe evidences that seem to imply that *neural models can learn to reason* (i.e. reliably emulate the correct reasoning function): (E1) examples in the problem space only test model's reasoning ability: they have *no* language variance and require *no* prior knowledge; (E2) we prove by construction that the BERT model has enough capacity to represent the correct reasoning function (Sec 2.2); (E3) the BERT model can be trained to achieve near-perfect test accuracy on data distributions covering the whole problem space.

On the other hand, we observe a contradictory phenomenon: the models attaining near-perfect accuracy on one data distribution do not generalize to other distributions within *the same problem space* (Sec. 3). Since the correct reasoning function does not change across data distributions, it follows that *the model has not learned to reason*.

The paradox lies in that if a neural model *has* learned reasoning, it should not exhibit such a generalization failure; if the model *has not* learned reasoning, it is baffling how it manages to achieve near-perfect test accuracy on training

[1] A.k.a., deductive reasoning; in this paper, we do not consider inductive reasoning, where rules need to be learned.

distributions covering the entire problem space. What we observed is not a common case of out-of-distribution (OOD) generalization failure: (1) our problem space is confined and simple (see E1, E2); (2) the correct reasoning function is invariant across data distributions; on the contrary, discussions about OOD generalization often involve open problem spaces [Lin *et al.*, 2019; Gontier *et al.*, 2020; Yin *et al.*, 2020; Wald *et al.*, 2021] and domain mismatch between training and testing distribution [Yin *et al.*, 2021; Koh *et al.*, 2021].

Upon further investigation, we provide an explanation for this paradox: the model attaining high accuracy **only** on in-distribution test examples **has not learned to reason**. In fact, the model has learned to use *statistical features* in logical reasoning problems to make predictions rather than to emulate the correct reasoning function.

Our first observation is that even the simplest statistic of a reasoning problem can give away significant information about the true label (Sec. 4.1): for example, by only looking at the number of rules in a reasoning problem, we can predict the correct label better than a random guess. This is different from the common artifacts/shortcuts identified in NLP datasets, which often arise when dataset collection/annotation are not representative of the real world (the ground-truth function) [Torralba and Efros, 2011; Gururangan *et al.*, 2018; Clark *et al.*, 2019; He *et al.*, 2019]. In our setting, even though (1) we have access to the ground-truth reasoning function and (2) the problem space is simple, finite and free of language variations, statistical features still **inherently** exist and are not specific to certain data distributions. We show that statistical features can hinder model generalization performance (Sec. 4.2); in addition, we argue that there are potentially countless statistical features and it is computationally expensive to jointly remove them from data (Sec. 4.3).

Our study implies the difficulty of learning to reason from data: while a model always tends to learn statistical features, it is difficult to construct a logical reasoning dataset with no statistical features. Though we use BERT as the running example throughout this paper, our argument assumes little about model architecture and naturally extends to other neural models. This intuition is supported by experiments with T5 [Raffel *et al.*, 2020], which exhibits similar behaviors.

Our findings unveil the fundamental difference between "learning to reason" and "learning to attain high performance on NLP benchmarks." Learning statistical features is not always undesirable; in fact, for most NLP tasks, one of the major goal for a neural model is to learn statistical patterns: for example, in sentiment analysis [Maas *et al.*, 2011], a model is *expected* to learn the strong correlation between the occurrence of the word "happy" and the positive sentiment. However, for logical reasoning, even though countless statistical features inherently exist, models should not use them to make predictions. Caution should be taken when we seek to train neural models end-to-end to solve logical reasoning tasks in NLP that involve prior knowledge and are presented with language variance [Welleck *et al.*, 2021; Yu *et al.*, 2020], which could potentially lead to even stronger statistical features, as demonstrated by [Elazar *et al.*, 2021] and [McCoy *et al.*, 2019].

2 SimpleLogic: A Simple Problem Space for Logical Reasoning

We define *SimpleLogic*, a class of logical reasoning problems based on propositional logic, as a controlled testbed for testing neural models' ability to conduct logical reasoning. SimpleLogic only contains deductive reasoning examples. To simplify the problem, we remove language variance by representing the reasoning problems in a templated language and limit their complexity (e.g., examples have limited input lengths and reasoning depths).

Solving SimpleLogic does not require significant model capacity. We show that the BERT model [Devlin *et al.*, 2019] has more than enough model capacity to solve SimpleLogic by constructing a parameterization of BERT that can solve all instances in SimpleLogic (Sec. 2.2).

2.1 Problem Space Definition

Before defining SimpleLogic, we introduce some basics for propositional logic. In general, reasoning in propositional logic is NP-complete [Cook, 1971]; hence, we only consider propositional reasoning with *definite clauses*. A definite clause is a *rule* of the form $A_1 \wedge A_2 \wedge \cdots \wedge A_n \rightarrow B$, where A_is and B are *atoms* that take values in "True" or "False"; we refer to the left hand side of a rule as its *body* and the right hand side as its *head*. A definite clause is called a *fact* if its body is empty (i.e. $n = 0$). A *propositional theory* T is a set of rules and facts, and we say a atom Q *can be proved* from T if either (1) Q is given in T as a fact or (2) $A_1 \wedge \cdots \wedge A_n \rightarrow Q$ is given in T as a rule where A_is can be proved.

Each example in SimpleLogic is a propositional reasoning problem that only involves definite clauses. In particular, each example is a tuple (*facts*, *rules*, *query*, *label*) where (1) *facts* is a list of atoms that are known to be True, (2) *rules* is a list of rules represented as definite clauses, (3) *query* is a single atom, and (4) *label* is either True or False, denoting whether the query atom can be proved from *facts* and *rules*. Figure 1 shows such an example. Furthermore, we enforce simple constraints to control the difficulty of the problems. For each example in SimpleLogic, we require that:

- the number of atoms (#atom) that appear in facts, rules and query ranges from 5 to 30, and all atoms are sampled from a fixed vocabulary containing 150 adjectives such as "happy" and "complicated"; note that the atoms in SimpleLogic have **no** semantics;
- $0 \leq$ the number of rules (#rule) $\leq 4\times$ #atom; the body of each rule contains 1 to 3 atoms; i.e. $A_1 \wedge \ldots \wedge A_n \rightarrow B$ with $n > 3$ is not allowed;
- $1 \leq$ the number of facts (#fact) $\leq$ #atom;
- the reasoning depth[2] required to solve an example ranges from 0 to 6.

We use a simple template to encode examples in SimpleLogic as English input. For example, we use "*Alice is X.*" to represent the fact that X is True; we use "*A and B, C.*" to represent

[2]For a query with label *True*, its reasoning depth is given by the depth of the shallowest proof tree; for a query with label *False*, its reasoning depth is the maximum depth of the shallowest failing branch in all *possible* proof trees.

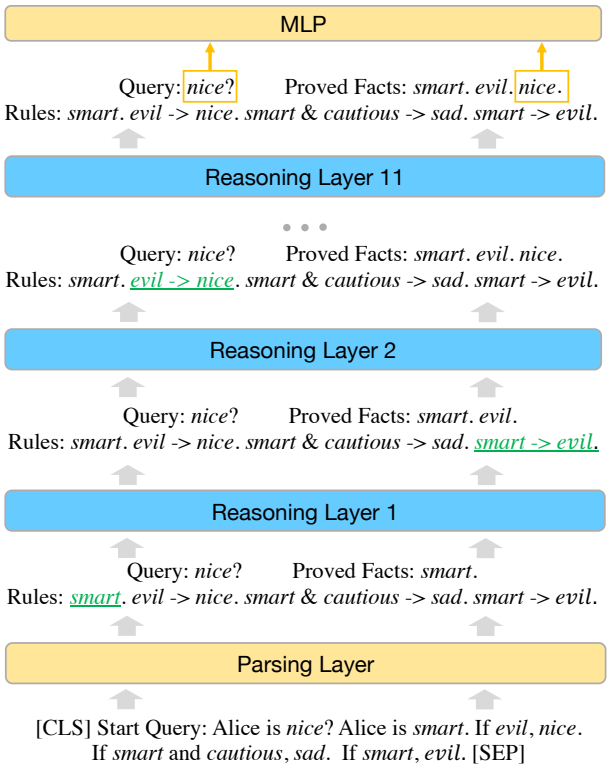


Figure 2: A BERT-base model that simulates the forward-chaining algorithm. The first layer parses text input into the desired format. Each reasoning layer performs one step of forward-chaining, adding some atoms to the Proved Facts, and the rules being used are underlined in green; e.g. Reasoning Layer 1 use the rule "smart $\rightarrow$ evil" to prove the atom *evil*.

the rule $A \wedge B \rightarrow C$; we use "*Query: Alice is Q.*" to represent the query atom Q. We concatenate *facts*, *rules* and *query* as "[CLS] *facts*. *rules* [SEP] *query* [SEP]" and supplement it to BERT to predict the correct label.

2.2 BERT Has Enough Capacity to Solve SimpleLogic

In the following, we show that BERT has enough capacity to solve all the examples in SimpleLogic. In particular, we explicitly construct a parameterization for BERT such that the fixed-parameter model solves all problem instances in SimlpleLogic. Note that we only prove the existence of such a parameterization, but do not discuss whether it can be learned from data until Sec. 3.

Theorem 1. *For a BERT model with n layers, there exists a set of parameters such that the model correctly solves any reasoning problem in SimpleLogic that requires $\leq n-2$ steps of reasoning.*

Proof Sketch. To prove this theorem, we construct a fixed set of parameters for BERT to simulate the forward-chaining algorithm. As illustrated in Figure 2, our construction solves a logical reasoning example in a layer-by-layer fashion. The 1st layer of BERT parses the input sequence into the desired format. Layer 2 to layer 10 are responsible for simulating the forward chaining algorithm: each layer performs one step of reasoning, updating the True/False label for atoms. The last layer also performs one step of reasoning, while implicitly checking if the query atom is proved and feeding the result to an MLP. The parameters are the same across all layers except for the Parsing Layer. See appendix for details. □

We implemented the construction in PyTorch, following the architecture of the BERT-base model. As supported by the theorem, the "constructed BERT" solves all the problems in SimpleLogic of reasoning depth ≤ 10 with 100% accuracy[3].

3 BERT Fails to Learn to Solve SimpleLogic

Next, we study whether it is possible to train a neural model (e.g., BERT) to reason on SimpleLogic. We follow [Clark *et al.*, 2020] to randomly sample examples from the problem space and train the BERT model on a large amount of sampled data.

3.1 Sampling Examples from SimpleLogic

When sampling examples from a finite domain, one naive approach is to uniformly sample from the domain. However, uniform sampling is *not desirable*: as described in Sec. 2.1, examples in SimpleLogic have #atom ranging from 5 to 30 and #rule ranging from 0 to 4 $\times$ #atom, as the number of combinations with #atom = 30 and #rule = 120 is significantly larger than other settings, if follows that over 99.99% of the examples generated by uniform sampling would have 30 atoms and 120 rules. This is a serious problem as we expect our training set to contain examples of different #atom, #fact and #rule. Hence, we instead consider the following two intuitive ways of sampling examples:

Rule-Priority (RP). In Rule-Priority, we first randomly sample #atom, #fact and #rule uniformly at random from $[5, 30]$, $[1, \#\text{atom}]$ and $[1, 4 \times \#\text{atom}]$ respectively, ensuring that all three aspects are covered by a non-trivial number of examples. Then, we randomly sample some atoms, facts and rules based on the given #atom, #rule and #fact. The query is also randomly sampled, and its label is computed by forward-chaining based on the given facts and rules.

Label-Priority (LP). In RP, we first randomly generate rules and facts, which then determines the label for each atom. In Label-Priority (LP), we consider generating examples in the "reversed" order: we first randomly assign a True/False label to each atom and then randomly sample rules and facts that are *consistent* with pre-assigned labels.

Figure 3 shows an example that illustrates the two sampling methods. Both LP and RP are general, and they cover the whole problem space. We refer readers to the Appendix for further details about the sampling algorithms.

3.2 BERT Trained on Randomly Sampled Data Cannot Generalize

Following the two sampling regimes described above, we randomly sample two sets of examples from SimpleLogic: for each reasoning depth from 0 to 6, we sample $40k$ examples from SimpleLogic via algorithm RP and aggregate

[3]https://github.com/joshuacnf/paradox-learning2reason

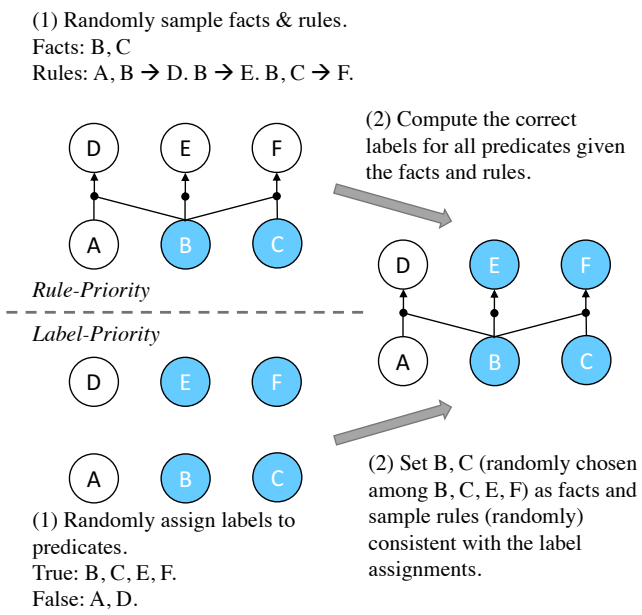


Figure 3: An illustration of a logical reasoning problem (right) in SimpleLogic being sampled by Rule-Priority (RP) and Label-Priority (LP), respectively. Atoms with label *True* are denoted by filled circles.

them as dataset RP, which contains $280k$ examples in total; we then split it as training/validation/test set. We use the same procedure to generate dataset RP. We train a BERT-base model [Devlin *et al.*, 2019] on RP and LP, respectively. See training details in appendix[4].

BERT Performs Well on Training Distributions

The first and last rows of Table 1 show the test accuracy when the test and train examples are sampled by the same algorithm (e.g., for row 1, the model is trained in the RP training set and tested in the RP test set): the models achieve near-perfect performance similar to the findings in prior work [Clark *et al.*, 2020]. Both sampling algorithms are general in the sense that every instance in SimpleLogic has a positive probability to be sampled; hence, the intuition is that the model has learned to emulate the correct reasoning function.

BERT Fails to Generalize

However, at the same time, we observe a rather counterintuitive finding: the test accuracy drops significantly when the train and test examples are sampled via different algorithms. Specifically, as shown in the second and third rows of Table 1, the BERT model trained on RP fails drastically on LP, and vice versa. Since the correct reasoning function does not change across different data distributions, this generalization failure indicates that BERT has not learned to conduct logical reasoning. A subsequent question naturally arises: *can the model learn to reason if we train it on both RP and LP?*

Training on Both RP and LP is Not Enough

We train BERT on the mixture of RP and LP, and BERT again achieves nearly perfect test accuracy. Can we now conclude

[4]https://arxiv.org/abs/2205.11502

Train	Test	0	1	2	3	4	5	6
RP	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
	LP	99.8	99.8	99.3	96.0	90.4	75.0	57.3
LP	RP	97.3	66.9	53.0	54.2	59.5	65.6	69.2
	LP	100.0	100.0	99.9	99.9	99.7	99.7	99.0

Table 1: Test accuracy on LP/RP for the BERT model trained on LP/RP; the accuracy is shown for test examples with reasoning depth from 0 to 6. BERT trained on RP achieves almost perfect accuracy on its test set; however the accuracy drops significantly when it's tested on LP (vice versa).

Test	0	1	2	3	4	5	6
RP&LP	99.9	99.9	99.8	99.4	98.8	98.1	95.6
LP*	98.1	97.2	92.5	80.3	65.8	55.6	55.2

Table 2: BERT trained on a mixture over RP and LP fails on LP*, a test set that slightly differs from LP.

that BERT has learned to approximate the correct reasoning function? We slightly tweak the sampling algorithm of LP by increasing the expected number of alternative proof trees to generate LP*. Unfortunately, we observe that the model performance again drops significantly on LP* (Table 2); such a result resembles what we observed in Table 1. In fact, we find *no evidence* that consistently enriching the training distribution will bring a transformative change such that the model can learn to reason.

Discussion

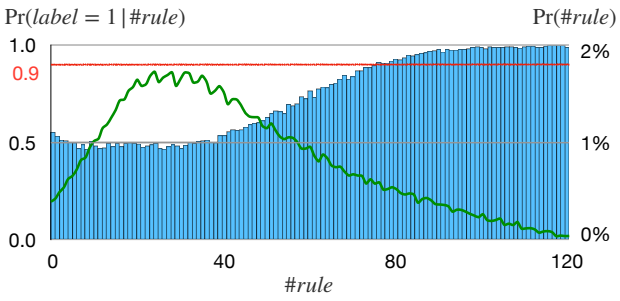
The experiments above reveal a pattern of generalization failure: *if we train the model on one data distribution, it fails almost inevitably on a different distribution.* In other words, the model seems to be emulating an incorrect "*reasoning function*" specific to its training distribution.

4 BERT Learns Statistical Features

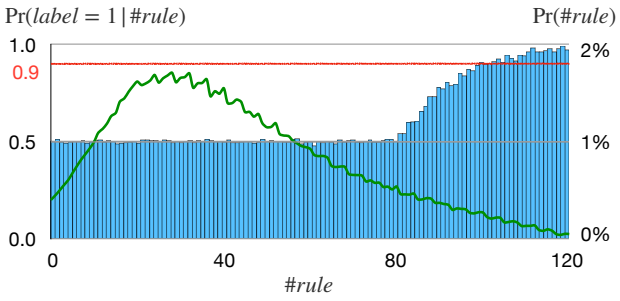
To this point, we have shown that a BERT model achieving high in-distribution accuracy does not learn the correct reasoning function. In this section, we seek to provide an explanation for this peculiar generalization failure. Our analysis suggests that even the simplest statistics of reasoning problems can provide significant information about their labels, which we denote as *statistical features*. Such statistical features are **inherent** to the task of logical reasoning rather than a problem with specific datasets. When BERT is trained on data with statistical features, it tends to make predictions based on such features rather than learning to emulate the correct reasoning function; thus, BERT fails to generalize to the whole problem space. However, unlike the shallow shortcuts found in other typical NLP tasks, such statistical features can be countless and extremely complicated, and thus very difficult to be removed from training data.

4.1 Statistical Features Inherently Exists

What is a statistical feature? If a certain statistic of examples has strong correlation with their labels, we call it a *statistical feature*. As an illustrating example, we consider the



(a) RP: $\Pr(\text{label} = 1 \mid \#\text{rule}) > 0.5$ for $\#\text{rule} > 40$.



(b) RP_balance: $\Pr(\text{label} = 1 \mid \#\text{rule}) \approx 0.5$ for $\#\text{rule} \leq 80$.

Figure 4: $\Pr(\text{label} = 1 | \#\text{rule})$ (the blue columns) and $\Pr(\#\text{rule})$ (the green curves) for RP and RP_balance, respectively. After removing #rule as a statistical feature (RP_balance), $\Pr(\text{label} = 1 \mid \#\text{rule})$ approaches 0.5 for $\#rule \leq 80$ while $\Pr(\#\text{rule})$ does not change.

number of rules in a reasoning problem (#rule). As shown in Figure 4a, the #rule for reasoning problems in RP exhibit a strong correlation with their labels: when #rule > 40, the number of positive examples exceeds 50% by large margins; formally, $\Pr_{e\sim\text{RP}}(\text{label(e)} = 1 \mid \#\text{rule}(e) = x) > 0.5$ for $x > 40$, which makes it possible for the model to guess the label of an example with relatively high accuracy by only using its #rule. Hence, we call #rule a statistical feature for the dataset RP.

Statistical Features are Inherent to Logical Reasoning

Continuing with our example, we show that #rule *inherently* exists as a statistical feature for logical reasoning problems in general; that is, it is not specific to the RP dataset. Consider the following property about logical entailment:

Property (Monotonicity of entailment)**.** *Let P be a logical formula and H a theory (i.e., a set of logical formulas) such that H entails P; then for any theory H' such that $H \subset H'$, P is also entailed by H'.*

It follows that, intuitively, given a fixed set of atoms and facts, any atom is more likely to be proved when more rules are given, that is, $\Pr(\text{label(e)} = 1 \mid \#\text{rule}(e) = x)$ should increase roughly monotonically as x increases. Since this intuition assumes nothing about data distributions, it follows that such statistical patterns should naturally exist in any dataset that is not adversarially constructed. In addition to RP, we also verify that both LP and the uniform distribution exhibit similar statistical patterns, which we refer readers to Appendix for further details.

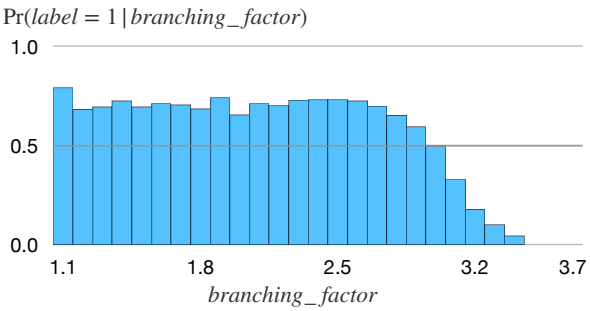


Figure 5: For RP, $\Pr(\text{label} = 1 \mid \text{branching\_factor})$ decreases as branching_factor increases.

Statistical Features are Countless

In addition to #rule, numerous statistical features potentially exist. For example, as facts can be seen as special form of rules, it follows from previous argument that #fact is also positively correlated with labels. Statistical features can be more complicated than just #rule or #fact. For example, the average number of atoms in rules of a reasoning problem can also leak information about its label. Note that the right-hand side of a rule is only proved if all atoms on its left-hand side are proved. Then, it is immediate that rules of the form $A, B, C \rightarrow D$ are less likely to be "activated" than rules of the form $A \rightarrow D$. Following this intuition, we can define the following statistic: for problem instance e, let

$$\text{branching\_factor}(e) := \frac{\#\text{fact}(e) + \sum_{\text{rule}\in e} \text{length of rule}}{\#\text{fact}(e) + \#\text{rule}(e)}.$$

In this definition, we compute the average number of atoms in the rules, where facts are treated as rules with one atom.[5] Our intuition suggests that the larger the branching_factor, the less likely an example will be positive; we verify that this intuition holds for RP, as shown in Figure 5. Just like #rule, we observe that branching_factor is also a statistical feature for both LP and the uniform distribution; see details in appendix.

Now we have shown that though there are simple statistical features like #rule, some (e.g. branching_factor) can be less intuitive to call to mind; in light of this, it is not hard to imagine that some statistical features can be so complex that they cannot even be manually constructed by humans. In particular, statistical features can also be *compositional*: one can define a *joint* statistical feature by combining multiple ones (e.g., branching_factor and #rule), which further adds to the complexity. Thus, it is infeasible to identify all of them.

4.2 Statistical Features Inhibit Model Generalization

Having verified that statistical features inherently exist for logical reasoning problems, in this section we study how they affect the model behavior. We show that (1) when statistical features are presented in training distributions, BERT tends

[5]Branching_factor: with more atoms on the left-hand side of the rules, the proof tree has more branches.

to utilize them to make predictions; (2) after removing **one** statistical feature from training data, the model generalizes better. It follows that statistical features can hinder the model from learning the correct reasoning function, explaining the generalization failure we observed in Section 3.

Example: Removing One Statistical Feature

We use #rule as an example to illustrate how to remove statistical features from a training dataset $\mathcal{D}$; in particular, there are three criteria that we need to satisfy: (1) label is balanced for the feature; (2) the marginal distribution of the feature remains unchanged; (3) the dataset size remains unchanged.

Formally, our first goal is to sample $\mathcal{D}' \subset \mathcal{D}$ such that, for all x:

$$\Pr_{e\sim\mathcal{D}'}(\text{label}(e) = 1 \mid \#\text{rule}(e) = x) = 0.5$$

Intuitively, this equation says that on $\mathcal{D}'$, one cannot do better than 50% by only looking at the #rule of an example. Specifically, for all possible values of x, if $\Pr_{e\sim\mathcal{D}}(\text{label}(e)=1 \mid \#\text{rule}(e)=x) > 0.5$, we drop some positive examples with #rule = x from $\mathcal{D}$; otherwise, we drop some negative examples.

However, we would not meet the second criterion by naively dropping the minimum number of examples; consider the following statistics for RP:

#rule	before drop #examples / positive %	after drop #examples / positive %
38	6860 / 49.9%	6822 / 50.0%
80	2322 / 92.7%	339 / 50.0%

As shown in the table, if we naively drop the minimum number of examples from RP such that Equation 1 is satisfied, we will be left with only 339 examples with #rule = 80, where the number (6822) of examples with #rule = 38 remains unchanged. This could be a serious issue in terms of dataset *coverage*: examples with some particular #rule will dominate $\mathcal{D}'$ and there will not be enough examples for other #rule. Recall that this is also the reason we choose RP/LP over uniform sampling to generate our datasets (Sec. 3.1). Hence, we also need to make sure that as we remove statistical features from $\mathcal{D}$, their marginal distributions in $\mathcal{D}'$ stay close to $\mathcal{D}$:

$$\Pr_{e\sim\mathcal{D}'}(\#\text{rule}(e)) = \Pr_{e\sim\mathcal{D}}(\#\text{rule}(e)).$$

In this way, $\mathcal{D}'$'s coverage of examples with different #rule remains the same as $\mathcal{D}$.

When both criteria (1) and (2) are satisfied, the size of $\mathcal{D}'$ will be *much smaller* than $\mathcal{D}$ and the ratio $k = |\mathcal{D}|/|\mathcal{D}'|$ can be estimated from $\min_x \Pr_{e\sim\mathcal{D}}(\text{label}(e)=1 \mid \#\text{rule}(e)=x)$. Hence, to make sure that criterion (3) is met, that is the size of $\mathcal{D}'$ is the same as $\mathcal{D}$, we need to pre-sample $k \times \mathcal{D}$ and obtain $\mathcal{D}'$ by down-sampling.

Following this approach, by down-sampling from $k \times$ RP, we construct RP_balance, where #rule is no longer a statistical feature. A rough estimation shows that if we were to balance $\Pr_{e\sim\text{RP}}(\text{label}(e) = 1 \mid \#\text{rule}(e) = x)$ for x up to 110, the ratio $k > 100$, that is, we need to spend over 100x running time (200 hours on a 40-core CPU) to pre-sample roughly 56 million examples; the computational cost would be even more expensive if we want to completely remove #rule as a statistical feature. Hence, we only balance this conditional probability for $0 \le x \le 80$, which takes 10x running time (20 hours on a 40-core CPU) to pre-sample 5.6 million examples. Not balancing the label for $x > 80$ is acceptable as 90% of the examples in RP have #rule ≤ 80. We train the BERT model on RP_balance, and the results are reported in Table 3.

Train	Test	0	1	2	3	4	5	6
RP_b	RP	99.8	99.7	99.7	99.4	98.5	98.1	97.0
	RP_b	99.4	99.6	99.2	98.7	97.8	96.1	94.4
	LP	99.6	99.6	99.6	97.6	93.1	81.3	68.1
RP	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
	RP_b	99.0	99.3	98.5	97.5	96.7	93.5	88.3
	LP	99.8	99.8	99.3	96.0	90.4	75.0	57.3

Table 3: The BERT model trained on RP performs worse on RP_balance (RP_b), indicating that the model uses #rule as a statistical feature to make predictions.

BERT Uses Statistical Features

As shown in Table 3, BERT trained on RP shows large performance drop when tested on RP_balance, while BERT trained on RP_balance shows even better performance on RP than RP-trained BERT. Since RP_balance is down-sampled from RP, the accuracy drop from RP to RP_balance is explained by that BERT trained on RP is using #rule to make predictions.

Removing Statistical Features Helps Generalization

As shown in Table 3, compared to RP-trained BERT, BERT trained on RP_balance achieves higher accuracy when tested on LP; in particular, for examples with reasoning depth 6, the model trained on RP_balance attains an accuracy of 68.1%, approximately 10% higher than the model trained on RP. This is a clear signal that when #rule is removed as a statistical feature, the model generalizes better, suggesting that statistical features can hinder the generalization of the model.

Statistical Features Explain the Paradox

Now we have a good explanation for the paradox: on the first hand, as we have discussed in Section 4.1, statistical features can be arbitrarily complex, thus powerful neural models can learn to use them to achieve high in-distribution accuracy; on the other hand, since the correlations between statistical features and labels can change as the data distribution changes, the model that relies on statistical features to make predictions does not generalize to out-of-distribution examples.

More importantly, as our argument assumes little about model architectures/pre-training procedures, most of our conclusions should also hold for other neural models. This hypothesis is supported by experiments with T5 [Raffel *et al.*, 2020], which exhibits behaviors similar to BERT: (1) the T5 model attaining near-perfect accuracy on the training distribution fails catastrophically on the other distributions; (2) the T5 model generalizes better after #rule is removed from RP, suggesting that it is using #rule to make predictions. See appendix for more details.

X	$\Pr(\text{label} = 1 \mid \text{X})$	k×
f = 15	0.908	5.5
f = 15, b ∈ [2.65,2.75]	0.975	20.0
f = 15, b ∈ [2.65,2.75], r = 58	0.991	55.6

Table 4: Jointly removing statistical features is difficult; e.g. second row shows: we need to sample *at least* 20 × RP to balance $\Pr(\text{label} = 1 | f = 15, b \in [2.65, 2.75])$.

4.3 On the Dilemma of Removing Statistical Features

We show that though removing one statistical feature (e.g., #rule) from training data can benefit model generalization, it is computationally infeasible to jointly remove multiple statistical features.

In the previous section, when we were trying to remove the #rule from RP, we could only afford to remove it for 90% of the examples. The general idea is that if a statistical feature X has a very strong correlation with the label on some dataset $\mathcal{D}$, i.e. $\Pr_{e\sim\mathcal{D}}(\text{label}(e) = 1 \mid X(e) = x)$ is very close to 1 or 0, then we would need to sample a lot of examples to have a balanced set.

The combination of multiple statistical features can give stronger signal about the label than the individual ones; thus it is even harder to jointly remove them. For example, we consider removing three statistical features from RP: #fact (f), branching_factor (b) and #rule (r). As shown in Table 4, as we try to jointly remove more statistical features X, $\Pr(\text{label} = 1|X)$ becomes more unbalanced; in particular, as we progressively remove #fact, branching_factor and #rule, the minimum times of examples we need to sample grows roughly exponentially: $5.5 \rightarrow 20.0 \rightarrow 55.6$. Note that we are only considering balancing *one* setting (#fact = 15, braching_factor ∈ [2.65,2.75], #rule = 58); for some other settings, the conditional probability can be more unbalanced, requiring us to pre-sample even more examples.

5 Related Work

Prior work contextualizes the problem of logical reasoning by proposing reasoning-dependent datasets and studies solving the tasks with neural models [Johnson *et al.*, 2017; Sinha *et al.*, 2019; Yu *et al.*, 2020; Liu *et al.*, 2020; Tian *et al.*, 2021]. However, most studies focus on solving a single task, and the datasets are either designed for a specific domain [Johnson *et al.*, 2017; Sinha *et al.*, 2019], or have confounding factors such as language variance [Yu *et al.*, 2020]; they cannot be used to strictly or comprehensively study the logical reasoning abilities of models.

Another line of research focuses on leveraging deep neural models to solve logical problems. For example, SAT solving [Selsam *et al.*, 2019], maxSAT [Wang *et al.*, 2019], temporal logical problems [Hahn *et al.*, 2021], DNF counting [Crouse *et al.*, 2019], learning embeddings for logical formula [Abdelaziz *et al.*, 2020; Crouse *et al.*, 2019] and mathematical problems [Saxton *et al.*, 2019; Lample and Charton, 2020]. In this work, we focus only on deductive reasoning, which is a general and fundamental class of reasoning problems. [Xu *et al.*, 2019] develops a theoretical framework to characterize how well neural models can generalize on different reasoning tasks. Prior to the study of reasoning in the context of deep learning, [Darwiche and Marquis, 2002] and [Khardon and Roth, 1997] studies the tractability of reasoning and learning to reason with propositional logic.

6 Conclusion

In this work, we study whether language models can learn to conduct logical reasoning by end-to-end training. We report and provide explanation to a seemingly contradictory phenomenon: while models can attain near-perfect test accuracy on training distributions, they fail catastrophically on other distributions; we demonstrate that they have learned to exploit statistical features rather than to emulate the correct reasoning function. Our study suggests that training on datasets might not be sufficient for model to learn certain complex behaviors such as reasoning and planning.

Acknowledgments

This work was funded in part by the DARPA Perceptually-enabled Task Guidance (PTG) Program under contract number HR00112220005, a DARPA MCS grant, NSF grants #IIS-1943641, #IIS-1956441, #CCF-1837129, Samsung, CISCO, Amazon Fellowship, and a Sloan Fellowship.

References

[Abdelaziz *et al.*, 2020] Ibrahim Abdelaziz, Veronika Thost, Maxwell Crouse, and Achille Fokoue. An experimental study of formula embeddings for automated theorem proving in first-order logic. *CoRR*, abs/2002.00423, 2020.

[Bowman *et al.*, 2015] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*. The Association for Computational Linguistics, 2015.

[Clark *et al.*, 2019] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *EMNLP/IJCNLP (1)*, pages 4067–4080. Association for Computational Linguistics, 2019.

[Clark *et al.*, 2020] Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *IJCAI*. ijcai.org, 2020.

[Cook, 1971] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158. ACM, 1971.

[Crouse *et al.*, 2019] Maxwell Crouse, Ibrahim Abdelaziz, Cristina Cornelio, Veronika Thost, Lingfei Wu, Kenneth D. Forbus, and Achille Fokoue. Improving graph neural network representations of logical formulae with subgraph pooling. *CoRR*, abs/1911.06904, 2019.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 2019.

[Elazar *et al.*, 2021] Yanai Elazar, Hongming Zhang, Yoav Goldberg, and Dan Roth. Back to square one: Artifact detection, training and commonsense disentanglement in the winograd schema. *arXiv preprint arXiv:2104.08161*, 2021.

[Gontier *et al.*, 2020] Nicolas Gontier, Koustuv Sinha, Siva Reddy, and Chris Pal. Measuring systematic generalization in neural proof generation with transformers. *Advances in Neural Information Processing Systems*, 33:22231–22242, 2020.

[Gururangan *et al.*, 2018] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *NAACL-HLT (2)*. Association for Computational Linguistics, 2018.

[Hahn *et al.*, 2021] Christopher Hahn, Frederik Schmitt, Jens U. Kreber, Markus Norman Rabe, and Bernd Finkbeiner. Teaching temporal logics to neural networks. In *ICLR*. OpenReview.net, 2021.

[He *et al.*, 2019] He He, Sheng Zha, and Haohan Wang. Unlearn dataset bias in natural language inference by fitting the residual. In *DeepLo@EMNLP-IJCNLP*, pages 132–142. Association for Computational Linguistics, 2019.

[Johnson *et al.*, 2017] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*. IEEE Computer Society, 2017.

[Khardon and Roth, 1997] Roni Khardon and Dan Roth. Learning to reason. *Journal of the ACM (JACM)*, 44(5):697–725, 1997.

[Koh *et al.*, 2021] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[Lample and Charton, 2020] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *ICLR*. OpenReview.net, 2020.

[Lin *et al.*, 2019] Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. Reasoning over paragraph effects in situations. *arXiv preprint arXiv:1908.05852*, 2019.

[Liu *et al.*, 2020] Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *IJCAI*. ijcai.org, 2020.

[Maas *et al.*, 2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[McCoy *et al.*, 2019] Tom McCoy, Ellie Pavlick, and Tal Linzen. 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, 2019.

[Raffel *et al.*, 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

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

[Saxton *et al.*, 2019] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *ICLR (Poster)*. OpenReview.net, 2019.

[Selsam *et al.*, 2019] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In *ICLR (Poster)*. OpenReview.net, 2019.

[Sinha *et al.*, 2019] Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *EMNLP/IJCNLP (1)*, pages 4505–4514. Association for Computational Linguistics, 2019.

[Talmor *et al.*, 2019] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL-HLT (1)*, pages 4149–4158. Association for Computational Linguistics, 2019.

[Talmor *et al.*, 2020] Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. *Advances in Neural Information Processing Systems*, 33:20227–20237, 2020.

[Tian *et al.*, 2021] Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. Diagnosing the first-order logical reasoning ability through logicnli. In *EMNLP (1)*. Association for Computational Linguistics, 2021.

[Torralba and Efros, 2011] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Wald *et al.*, 2021] Yoav Wald, Amir Feder, Daniel Greenfeld, and Uri Shalit. On calibration and out-of-domain generalization. In *NeurIPS*, pages 2215–2227, 2021.

[Wang *et al.*, 2019] Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *ICML*, Proceedings of Machine Learning Research. PMLR, 2019.

[Welleck *et al.*, 2021] Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. Naturalproofs: Mathematical theorem proving in natural language. *arXiv preprint arXiv:2104.01112*, 2021.

[Williams *et al.*, 2018] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*. Association for Computational Linguistics, 2018.

[Xu *et al.*, 2019] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.

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

[Yin *et al.*, 2020] Fan Yin, Quanyu Long, Tao Meng, and Kai-Wei Chang. On the robustness of language encoders against grammatical errors. In *ACL*, pages 3386–3403. Association for Computational Linguistics, 2020.

[Yin *et al.*, 2021] Da Yin, Liunian Harold Li, Ziniu Hu, Nanyun Peng, and Kai-Wei Chang. Broaden the vision: Geo-diverse visual commonsense reasoning. In *EMNLP (1)*, pages 2115–2129. Association for Computational Linguistics, 2021.

[Yu *et al.*, 2020] Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations (ICLR)*, April 2020.

[Zellers *et al.*, 2018] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP*. Association for Computational Linguistics, 2018.