On the Paradox of Learning to Reason from Data

Fall 2023, NLP Reading Group, University of Arizona

https://arxiv.org/pdf/2205.11502.pdf

Sushma Akoju
Advisor: Prof. Mihai Surdeanu

NLP reading Group Fall 2023: On the Paradox of Learning to Reason from Data

ICML: Knowledge and Logical Reasoning in the Era of Data-driven Learning

https://icml.cc/virtual/2023/workshop/21498

I attended ICML this Summer just for this specific workshop.

I added slide 5 for the discussion from the workshop about this paper.

Main idea

Idea is to test over simple reasoning examples to set the Transformers to succeed

- Tested over dataset created from propositional reasoning (np-complete).
- The model attains high accuracy only on in-distribution test examples.
- Learns to use statistical features
- Fails to emulate correct reasoning function

Contd..

Main Idea...

1. The rules of logic never rely on statistical patterns to conduct reasoning
2. Models inherently learn statistical features
3. Example from ICML, workshop that was discussed:
4. “The Weather is ……” and a constraint contains “winter”

p(next-token|prefix) = [cold: 0.05, warm: 0.10]

Intractable vs Tractable : How often the next token has winter in it and what are such possible next tokens

Present models use some model q(.|constraint) :

- amortized inference, encoding, masked, seq2seq, prompt tuning
- Learns statistical features that inherently exist in reasoning examples.
- Because constraint = winter, " The weather is..."
- p(next-token|prefix) = [cold: 0.05, warm: 0.10]
- q(next-token|prefix, α), α = winter

For example, (approximated) "The weather is... **[cold, warm, winter, in winter season, like winter, fall, autumn, windy..]" - Not tractable in transformers?**

Contd...

But what we want needs to be made into Tractable:

- the rules of logic never rely on statistical patterns to conduct reasoning
- Possible solution: Marginalization
- Posterior on next token- somehow look at all possible future texts, sum over all things possible, count how often prefix and next token contains winter in it.
- "Tractable Control for Autoregressive Language Generation"
- HMMs

Evidences that seem to imply following:

E1: Logical reasoning problems in the problem space are self-contained: they have no language variance and require no prior knowledge.

E2: We show that theoretically, 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 a data distribution covering the whole problem space.

Verifying Contradictory Phenomena

- Models attaining near-perfect accuracy on data in-distribution

  do not generalize to other distributions within the same problem space.

- correct reasoning function does not change across data distributions
- it follows that the model has not learned to reason
- Evidence of learning statistical features in reasoning problems

Example Data

Problem configuration:

With circles and triangles is

The Confined problem space

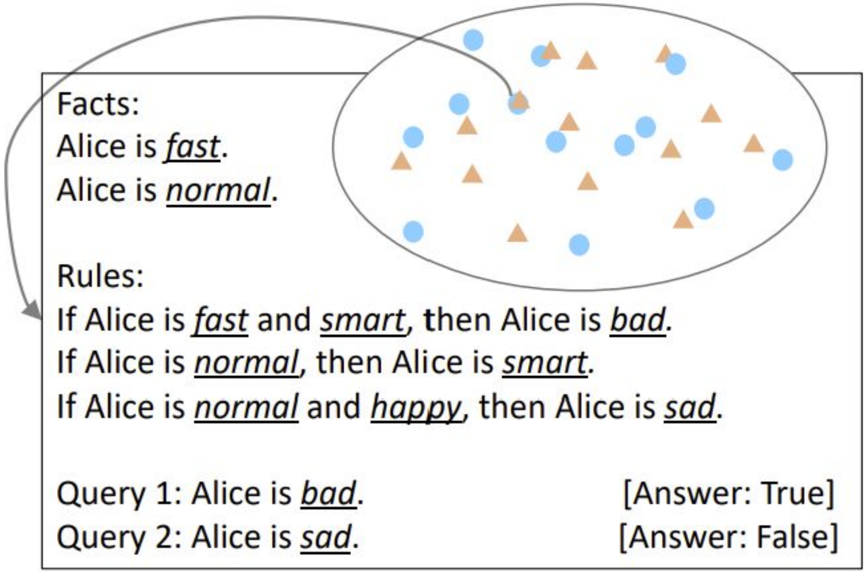


Figure 1: Problem setting: a confined problem space consisting of logical reasoning problems; dots and triangles denote examples sampled from different distributions over the same problem space.

Propositional reasoning with definite clauses

rule of the form $A1 \land A2 \land A3 \land \ldots \land An \rightarrow B$

(As and B take true or false)

LHS -> Body

RHS -> Head

Propositional Logic : If P and Q, then R

$$A1 \land A2 \land A3 \land \ldots \land An \rightarrow B$$

Facts: Body is empty (n = 0)

Propositional theory T :

Predicate Q can be proved from T if either

1. Q is given in T as a fact
2. $A1 \land \cdots \land An \rightarrow Q$ is given in T as a rule (Each of Ai can be proved)

Examples Generation

1. Facts, rules, query, label
2. Facts: list of predicates that are known to be True
3. Rules : list of rules represented as definite clauses
4. Query is a single predicate
5. label is either True or False, query pred can be proved true or false from facts

Predicates are adjectives & Bounded

- Bounded Vocabulary: 150 adjectives
- Bounded reasoning depth (depth <=6)
- Bounded problem space 10^360

Adjectives:

- Happy, elegant, witty, confident, inquisitive...
- Predicates in SimpleLogic have no semantics

- # of rules : 0 to #pred
- For each rule, body has n<=3 ($A1 \land A2 \land A3 \land \ldots \land An \rightarrow B$)
- Bounded Facts: 1 to #pred
- Reasoning depth: <= 6

About encoding examples

We use a simple template to encode examples in SimpleLogic as natural language input. For example, we use "*Alice is X.*" to represent the fact that X is True; we use "*A and B, C.*" to represent the rule $A \wedge B \rightarrow C$; we use "*Query: Alice is Q.*" to represent the query predicate Q. Then we concatenate *facts*, *rules* and *query* as *[CLS] facts. rules*

facts, rules and query as [CLS] facts.

rules [SEP] query [SEP]

Data in-distribution

1) RP: Randomly sample, predicates, facts, rules and Label using forward chaining

2) LP: Randomly assign True/False label to predicate and randomly sample rules & facts

+ consistent with pre-assigned labels

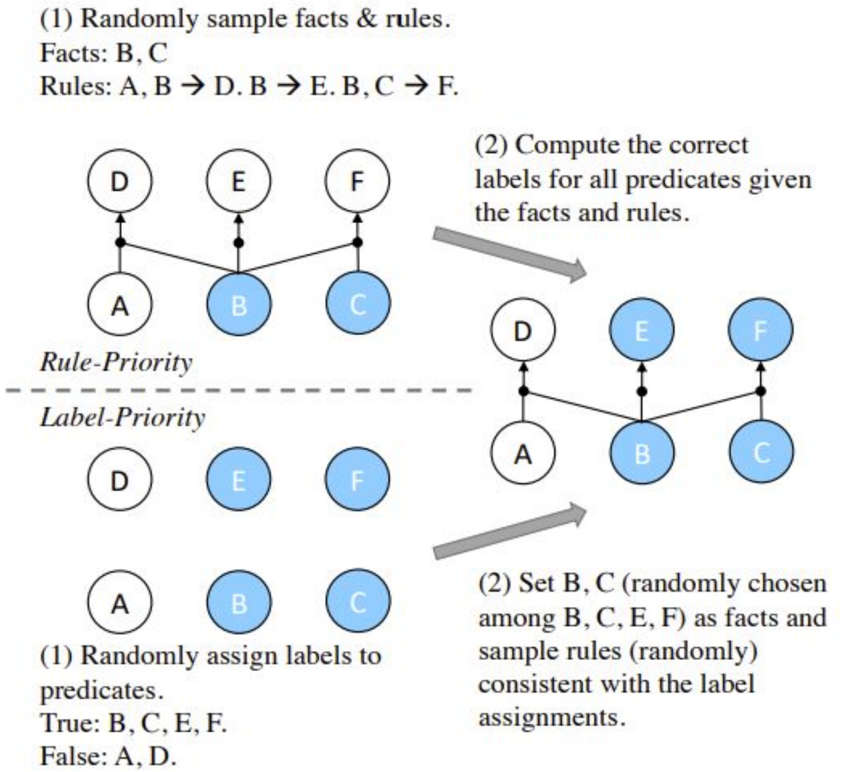


Figure 3: An illustration of a logical reasoning problem (right) in SimpleLogic being sampled by Rule-Priority (RP) and Label-Priority (LP), respectively. Predicates with label *True* are denoted by filled circles.

B Sampling Examples from SimpleLogic

B.1 Algorithms: Rule-Priority & Label-Priority

a *Rule-Priority* (RP)

$pred\_num \sim U[5, 30]$
$preds \leftarrow Sample(vocab, pred\_num)$
$fact\_num \sim U[1, pred\_num]$
$rule\_num \sim U[0, 4 * pred\_num]$
$rules \leftarrow$ empty array size of $rules < rule\_num$
$body\_num \sim U[1, 3]$
$body \leftarrow Sample(preds, body\_num)$
$head \leftarrow Sample(preds, 1)$ $tail \notin body$
add $body \rightarrow head$ to $rules$
$fact\_num \sim U[0, pred\_num]$
$facts \leftarrow Sample(preds, fact\_num)$
$query \leftarrow Sample(preds, 1)$
Compute $label$ via forward-chaining.
$(facts, rules, query, label)$

b *Label-Priority* (LP)

$pred\_num \sim U[5, 30]$
$preds \leftarrow Sample(vocab, pred\_num)$
$rule\_num \sim U[0, 4 * pred\_num]$
set $l \sim U[1, pred\_num/2]$ and group $preds$
into l layers predicate p in layer $1 \leq i \leq l$
$q \sim U[0, 1]$
assign label q to predicate p $i > 1$
$k \sim U[1, 3]$
$cand \leftarrow$ nodes in layer $i - 1$
with label $= q$
$body \leftarrow Sample(cand, k)$
add $body \rightarrow p$ to $rules$ size of $rules < rule\_num$
$body\_num \sim U[1, 3]$
$body \leftarrow Sample(preds, body\_num)$
$head \leftarrow Sample(preds, 1)$
add $body \rightarrow tail$ to $rules$ unless $tail$ has label 0 and
all predicates in $body$ has label 1.
$facts \leftarrow$ predicates in layer 1 with label $= 1$
$query \leftarrow Sample(preds, 1)$
$label \leftarrow$ pre-assigned label for $query$
$(facts, rules, query, label)$

Figure 9: Two sampling algorithms Rule-Priority and Label-Priority. $Sample(X, k)$ returns a random subset from X of size k. $U[X, Y]$ denotes the uniform distribution over the integers between X and Y.

RP vs LP

RP: Uniformly at random

sampling of rules/facts

Vs

LP: Random Sampling

of rules/facts

Consistent over assigned labels

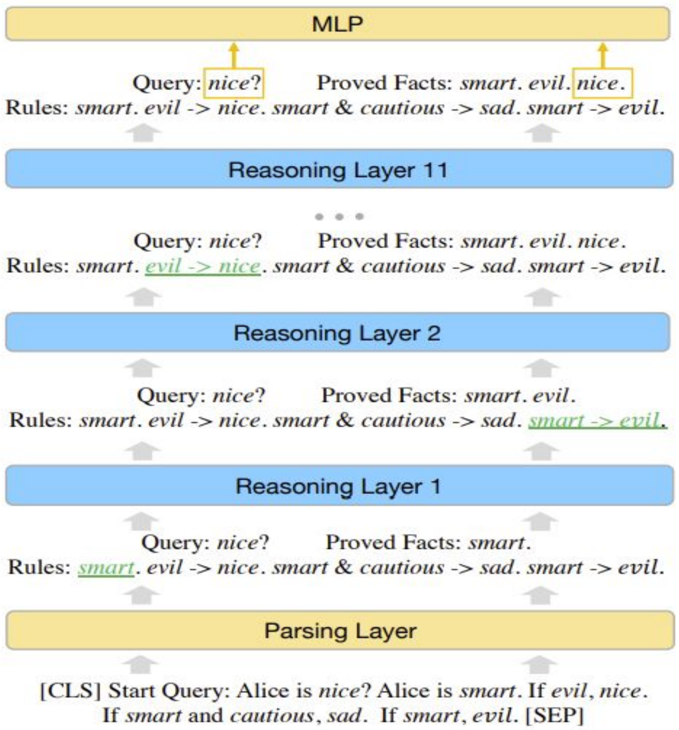


Figure 2: A BERT-base model that simulates the forward-chaining algorithm. The first layer parses text input into the desired format. Each reasoning layer performs one step of forward-chaining, adding some predicates to the Proved Facts, and the rules being used are underlined in green; e.g. Reasoning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

Evaluation on LP vs RP trained on in-distribution data

- Same vocabulary
- Confined problem space
- But assign labels after

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).

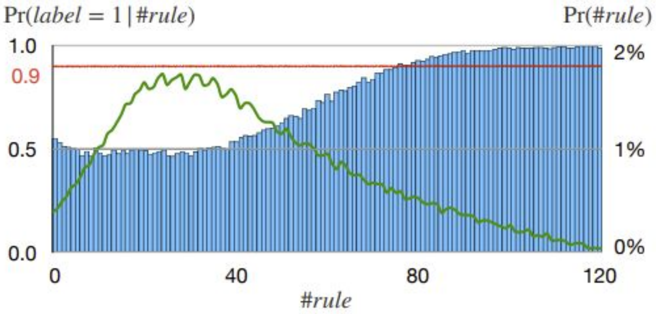
Statistical features are inherent to logical reasoning problems: Monotonicity of entailment

Property (Monotonicity of entailment). Any additional facts and rules can be freely added to the hypothesis of any proven fact.

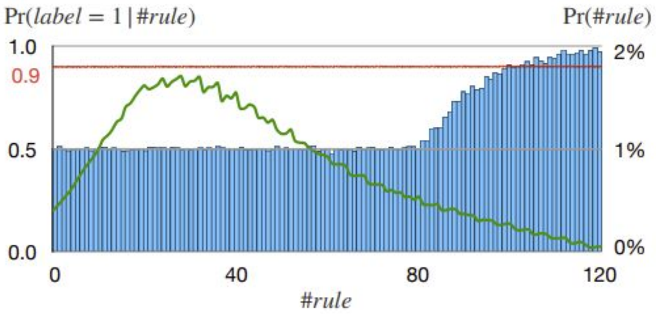
Pr(label(e) = 1 | #rule(e) = x) should increase (roughly) monotonically as x increases

Statistical features are countless.

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



(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: $\Pr(\text{label} = 1 \mid \#\text{rule})$ (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.

#rules highly correlated with labels. #fact is also positively correlated with labels.

Average number of predicates in rules can leak information.

Branching factor: A,B,C -> D less likely activated than A ->D

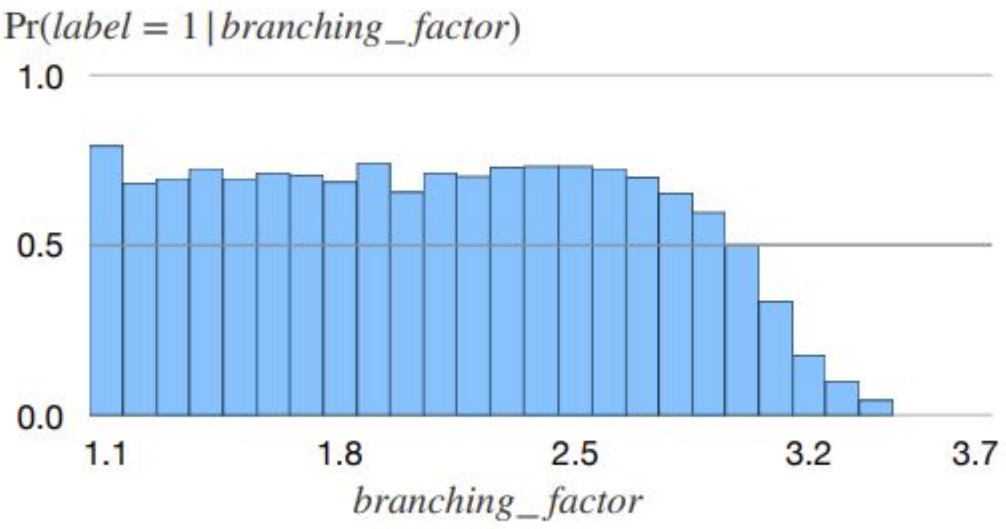
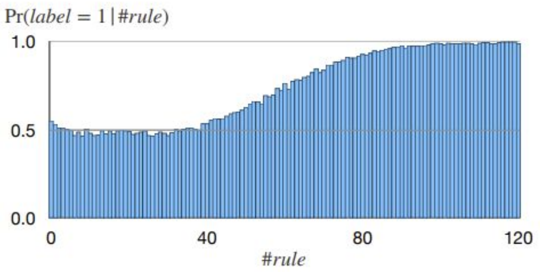


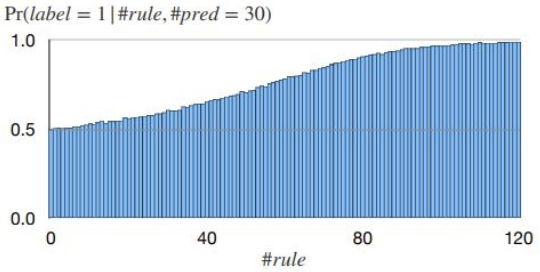
Figure 5: For RP, Pr(label = 1 | branching_factor) decreases as branching_factor increases.

A Statistical Features in Different Data Distributions



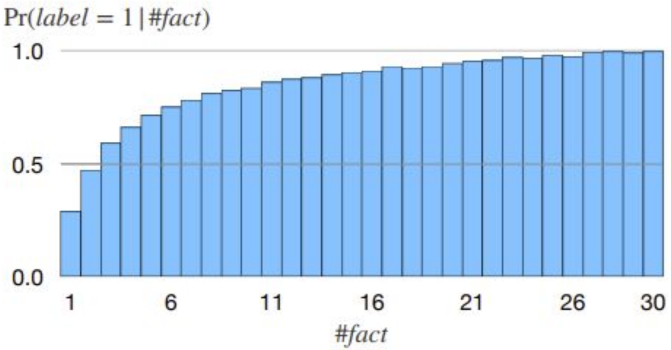
(a) Statistics for examples generated by Rule-Priority (RP).

(b) Statistics for examples generated by Label-Priority (LP).

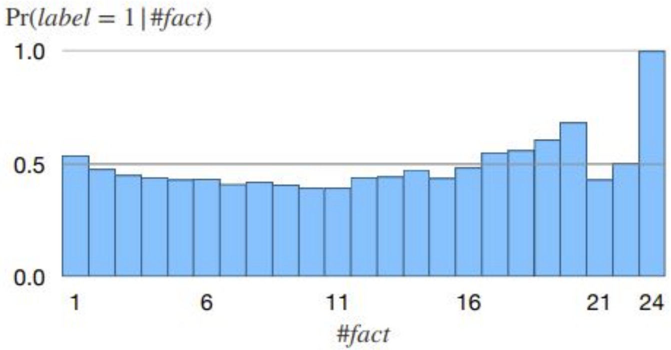


(c) Statistics for examples generated by uniform sampling; we only consider examples with #pred = 30 as a good-enough approximation: over 99% of the examples generated by uniform sampling have #pred = 30.

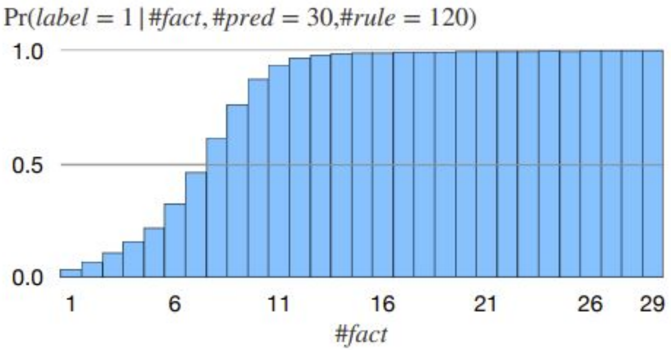
Figure 6: #rule is a statistical feature for RP, LP and the uniform distribution. Even though $\Pr(\text{label} = 1|\#\text{rule})$ increases as #rule increases for all three distributions, it follows a slightly different pattern for each distribution; that is to say, the correlation between #rule and the label changes as the underlying data distribution changes, which explains the generalization failure we observed.



(a) Statistics for examples generated by Rule-Priority (RP).

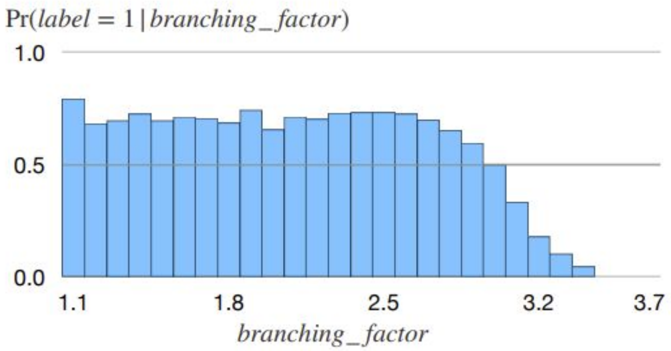


(b) Statistics for examples generated by Label-Priority (LP).

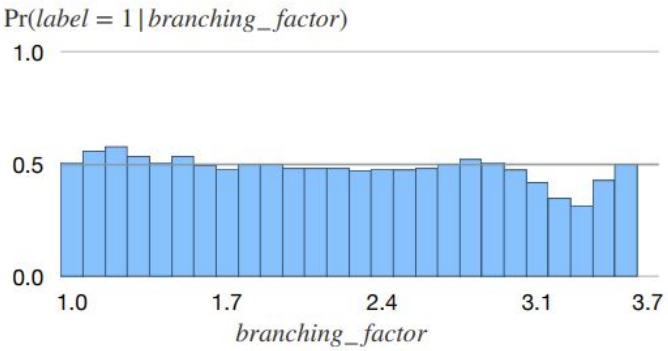


(c) Statistics for examples generated by uniform sampling; we only consider examples with #pred = 30 and #rule = 120 as a good-enough approximation: over 99% of the examples generated by uniform sampling have #pred = 30 and #rule = 120.

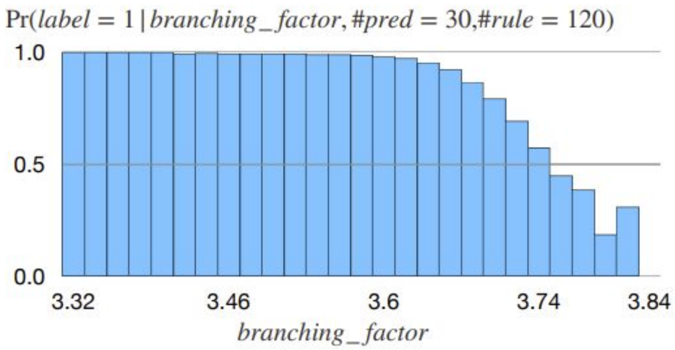
Figure 7: #fact is a statistical feature for RP, LP and the uniform distribution.



(a) Statistics for examples generated by Rule-Priority (RP).



(b) Statistics for examples generated by Label-Priority (LP).



(c) Statistics for examples generated by uniform sampling; we only consider examples with #pred = 30 and #rule = 120 as a good-enough approximation: over 99% of the examples generated by uniform sampling have #pred = 30 and #rule = 120.

Figure 8: branching_factor is a statistical feature for RP, LP and the uniform distribution.

BERT uses statistical features to make predictions

Statistical features explain the paradox.

On the Dilemma of Removing Statistical Features:

X	Pr(label = 1 | 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(label = 1 | f = 15, b ∈ [2.65, 2.75]).

Strategy to verify if Statistical Features Inhibit Model Generalization

- Use RP_balance: downsample k * RP so #rule is no longer a feature
- To verify if
  - Statistical Features Inhibit Model
  - Model generalizes better after removing a feature

D0 ⊂ D such that, for all x: $\Pr_{e \sim \mathcal{D}'}(\text{label}(e) = 1 \mid \#\text{rule}(e) = x) = 0.5$

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

Strategy to remove a statistical feature

1. label is balanced for the feature
2. the marginal distribution of the feature remains unchanged
3. the dataset size remains unchanged.

statistical features can also be compositional

it is infeasible to identify all statistical features.

Balanced RP

Train	Test	0	1	2	3	4	5	6
	RP	99.8	99.7	99.7	99.4	98.5	98.1	97.0
RP_b	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	99.9	99.8	99.7	99.3	98.3	97.5	95.5
RP	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 model trained on RP performs worse on RP_balance (RP_b). This indicates that the model is using the statistical feature #rule to make predictions.

Theorem

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