Logically Consistent Adversarial Attacks for Soft Theorem Provers

Alexander Gaskell[1,2] , **Yishu Miao**[1,2] , **Francesca Toni**[1] and **Lucia Specia**[1]
[1]Imperial College London
[2]ByteDance
{alexander.gaskell19, f.toni, l.specia}@imperial.ac.uk, yishu.miao@gmail.com

Abstract

Recent efforts within the AI community have yielded impressive results towards "soft theorem proving" over natural language sentences using language models. We propose a novel, generative adversarial framework for probing and improving these models' reasoning capabilities. Adversarial attacks in this domain suffer from the *logical inconsistency* problem, whereby perturbations to the input may alter the label. Our **L**ogically consistent **A**d**V**ersarial **A**ttacker, **LAVA**, addresses this by combining a structured generative process with a symbolic solver, guaranteeing logical consistency. Our framework successfully generates adversarial attacks and identifies global weaknesses common across multiple target models. Our analyses reveal naive heuristics and vulnerabilities in these models' reasoning capabilities, exposing an incomplete grasp of logical deduction under logic programs. Finally, in addition to effective probing of these models, we show that training on the generated samples improves the target model's performance.

1 Introduction

Recent research highlights that language models are surprisingly adept at memorising and recalling factual information when trained on internet-scale corpora [Petroni *et al.*, 2019]. Less clear, though, is their ability to deduce new conclusions by manipulating and combining this stored knowledge. The NLP community explore this via the publication of natural language benchmarks requiring logical reasoning [Yu *et al.*, 2020]. While neural models are increasingly improving on these tasks, their reasoning capabilities can prove brittle and shallow upon closer examination [Helwe *et al.*, 2021].

Existing studies seemingly demonstrate that language models can act as "soft theorem-provers" (STPs) over natural language sentences. [Clark *et al.*, 2020] demonstrate that Transformers [Vaswani *et al.*, 2017] can learn to predict whether a natural language query is logically entailed by a collection of natural language facts and rules, as illustrated in Fig. 1. Their results appear impressive, showing near-perfect performance on the original task, demonstrating robustness to syntactic and lexical variations and generalization beyond the training set. Despite this, an important question remains: have these systems learned the desired reasoning processes, or are they dependent on shallow heuristics and shortcuts?

Facts: John is smart. Alan is small. Mark is smart. Sarah is tall.
Rules: Tall people are nice. If Mark is smart then John is tall. Small people are friendly. Anyone who is not happy is smart.

Queries [T/F]:	**Answers:**
Q1. John is nice.	T
Q2. Sarah is not tall.	F
Q3. Sarah is smart.	T

Figure 1: A (simplified) illustration of the *RuleTakers* dataset [Clark *et al.*, 2020]. The task is to predict whether a query is entailed by a set of facts and rules.

Adversarial machine learning [Szegedy *et al.*, 2014] offers a natural inroad into this question. It is widely used within NLP to probe model robustness and defend against adversarial attacks [Ebrahimi *et al.*, 2018; Li *et al.*, 2020]. A successful attack minimally perturbs an input such that: 1) its semantics is preserved, and 2) the model alters its prediction. The first challenge is usually addressed by assuming that "small" perturbations will not meaningfully impact semantics. However, previous works focus on lexical semantics while we are interested in logical semantics. The assumption that "small" perturbations will not impact logical semantics is unduly strong as logical entailment is sensitive to any perturbations, causing standard methods to generate *inconsistent* attacks by inadvertently flipping the label. Considering Fig. 1, existing methods may perturb *"Mark is smart"* to *"Mark is intelligent"*, but this attack is inconsistent for Q1 as it is no longer entailed after the perturbation. To solve this *logical inconsistency* problem, our contributions are as follows:

1. We propose *logically consistent attacks*, whereby the perturbed sample's label faithfully reflects its new entailment, and show that standard methods produce logically inconsistent attacks.
2. We propose LAVA (**L**ogically consistent **A**d**V**ersarial **A**ttacker): a black-box, generative adversarial framework to select, apply and verify adversarial attacks on STPs. We demonstrate that the vanilla version significantly outperforms standard methods, and can be further improved via a simple best-of-k decoding enhancement.

3. LAVA exposes naive heuristics and global weaknesses common across multiple STPs, such as a flawed usage of quantified rules. Training on the adversarial samples improves the target model's performance.

Our implementation is available at https://github.com/alexgaskell10/LAVA.

2 Background: Soft Theorem-Proving

[Clark *et al.*, 2020] examine neural systems' capacities to conduct automated reasoning using language and introduce the *RuleTakers* dataset to this end. They demonstrate that Transformers can act as "soft theorem provers" over natural language by solving a logical deduction task. Their objective is to determine whether a natural language claim, the *query*, is "entailed" by a set of rules and facts, the *context*.

To generate the dataset, the authors first synthesise stratified logic programs, amounting to sets of rules and facts. Facts are atomic sentences and rules are implications that have, as a body, conjunctions of atoms and, as the head, atoms. The authors also consider more general versions of rules and facts, where atoms may be negated. All rules can be propositional or (implicitly) universally quantified. Then, rules and facts are converted to natural language representations using templates (see Fig. 1 for an example).

To generate labels for queries in the dataset, the authors use the closed world assumption and an interpretation of negation as negation-as-failure (NAF), adapting the standard semantics for stratified logic programs to allow a symbolic solver to label whether a query is entailed by a context. We use $\models$ to represent this entailment. We illustrate NAF in Fig. 1, Q3., as Sarah cannot be shown to be happy so she is smart.

The dataset is subdivided on a $[0..5]$ range according to the number of deductive steps required (*proof depth*), approximating sample difficulty. Proof depth is closely related to proof *length*, the number of context sentences in the proof.

3 Logically Consistent Adversarial Attacks

We introduce LAVA as a framework for generating logically consistent adversarial attacks. In §3.1 we define our problem statement and introduce the *attacker* and *victim*, and derive our objective and gradients in §3.2. LAVA's overall flow is illustrated in Fig. 2. The sample is first fed into the attacker which predicts the perturbations to apply, relating to stages 2 and 3 from Fig. 2 and §3.4. The perturbed sample is fed into the victim (stage 5). Logical consistency is ensured by recomputing the label for the perturbed sample, relating to step 6 in Fig. 2 and §3.5, and this is used to compute the attacker's learning signal (stage 7 in Fig. 2).

3.1 Problem Definition

STP aims to predict if a context, c, entails, y, a query, q:

$$\max_{\theta_{\mathcal{V}}} \; \log p_{\theta_{\mathcal{V}}}(y \mid q, c) \quad \text{with} \quad y = \mathbb{1}(c \models q),$$

where $\mathbb{1}$ is the indicator function and $p_{\theta_{\mathcal{V}}}$ is a probability distribution over the entailment label. We refer to the STP, parameterized with $\theta_{\mathcal{V}}$ (a Transformer), as the **victim**, as this is the model we are attacking. We probe the victim using

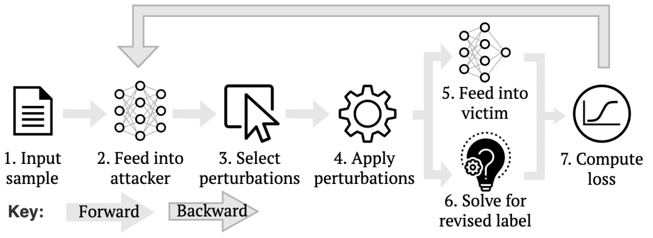


Figure 2: An overview of LAVA. The attacker predicts which perturbations to apply. The perturbed sample is fed into the victim and its loss is computed against the revised label, obtained using the *solver*. This acts as the attacker's learning signal.

adversarial attacks, perturbing the query, q', and/or context, c', so as to fool the victim into predicting the opposite label, $\hat{y} = 1 - y = 1 - \mathbb{1}(c' \models q')$. We train an **attacker**, parameterized by $\theta_{\mathcal{A}}$, to generate victim-fooling perturbations, (q'_i, c'_i), conditioned on the original sample. We represent this as the distribution $p_{\theta_{\mathcal{A}}}(q'_i, c'_i \mid q, c)$, abbreviated to $p_{\theta_{\mathcal{A}}}(q', c')$. The objective now is to sample (denoted $\sim$) perturbations from the attacker's distribution to maximize the log likelihood of fooling the victim:

$$\max_{\theta_{\mathcal{A}}} \; \log p_{\theta_{\mathcal{V}}}(\hat{y} \mid q', c') \quad \text{where} \quad (q', c') \sim p_{\theta_{\mathcal{A}}}(q', c'). \tag{1}$$

Note that we only train the attacker's parameters and the victim is solely used for querying its probabilities (without gradient access), making this a *black-box* adversarial attack method suitable with any text-based binary classifier.

3.2 Deriving the Objective Function

We now introduce the perturbations, q' and c', as latent variables. We seek to maximize the (log) probability of incorrect victim answers, $J = \log p_{\theta_{\mathcal{V}}}(\hat{y} \mid q, c)$. Marginalizing over the perturbations, we obtain

$$\begin{aligned} J &= \log \sum_{q',c'} p_{\theta_{\mathcal{V}}}(\hat{y} \mid q', c') p_{\theta_{\mathcal{A}}}(q', c' \mid q, c) \\ &= \log \mathop{\mathbb{E}}_{p_{\theta_{\mathcal{A}}}(q',c')} \left[p_{\theta_{\mathcal{V}}}(\hat{y} \mid q', c') \right], \end{aligned} \tag{2}$$

where perturbations are sampled from the attacker's joint distribution, as outlined in Eq. 1. Following [Mnih and Gregor, 2014], we derive a lower bound using Jensen's inequality, which says that for any convex function $g(x)$, $\mathbb{E}[g(x)] \geq g(\mathbb{E}[x])$ [Jordan *et al.*, 1999]:

$$J \geq \mathop{\mathbb{E}}_{p_{\theta_{\mathcal{A}}}(q',c')} \left[\log p_{\theta_{\mathcal{V}}}(\hat{y} \mid q', c') \right].$$

Expanding the expectation gives our **final optimisation problem**, $\mathcal{L}$:

$$\begin{aligned} \max_{\theta_{\mathcal{A}}} \mathcal{L} &= \sum_{q',c'} p_{\theta_{\mathcal{A}}}(q', c') \log p_{\theta_{\mathcal{V}}}(\hat{y} \mid q', c') \\ & s.t. \; \hat{y} = 1 - \mathbb{1}(c' \models q'). \end{aligned} \tag{3}$$

From Eq. 3 we obtain an expression for the attacker's gradients. We subtract a baseline b to reduce the variance of the learning signal without affecting its expectation:

$$\nabla_{\theta_{\mathcal{A}}} \mathcal{L} = \sum_{q',c'} \left(\log p_{\theta_{\mathcal{V}}}(\hat{y} \mid q', c') - b \right) \nabla_{\theta_{\mathcal{A}}} p_{\theta_{\mathcal{A}}}(q', c'). \tag{4}$$

$\mathcal{O}$ **q:** Dave is not big. **c:** All red people are smart. All smart people are furry. Charlie is blue. Charlie is furry. Dave is big. Dave is smart. Erin is big. Fiona is blue. Fiona is round. Furry, smart people are nice. If someone is nice and red then they are furry. Nice, furry people are big. Round people are red. Label: False Victim prediction: False

$\mathcal{S}$ q: Dave is big. c: All red people are smart. All smart people are furry. Fiona is blue. Furry, smart people are nice. If Fiona is blue and Charlie is furry then Dave is smart. If Fiona is round and Charlie is furry then Dave is big. If Fiona is round and Dave is big then Erin is big. If someone is nice and red then they are furry. Nice, furry people are big. Round people are red. Label: False Victim prediction: True

Figure 3: A randomly sampled successful attack, $\mathcal{S}$. The perturbations applied to the original $\mathcal{O}$ are coloured green, red and blue corresponding to QuesFlip, SentElim and EquivSub strategies respectively.

Eq. 4 relates to the backward pass component of Fig. 2. As in [Mnih and Gregor, 2014], the baseline is an exponentially-decaying moving average of $\log p_{\theta_\mathcal{V}}(\hat{y} \mid q', c')$.

3.3 Attack Strategies

Generating textual adversarial attacks is challenging as there is no universal definition for measuring semantic distances between texts, making it difficult to quantify the distributional shift introduced during attacks. Controlling which perturbations preserve the sample's label is another challenge. These are often addressed by assuming locality, namely that several word/sub-word/character perturbations are unlikely to materially impact the text's semantics [Zhu *et al.*, 2020]. This assumption is invalid for logical entailment as this is sensitive even to local perturbations - the *logical inconsistency* problem. As for text generation in general, ensuring that outputs are fluent and coherent is also a consideration.

We address these challenges using a structured generative process, specifically by constraining LAVA to select from a set of perturbations. These perturbations were chosen to exploit the logical semantics underlying the data, probing the victim's robustness to logical transformations such as negation. These are implemented using deterministic rules which we refer to as templates. The use of these perturbations ensures the attacks are linguistically aligned with the original data distribution, while also guaranteeing grammaticality and fluency. The remainder of this section outlines the perturbations candidates (stage 4 in Fig. 2), summarised in Table 1. These are annotated on a successful LAVA attack in Fig. 3.

Question flipping (QuesFlip) Predict whether to negate the question. [Kassner and Schütze, 2020] motivate this by showing that neural models struggle with negation. To illustrate the use of templates, the sentence *"The **entity** is **attribute**"* may be mapped into: *"The **entity** is not **attribute**"*.

Sentence elimination (SentElim) Predict whether to retain each context sentence in the perturbed output.

Equivalence substitution (EquivSub) Predict whether each context fact should be substituted with a logically equivalent propositional rule where the fact is the rule head and the body consists of other (randomly sampled) context facts.

Perturb.	Before	After
QuesFlip:	$q, \{a, b, a \rightarrow c\}$	$\neg q, \{a, b, a \rightarrow c\}$
SentElim:	$q, \{a, b, a \rightarrow c\}$	$q, \{\cancel{a}, b, \cancel{a \rightarrow c}\})$
EquivSub:	$q, \{a, b, a \rightarrow c\}$	$q, \{\cancel{a}, b, a \rightarrow c, b \rightarrow a\}$

Table 1: Illustration of the perturbations on a logic program with query q and context $\{a, b, a \rightarrow c\}$.

3.4 Attacker Architecture

The attack model outputs a categorical distribution with $2N^c + 1$ parameters, N^c for EquivSub and SentElim and one for QuesFlip, where N^c is the number of context sentences. An attack is created by sampling perturbations and applying them to the input whilst recomputing the associated label. The perturbed output is fed into the victim.

The attacker's architecture is similar to [Saha *et al.*, 2020], differing mainly through our three output heads, g^1, g^2 and g^3, for the attack strategies in §3.3. The input, $x = [CLS]\ q\ [SEP]\ c\ [SEP]$, with ids $\mathbf{x}$, is fed into a Transformer encoder, $\mathrm{E}_\mathcal{A}$, giving hidden states $\mathbf{H} = \mathrm{E}_\mathcal{A}(\mathbf{x})$. We denote the sequence of tokens and hidden states corresponding to sentence j as c_j and $\mathbf{H}_j$ respectively with $c_0 := q$. The representation, $\mathbf{h}_j$, is formed using mean pooling as $\mathbf{h}_j = \text{mean\_pool}(\mathbf{H}_j)$. This is fed into the output heads, g^i, to obtain the categorical parameter, μ^i_j:

$$\mu^i_j = g^i(\mathbf{h}_j) = \sigma\left(\mathbf{W}^i_1 \text{ReLU}\left(\mathbf{W}^i_2 \mathbf{h}_j + \mathbf{b}^i_2\right) + b^i_1\right). \quad (5)$$

$\mathbf{W}_1, \mathbf{W}_2, b_1, \mathbf{b}_2$ are learnable weights and $\sigma(z) = 1/(1 + e^{-z})$. The question representation, $\mathbf{h}_0$, is only fed into the QuesFlip head, g^3, while other representations, $\mathbf{h}_1, \ldots, \mathbf{h}_{N^c}$, are fed into the SentElim and EquivSub heads, g^1 and g^2.

The attacker is trained using a REINFORCE-style objective [Williams, 1992], with the victim's loss as the learning signal. Crucially, the logically consistent label is used to compute the loss and attacker's gradients; else the attacker may learn to produce inconsistent attacks. This relates to stage 6 in Fig. 2 and §3.5.

3.5 Logically Consistent Attacks

To address the logical consistency challenge outlined in §3.3, we must solve for the perturbed sample's entailment relationship ($c' \models q'$ in Eq. 3). While the entailment label is predictable under the EquivSub and QuesFlip perturbations (the former preserves and the latter flips the label), the impact of applying SentElim is unpredictable. Our solution is the *solver* module which uses a symbolic solver to recompute the label for the perturbed sample, denoted the *modified* label. LAVA uses ProbLog [Raedt *et al.*, 2007] as the solver but could be easily adapted to use an alternative if desired. We apply the perturbations to the natural language sentences as well as the underlying logic programs which were used to generate them. The perturbed logic program is then fed into the *solver* module to compute the modified label. This occasionally ($< 2\%$)

fails due to negative cycles in the perturbed sample, in which case we denote the attack as unsuccessful.

4 Experiments

To our knowledge, LAVA is the first to use logically consistent attacks. Here, we first introduce baselines and metrics (§4.1 - 4.2). Next, we validate LAVA's effectiveness by showing that it successfully generates attacks (§4.3) which transfer across victims and address the victims' weaknesses when trained upon (§4.4). All results are for the RuleTakers test set (20,192 samples), using the validation set for early stopping.

4.1 Benchmarks

HotFlip (HF) This is a white-box attack method which uses the victims' gradients to estimate the loss of substituting words/characters in the input text, selecting the highest-impact perturbation [Ebrahimi *et al.*, 2018]. For comparability, we use the black-box OpenAttack variant [Zeng *et al.*, 2021], substituting words with synonyms and replacing gradient-guided perturbation search with brute force.

TextFooler (TF) This is a black-box method which generates attacks by substituting input words with synonyms and ranking them by impact, verifying that the substitutions maintain grammaticality and semantic similarity [Jin *et al.*, 2020]. We use the OpenAttack [Zeng *et al.*, 2021] implementation.

As mentioned in §3.5, perturbing the input can compromise a sample's logical consistency, resulting in invalid attacks. We thus integrate the *solver* module into the above HF and TF, modifying the logic programs and computing a label for the perturbed sample, as in §3.5. We report the logical consistency-adjusted attack success rates below.

Random Selector (RS) This sets $\mu_j^i = 0.5$ from Eq. 5. This ablates LAVA's learned component and makes its perturbations independent of the input, providing a comparison against an attacker that selects perturbations randomly.

Unigram Selector (US) This is a heuristic version of RS. Perturbation probability is biased towards sentences with greater word overlap with the question, replacing Eq. 5 with

$$\mu_j^i = 0.5 - \bar{r} + r_j, \text{ where } r_j = \text{ROUGE-1}(c_0, c_j).$$

ROUGE-1 [Lin, 2004] computes the unigram overlap score between context sentence c_j and the query, c_0. $\bar{r}$ is the mean unigram score over the training data, giving the same expected categorical parameters as under RS.

4.2 Metrics

Attack success rate (ASR) This represents the attacker's performance: ASR = # successful attacks / # attacks.

F1 sentence overlap score (F1) This is a measure of the distance between perturbed and original examples in terms of sentence overlap. Higher scores reflect more overlap and while a low F1 is bad ("adversarial" implies minimal perturbations), we do not seek to maximise F1 as this can overly constrain the attacker and restrict attacks' diversity (thus, a higher value for F1 is not necessarily preferable).

Prior studies [Simoncini and Spanakis, 2021; Li *et al.*, 2020] report grammaticality and fluency, but these are unnecessary here due to our template-based generative process.

	Name	**ASR (%) ↑**	**F1 (%)**
Baselines	TF	10.1	88.9
	RS	10.4	66.9
	US	12.1	65.3
	HF	21.5	94.5
Abls.	$\mathcal{R}_l-$ SentElim	10.0	74.3
	$\mathcal{R}_l-$ EquivSub	10.9	80.3
	$\mathcal{R}_l-$ QuesFlip	25.1	68.4
Ours	$\mathcal{R}_l$ **(main)**	28.6	67.6
	$\mathcal{R}_b$	30.1	67.8
	$\mathcal{R}_d$	52.5	68.6

Table 2: Attacker results. Our main model (used throughout this study) uses RoBERTa-large, $\mathcal{R}_l$, as the victim. We provide RoBERTa-base, $\mathcal{R}_b$, and Distil-RoBERTa-base, $\mathcal{R}_d$, as references as these are weaker (easier to attack) victims, hence ASR is higher. The baselines (see §4.1) and ablations (Abls.) use $\mathcal{R}_l$ as the victim so should be compared to $\mathcal{R}_l$ **(main)**. The ablations begin from $\mathcal{R}_l$ **(main)** and show which attack strategy was removed. Higher is better for ASR, but not necessarily for F1, as per §4.2.

4.3 Attacker Performance

Before training our attacker, we train the victims until convergence on the RuleTakers dataset. Unless specified, the victim's core uses RoBERTa-large, $\mathcal{R}_l$, [Liu *et al.*, 2019] and we provide RoBERTa-base, $\mathcal{R}_b$, and Distil-RoBERTa-base, $\mathcal{R}_d$, [Sanh *et al.*, 2019] for comparison. The attacker is trained for five epochs with a batch size of 8 on a single 11Gb NVIDIA GeForce RTX 2080 GPU. The learning rate was set to 5e^{-6}, with 1e^{-5} and 2.5e^{-6} also tested. We provide configuration files in the accompanying code for reproducibility.

ASR Table 2 gives the attackers' results. We report the four benchmarks from §4.1 and model ablations. For HF and TF, we report the adjusted results, described in §4.1, to ensure logical consistency. The unadjusted ASRs are 81% and 84% (not shown in the table) respectively, vs 22% and 10% after adjusting. Recall that the unadjusted figures contain many logically-inconsistent attacks, hence we only discuss the adjusted ASR. This demonstrates that naively perturbing the context will likely flip the entailment relationship, hence integrating the *solver* within the generative process is desirable. This result corroborates recent findings that adversarial methods often generate invalid examples [Mozes *et al.*, 2021]. Using an independent t-test, our method significantly outperforms all baselines. The F1 metric is discussed in §4.5.

Best-of-k Decoding The results in Table 2 use *one-sample* decoding: the attacker generates one attack for each sample.

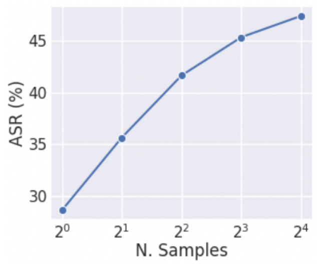


Figure 4: ASR vs $\mathcal{N}^k$.

Method	Victim	Trf. ASR (%) ↑	Atk. ASR (%)
Ours	$\mathcal{R}_l \rightarrow \mathcal{R}_b$	84.4	28.6
	$\mathcal{R}_l \rightarrow \mathcal{R}_d$	82.1	28.6
	$\mathcal{R}_b \rightarrow \mathcal{R}_d$	81.8	30.1
	$\mathcal{R}_b \rightarrow \mathcal{R}_l$	80.2	30.1
TF	$\mathcal{R}_l \rightarrow \mathcal{R}_b$	64.6	10.1
HF	$\mathcal{R}_l \rightarrow \mathcal{R}_b$	55.1	21.5

Table 3: The transferability, *Trf.*, of adversarial samples. $\mathcal{V}_x \rightarrow \mathcal{V}_y$ means the attacks were generated against victim $\mathcal{V}_x$ and are evaluated against $\mathcal{V}_y$. Attacker, *Atk.*, ASR is shown for comparison.

Victim	DSet	Before	After ↑	Δ (%) ↑
$\mathcal{R}_l$	$\mathcal{D}^A$	71.4	96.9	35.7
	$\mathcal{D}^O$	99.3	99.7	0.4
$\mathcal{R}_b$	$\mathcal{D}^A$	69.9	94.9	35.8
	$\mathcal{D}^O$	97.9	98.6	0.7
$\mathcal{R}_d$	$\mathcal{D}^A$	47.5	85.4	79.7
	$\mathcal{D}^O$	75.7	89.1	17.7

Table 4: Adversarial training results showing the victim's accuracy on the original, $\mathcal{D}^O$, and adversarial, $\mathcal{D}^A$, datasets before and after training on the augmented dataset, $\mathcal{D}^C$.

Alternatively, we can use *best-of-k decoding* by taking $\mathcal{N}^k$ adversarial samples and retaining the one which most successfully fools the victim. Fig. 4 shows the benefits of best-of-k decoding, yielding an ASR of 47% when $\mathcal{N}^k = 16$.

The $\mathcal{R}_l$ victim performs near-perfectly on the original task with 99.3% accuracy (reported in Table 4, discussed in §4.6). Given this, it is interesting that the attacker fools the victim with 29% and 49% ASR when $\mathcal{N}^k = 1$ and 16 respectively. This reveals weaknesses in the victim's reasoning capabilities which the attacker learns to exploit.

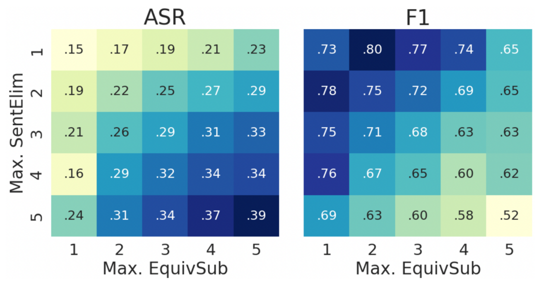
Ablations Here we ablate each perturbation strategy from §3.3 and compare this against the performance when using all three ($\mathcal{R}_l$ **(main)** in Table 2). Using a t-test, all ablations significantly reduce the attacker's ASR ($p < 3e^{-15}$). Equipping the attacker with a greater range of perturbations helps it to identify weaknesses in the victim's reasoning capabilities.

4.4 Transferability

In general, when a model is used to generate attacks, a concern is that the attacker can overfit to a single victim. This can be formalized using *transferability* [Chen *et al.*, 2017], defined as the proportion of the successful attacks generated against victim x, denoted $\mathcal{V}_x$, that also successfully fool a different victim y, denoted $\mathcal{V}_y$, shown in Table 3. In Table 3, the top row says that 84% of the successful attacks against $\mathcal{R}_l$ also succeed against $\mathcal{R}_b$. Our methods' attacks are more transferable than the baselines, implying that LAVA identifies global rather than local vulnerabilities.

4.5 ASR vs Number of Perturbations

The results according to the two metrics, ASR and F1, cannot be taken in isolation, since they encapsulate an important trade-off between the ease of fooling the victim and the number of applied perturbations. In our early experiments we observed that the attacker could easily achieve a high ASR by perturbing most of the context sentences. This shifts the original and perturbed samples' distributions, making it trivial to fool the victim but hard to attribute its failure to poor reasoning versus out-of-distribution generalization. Addressing this, we introduce hyper-parameters capping the number of SentElim and EquivSub perturbations the attacker can apply. Fig. 5 shows the trade-off between ASR and F1: increasing this threshold leads to more successful attacks but also makes the original and perturbed samples less similar. Note that these parameters specify the maximum number of perturbations; the attacker can apply fewer. Interestingly, our main model (with the caps set at 3) uses 2.3 EquivSub perturbations on average. As EquivSub adds a reasoning step, thus in principle making the sample more difficult, we expected the attacker to use EquivSub as frequently as permitted.



ASR:

Max. SentElim \ Max. EquivSub	1	2	3	4	5
1	.15	.17	.19	.21	.23
2	.19	.22	.25	.27	.29
3	.21	.26	.29	.31	.33
4	.16	.29	.32	.34	.34
5	.24	.31	.34	.37	.39

F1:

Max. SentElim \ Max. EquivSub	1	2	3	4	5
1	.73	.80	.77	.74	.65
2	.78	.75	.72	.69	.65
3	.75	.71	.68	.63	.63
4	.76	.67	.65	.60	.62
5	.69	.63	.60	.58	.52

Figure 5: Heatmaps depicting the impact of the maximum SentElim and EquivSub perturbations on ASR and F1 score.

The results reported elsewhere in this study assume the SentElim and EqivSub caps are set at 3, the level we consider to optimally trade-off ASR and F1. In Table 2 the F1 scores of HF and TF are near 1, reflecting the local nature of these methods' attacks which limits their diversity. This metric does not capture the fact that logically equivalent sentences can be realised linguistically distinctly, thus requiring a high F1 score can overly constrain the attacker.

4.6 Adversarial Training

The previous sections demonstrated that an attacker can learn to exploit weaknesses in the victim's reasoning capabilities. Here, we investigate whether training the victim on the successful attacks helps address these blind spots. To test this, we initially train a victim to convergence on the original dataset $\mathcal{D}^O$ and train an attacker against this victim. The attacker is evaluated on $\mathcal{D}^O$, generating an augmented dataset of adversarial samples, $\mathcal{D}^A$. We then resume training the victim on the combined dataset $\mathcal{D}^C = \mathcal{D}^O \bigcup \mathcal{D}^A$.

The results in Table 4 show that training on $\mathcal{D}^C$ improves the victims' performance on $\mathcal{D}^O$ and $\mathcal{D}^A$. Using an independent t-test, all differences are significant with $p < 1e^{-7}$. While there is limited headroom for improvement on $\mathcal{D}^O$ using $\mathcal{R}_l$ and $\mathcal{R}_b$ as victims, the weaker distilled model, $\mathcal{R}_d$, sees a sizable 17.5% improvement. These show that data augmentation via logically consistent adversarial examples is a promising strategy for enhancing STPs.

5 Analysis of Attacks

Qualitative We have identified several blind spots in the victim's reasoning capabilities by qualitatively analysing the successful attacks. Fig. 3 displays an example of one such vulnerability, whereby the victim is often fooled if the query literal appears within the body of a rule, i.e. "If Fiona is round and *Dave is big* then ...". This can be exploited by the attacker's EquivSub perturbation. Another common failure type is when the attacker eliminates a leaf fact on samples containing deeper proofs. This operation flips the label while the victim's prediction is unchanged. This suggests the victim performs backward-chaining from the query, but its search is shallow and terminates before the proof tree is complete.

More generally, the victim appears brittle when using quantified rules. It struggles to correctly bind variables on either side of an implication, e.g. "if someone is happy then they like John" versus "if someone is happy then Anne likes John". Similarly, many failures occur on conjunctive implications containing both variables and facts. Using the example "if Anne likes John and someone is happy then they like Beth", *someone* and *they* refer to a common variable but could be misconstrued as relating to *Anne* or *John*. However, none of the attacker's perturbations can create quantified rules so it cannot exploit these vulnerabilities.

These observations yield intriguing insights into STPs' reasoning processes and suggest avenues for improving the STPs and attackers. The first and second failure modes imply that short-cuts are used, although we believe they play a minor role else the victim would prove more brittle. The failures involving quantified rules suggest a coarse word-sense disambiguation mechanism which is not scaling to more complex rules. It also suggests an incomplete understanding of the properties of variables within logic programs.

Quantitative Breaking down the attacker's ASR by proof length, we see that the ASR is 0%, 19%, 30% and 47% for proofs of length 0, 1, 2 and $\geq$3 respectively. Predictably, as samples get harder the victim is more easily fooled. Note that for proofs of length zero (20% of samples) the attacker's ASR is zero. By definition, proofs of length zero have no context sentences unifying with the query (it does not "match" any facts or rule heads), rendering attacks ineffective as the attacker cannot synthesise a fact/rule to link the query and the context.

6 Related Work

Transformers as Soft Theorem Provers Our study is closely related to the *RuleTakers* line of work [Clark *et al.*, 2020]. These posit that Transformers can act as "soft theorem provers" over natural language sentences, analogous to symbolic theorem provers for formally represented theories. These studies exhibit impressive model capabilities, such as near-perfect entailment prediction, an ability to generate reasoning proofs [Saha *et al.*, 2020], generalizing beyond the training set and reasoning over implicit and explicit knowledge [Talmor *et al.*, 2020]. Our work stands apart by identifying weaknesses in these approaches using adversarial attacks.

Other studies have noted the shortcomings of Transformers on reasoning problems. Vulnerability to deeper Horn Rule reasoning [Gontier *et al.*, 2020], word-order permutations [McCoy *et al.*, 2019], mis-priming [Misra *et al.*, 2020] and shallow heuristics [Williams *et al.*, 2018] have all been observed (see [Helwe *et al.*, 2021] for more examples). Our study is distinct from these as we train a generative adversary end-to-end to discover these weaknesses.

Adversarial Attacks These have seen recent attention by the NLP community as a method for probing model robustness and creating more challenging datasets. Borrowing the taxonomy in [Simoncini and Spanakis, 2021], these can be grouped according to: 1) specificity - are the attacks targeted at a label? 2) Knowledge of the victim - *white-box* attacks assume all information about the victim is known, whereas *black-box* attacks assume only its probabilities are known. 3) Granularity - are characters, tokens or phrases perturbed? We add a fourth *data generation process* criterion - is it solely a human, a hybrid human-in-the-loop or a solely model-based procedure? Using this taxonomy, our model is an unspecified, black-box, phrase-level, model-based attack procedure.

Several methods have recently emerged at character-level [Ebrahimi *et al.*, 2018; Gao *et al.*, 2018] and word-level [Li *et al.*, 2020; Jin *et al.*, 2020]. These generally rely on inserting/removing characters or substituting words with synonyms or tokens with similar embeddings. These methods implicitly assume that the locality of attacks preserves the original passage's semantics. This assumption has proven invalid [Mozes *et al.*, 2021], particularly for logical reasoning we posit as entailment is sensitive to token-level perturbations.

LAVA was inspired by a recent movement towards training generative models for adversarial attacks. Ours is closely related to SAGE [Zhu *et al.*, 2020], a Wasserstein Auto-Encoder trained to attack TableQA systems. A particular challenge in this domain is controlling the semantic shift introduced by attacks [Mozes *et al.*, 2021] - logical consistency can be viewed as a special case of semantic shift. Other generative strategies have been successfully employed, such as using GANs [Ren *et al.*, 2020] or integrating victim gradients [Guo *et al.*, 2021].

7 Conclusions and Future Work

We use adversarial examples to probe the robustness of STPs. To this end, we propose LAVA, a black-box attack framework which learns to exploit vulnerabilities in the victim's reasoning capabilities. We define the *logical inconsistency* problem, namely that a given sample's logical entailment relationship may be sensitive to even small perturbations, and we address this via a structured generative process in conjunction with a integrated symbolic solver. LAVA outperforms standard methods in terms of attack success rate and transferability and training the victim on the adversarial samples improves performance. Our analyses reveal weaknesses and crude heuristics in the victim's reasoning processes, exposing a flawed grasp of the semantics of logic programs. Future work may seek to equip the attacker with the ability to synthesise rules as we expect this would expose a wider range of vulnerabilities. Going forward, we envisage that LAVA could play a useful role in future STP development as a framework for identifying and debugging their vulnerabilities.

Acknowledgments

This work was supported by UKRI [grant number EP/S0233356/1], in the CDT in Safe and Trusted AI. We thank Alex Spies for his insightful comments and feedback.

References

[Chen *et al.*, 2017] P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh. ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM*, 2017.

[Clark *et al.*, 2020] P. Clark, O. Tafjord, and K. Richardson. Transformers as soft reasoners over language. In *IJCAI*, 2020.

[Ebrahimi *et al.*, 2018] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. Hotflip: White-box adversarial examples for text classification. In *ACL*, 2018.

[Gao *et al.*, 2018] J. Gao, J. Lanchantin, M. Soffa, and Y. Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *IEEE*, 2018.

[Gontier *et al.*, 2020] N. Gontier, K. Sinha, S. Reddy, and C. Pal. Measuring systematic generalization in neural proof generation with transformers. In *NeurIPS*, 2020.

[Guo *et al.*, 2021] C. Guo, A. Sablayrolles, H. Jégou, and D. Kiela. Gradient-based adversarial attacks against text transformers. In *EMNLP*, 2021.

[Helwe *et al.*, 2021] C. Helwe, C/ Clavel, and F. Suchanek. Reasoning with transformer-based models: Deep learning, but shallow reasoning. In *AKBC*, 2021.

[Jin *et al.*, 2020] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *AAAI*, 2020.

[Jordan *et al.*, 1999] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, 1999.

[Kassner and Schütze, 2020] N. Kassner and H. Schütze. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *ACL*, 2020.

[Li *et al.*, 2020] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu. BERT-ATTACK: adversarial attack against BERT using BERT. In *EMNLP*, 2020.

[Lin, 2004] C. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.

[Liu *et al.*, 2019] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv e-prints*, page arXiv:1907.11692, 2019.

[McCoy *et al.*, 2019] Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *ACL*, 2019.

[Misra *et al.*, 2020] K. Misra, A. Ettinger, and J. Rayz. Exploring bert's sensitivity to lexical cues using tests from semantic priming. In *EMNLP*, 2020.

[Mnih and Gregor, 2014] A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. In *ICML*, 2014.

[Mozes *et al.*, 2021] M. Mozes, M. Bartolo, P. Stenetorp, B. Kleinberg, and L. Griffin. Contrasting human & machine-generated word-level adv. examples for text classification. In *EMNLP*, 2021.

[Petroni *et al.*, 2019] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. H. Miller. Language models as knowledge bases? In *EMNLP-IJCNLP*, 2019.

[Raedt *et al.*, 2007] L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, 2007.

[Ren *et al.*, 2020] Y. Ren, J. Lin, S. Tang, J. Zhou, S. Yang, Y. Qi, and X. Ren. Generating natural language adversarial examples on a large scale with generative models. In *ECAI*, 2020.

[Saha *et al.*, 2020] S. Saha, S. Ghosh, S. Srivastava, and M. Bansal. Prover: Proof generation for interpretable reasoning over rules. In *EMNLP*, 2020.

[Sanh *et al.*, 2019] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv e-prints*, page arXiv:1910.01108, 2019.

[Simoncini and Spanakis, 2021] W. Simoncini and G. Spanakis. Seqattack: On adversarial attacks for named entity recognition. In *EMNLP*, 2021.

[Szegedy *et al.*, 2014] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[Talmor *et al.*, 2020] A. Talmor, O. Tafjord, P. Clark, Y. Goldberg, and J. Berant. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. In *NeurIPS*, 2020.

[Vaswani *et al.*, 2017] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[Williams *et al.*, 2018] A. Williams, N. Nangia, and S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, 2018.

[Williams, 1992] R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8, 1992.

[Yu *et al.*, 2020] W. Yu, Z. Jiang, Y. Dong, and J. Feng. Reclor: A reading comprehension dataset requiring logical reasoning. In *ICLR*, 2020.

[Zeng *et al.*, 2021] G. Zeng, F. Qi, Q. Zhou, T. Zhang, B. Hou, Y. Zang, Z. Liu, and M. Sun. Openattack: An open-source textual adversarial attack toolkit. In *IJCAI*, 2021.

[Zhu *et al.*, 2020] Y. Zhu, Y. Zhou, and M. Xia. Generating Semantically Valid Adversarial Questions for TableQA. *arXiv e-prints*, page arXiv:2005.12696, 2020.