Semantically Equivalent Adversarial Rules for Debugging NLP Models

Marco Tulio Ribeiro
University of Washington
marcotcr@cs.uw.edu

Sameer Singh
University of California, Irvine
sameer@uci.edu

Carlos Guestrin
University of Washington
guestrin@cs.uw.edu

Abstract

Complex machine learning models for NLP are often brittle, making different predictions for input instances that are extremely similar semantically. To automatically detect this behavior for individual instances, we present *semantically equivalent adversaries* (SEAs) – semantic-preserving perturbations that induce changes in the model's predictions. We generalize these adversaries into *semantically equivalent adversarial rules* (SEARs) – simple, universal replacement rules that induce adversaries on many instances. We demonstrate the usefulness and flexibility of SEAs and SEARs by detecting bugs in black-box state-of-the-art models for three domains: machine comprehension, visual question-answering, and sentiment analysis. Via user studies, we demonstrate that we generate high-quality local adversaries for more instances than humans, and that SEARs induce four times as many mistakes as the bugs discovered by human experts. SEARs are also actionable: retraining models using data augmentation significantly reduces bugs, while maintaining accuracy.

1 Introduction

With increasing complexity of models for tasks like classification (Joulin et al., 2016), machine comprehension (Rajpurkar et al., 2016; Seo et al., 2017), and visual question answering (Zhu et al., 2016), models are becoming increasingly challenging to debug, and to determine whether they are ready for deployment. In particular, these complex models are prone to brittleness: different ways of phrasing the same sentence can often cause the model to output different predictions. While held-out accuracy is often useful, it is not sufficient: practitioners consistently overestimate their model's generalization (Patel et al., 2008) since test data is usually gathered in the same manner as training and validation. When deployed, these seemingly accurate models encounter sentences that are written very differently than the ones in the training data, thus making them prone to mistakes, and fragile with respect to distracting additions (Jia and Liang, 2017). These problems are exacerbated by the variability in language, and by cost and noise in annotations, making such bugs challenging to detect and fix.

A particularly challenging issue is *oversensitivity* (Jia and Liang, 2017): a class of bugs where models output different predictions for very similar inputs. These bugs are prevalent in image classifi-

In the United States especially, several high-profile cases such as Debra LaFave, Pamela Rogers, and Mary Kay Letourneau have caused increased scrutiny on teacher misconduct.

(a) Input Paragraph

Q: What has been the result of this publicity?
A: increased scrutiny on teacher misconduct

(b) Original Question and Answer

Q: What haL been the result of this publicity?
A: teacher misconduct

(c) Adversarial Q & A (Ebrahimi et al., 2018)

Q: What's been the result of this publicity?
A: teacher misconduct

(d) **Semantically Equivalent Adversary**

Figure 1: Adversarial examples for question answering, where the model predicts the correct answer for the question and input paragraph (1a and 1b). It is possible to fool the model by adversarially changing a single character (1c), but at the cost of making the question nonsensical. A **Semantically Equivalent Adversary** (1d) results in an incorrect answer while preserving semantics.

Transformation Rules	#Flips
(`WP` *is*→`WP`*'s*)	70 (1%)
(*?*→*??*)	202(3%)

(a) Example Rules

Original: What is the oncorhynchus also called? **A:** chum salmon
Changed: What's the oncorhynchus also called? **A:** keta

(b) Example for (`WP` *is*→`WP`*'s*)

Original: How long is the Rhine? **A:** 1,230 km
Changed: How long is the Rhine?? **A:** more than 1,050,000

(c) Example for (*?*→*??*)

Figure 2: **Semantically Equivalent Adversarial Rules:** For the task of question answering, the proposed approach identifies transformation rules for questions in (a) that result in paraphrases of the queries, but lead to incorrect answers (#Flips is the number of times this happens in the validation data). We show examples of rephrased questions that result in incorrect answers for the two rules in (b) and (c).

cation (Szegedy et al., 2014), a domain where one can measure the magnitude of perturbations, and many small-magnitude changes are imperceptible to the human eye. For text, however, a single word addition can change semantics (e.g. adding "not"), or have no semantic impact for the task at hand.

Inspired by adversarial examples for images, we introduce *semantically equivalent adversaries* (**SEA**s) – text inputs that are perturbed in semantics-preserving ways, but induce changes in a black box model's predictions (example in Figure 1). Producing such adversarial examples systematically can significantly aid in debugging ML models, as it allows users to detect problems that happen in the real world, instead of oversensitivity only to malicious attacks such as intentionally scrambling, misspelling, or removing words (Bansal et al., 2014; Ebrahimi et al., 2018; Li et al., 2016).

While SEAs describe local brittleness (i.e. are specific to particular predictions), we are also interested in bugs that affect the model more globally. We represent these via simple replacement rules that induce SEAs on multiple predictions, such as in Figure 2, where a simple contraction of "is"after `Wh` pronouns (what, who, whom) (2b) makes 70 (1%) of the previously correct predictions of the model "flip" (i.e. become incorrect). Perhaps more surprisingly, adding a simple "?" induces mistakes in 3% of examples. We call such rules *semantically equivalent adversarial rules* (**SEAR**s).

In this paper, we present SEAs and SEARs, designed to unveil local and global oversensitivity bugs in NLP models. We first present an approach to generate semantically equivalent adversaries, based on paraphrase generation techniques (Lapata et al., 2017), that is model-agnostic (i.e. works for any black box model). Next, we generalize SEAs into semantically equivalent rules, and outline the properties for optimal rule sets: semantic equivalence, high adversary count, and non-redundancy. We frame the problem of finding such a set as a submodular optimization problem, leading to an accurate yet efficient algorithm.

Including the human into the loop, we demonstrate via user studies that SEARs help users uncover important bugs on a variety of state-of-the-art models for different tasks (sentiment classification, visual question answering). Our experiments indicate that SEAs and SEARs make humans significantly better at detecting impactful bugs – SEARs uncover bugs that cause 3 to 4 times more mistakes than human-generated rules, in much less time. Finally, we show that SEARs are actionable, enabling the human to close the loop by fixing the discovered bugs using a data augmentation procedure.

2 Semantically Equivalent Adversaries

Consider a black box model f that takes a sentence x and makes a prediction $f(x)$, which we want to debug. We identify adversaries by generating paraphrases of x, and getting predictions from f until the original prediction is changed.

Given an indicator function $\text{SemEq}(x, x')$ that is 1 if x is semantically equivalent to x' and 0 otherwise, we define a *semantically equivalent adversary* (SEA) as a semantically equivalent instance that changes the model prediction in Eq (1). Such adversaries are important in evaluating the robustness of f, as each is an undesirable bug.

$$\text{SEA}(x, x') = \mathbb{1}\left[\text{SemEq}(x, x') \wedge f(x) \neq f(x')\right] \quad (1)$$

While there are various ways of scoring semantic similarity between pairs of texts based on embeddings (Le and Mikolov, 2014; Wieting and Gimpel, 2017), they do not explicitly penalize unnatural sentences, and generating sentences requires surrounding context (Le and Mikolov, 2014) or training a separate model. We turn instead to paraphrasing based on neural machine translation (Lapata et al., 2017), where $P(x'|x)$ (the probability of a paraphrase x' given original sentence x) is proportional to translating x into multiple pivot languages and then taking the score of back-translating the translations into the original language. This approach scores semantics and "plausibility" simultaneously (as translation models have "built in" language models) and allows for easy paraphrase generation, by linearly combining the paths of each back-decoder when back-translating.

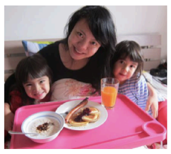
Unfortunately, given source sentences x and z, $P(x'|x)$ is not comparable to $P(z'|z)$, as each has a different normalization constant, and heavily depends on the shape of the distribution around x or z. If there are multiple perfect paraphrases near x, they will all share probability mass, while if there is a paraphrase much better than the rest near z, it will have a higher score than the ones near x, even if the paraphrase quality is the same. We thus define the semantic score $S(x, x')$ as a ratio between the probability of a paraphrase and the probability of the sentence itself:

$$S(x, x') = \min\left(1, \frac{P(x'|x)}{P(x|x)}\right) \tag{2}$$

We define $\text{SemEq}(x, x') = \mathbb{1}[S(x, x') \geq \tau]$, i.e. x' is semantically equivalent to x if the similarity score between x and x' is greater than some threshold τ (which we crowdsource in Section 5). In order to generate adversaries, we generate a set of paraphrases Π_x around x via beam search and get predictions on Π_x using the black box model until an adversary is found, or until $S(x, x') < \tau$. We may be interested in the best adversary for a particular instance, i.e. $\text{argmax}_{x' \in \Pi_x} S(x, x')\text{SEA}_x(x')$, or we may consider multiple SEAs for generalization purposes. We illustrate this process in Figure 3, where we generate SEAs for a VQA model by generating paraphrases around the question, and checking when the model prediction changes. The first two adversaries with highest $S(x, x')$ are semantically equivalent, the third maintains the semantics enough for it to be a useful adversary, and the fourth is ungrammatical and thus not useful.

3 Semantically Equivalent Adversarial Rules (SEARs)

While finding the best adversary for a particular instance is useful, humans may not have time or patience to examine too many SEAs, and may not be able to generalize well from them in order to understand and fix the most impactful bugs. In this section, we address the problem of generalizing local adversaries into Semantically Equivalent Adversarial Rules for Text (SEARs), search and replace rules that produce semantic adversaries with little or no change in semantics, when applied to a corpus of sentences. Assuming that humans have limited time, and are thus willing to look at B rules, we propose a method for selecting such a set of rules given a reference dataset X.



What color is the tray?	**Pink**
What colour is the tray?	Green
Which color is the tray?	Green
What color is it?	Green
How color is tray?	Green

Figure 3: **Visual QA Adversaries:** Paraphrasing questions to find adversaries for the original question (top, in bold) asked of a given image. Adversaries are sorted by decreasing semantic similarity.

A rule takes the form $r = (a \rightarrow c)$, where the first instance of the antecedent a is replaced by the consequent c for every instance that includes a, as we previously illustrated in Figure 2a. The output after applying rule r on a sentence x is represented as the function call $r(x)$, e.g. if $r =$(*movie*→*film*), r("Great movie!") = "Great film!".

Proposing a set of rules: In order to generalize a SEA x' into a candidate rule, we must represent the changes that took place from $x \rightarrow x'$. We will use x = "What color is it?" and x' = "Which color is it?" from Figure 4 as a running example.

One approach is exact matching: selecting the minimal contiguous sequence that turns x into x', (*What*→*Which*) in the example. Such changes may not always be semantics preserving, so we also propose further rules by including the immediate context (previous and/or next word with respect to the sequence), e.g. (*What color*→*Which color*). Adding such context, however, may make rules very specific, thus restricting their value. To allow for generalization, we also represent the antecedent of proposed rules by a product of their raw text with coarse and fine-grained Part-of-Speech tags, and allow these tags to happen in the consequent if they match the antecedent. In the running example, we would propose rules like (*What color*→*Which color*), (*What* NOUN→*Which* NOUN), (WP *color*→*Which color*), etc.

We generate SEAs and propose rules for every $x \in X$, which gives us a set of candidate rules (second box in Figure 4, *for* loop in Algorithm 1).

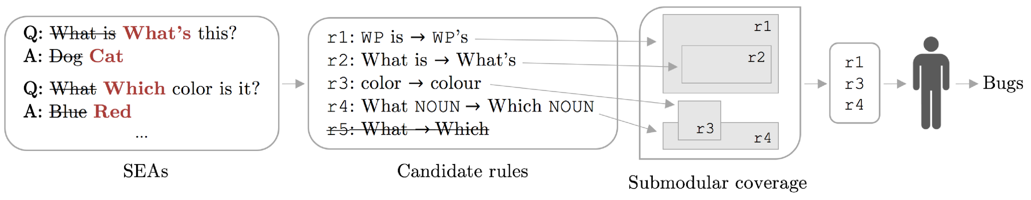


Figure 4: **SEAR process.** (1) SEAs are generalized into candidate rules, (2) rules that are not semantically equivalent are filtered out, e.g. `r5`: (*What→Which*), (3) rules are selected according to Eq (3), in order to maximize coverage and avoid redundancy (e.g. rejecting `r2`, valuing `r1` more highly than `r4`), and (4) a user vets selected rules and keeps the ones that they think are bugs.

Selecting a set of rules: Given a set of candidate rules, we want to select a set R such that $|R| \leq B$, and the following properties are met:

1. Semantic Equivalence: Application of the rules in the set should produce semantically equivalent instances. This is equivalent to considering rules that have a high probability of inducing semantically equivalent instances when applied, i.e. $E[\text{SemEq}(x, r(x))] \geq 1-\delta$. This is the *Filter* step in Algorithm 1. For example, consider the rule (*What→Which*) in Fig 4 which produces some semantically equivalent instances, but also produces many instances that are unnatural (e.g. "What is he doing?" → "Which is he doing?"), and is thus filtered out by this criterion.

2. High adversary count: The rules in the set should induce as many SEAs as possible in validation data. Furthermore, each of the induced SEAs should have as high of a semantic similarity score as possible, i.e. for each rule $r \in R$ we want to maximize $\sum_{x \in X} S(x, r(x))\text{SEA}(x, r(x))$. In Figure 4, `r1` induces more and more similar mistakes when compared to $r4$, and is thus superior to `r4`.

3. Non-redundancy: Different rules in the set may induce the same SEAs, or may induce different SEAs for the same instances. Ideally, rules in the set should cover as many instances in the validation as possible, rather than focus on a small set of fragile predictions. Furthermore, rules should not be repetitive to the user. In Figure 4 (mid), `r1` covers a superset of `r2`'s adversaries, making `r2` completely redundant and thus not included in R.

Properties 2 and 3 combined suggest a weighted coverage problem, where a rule r covers an instance x if $\text{SEA}(x, r(x))$, the weight of the connection being given by $S(x, r(x))$. We thus want to

Algorithm 1 Generating SEARs for a model

```
Require: Classifier $f$, Correct instances $X$
Require: Hyperparameters, $\delta$, $\tau$, Budget $B$
  $\mathcal{R} \leftarrow \{\}$ {Set of rules}
  for all $x \in X$ do
    $X' = \text{GenParaphrases}(X, \tau)$
    $\mathcal{A} \leftarrow \{x' \in X' \mid f(x) \neq f(x')\}$ {SEAs; §2}
    $\mathcal{R} \leftarrow \mathcal{R} \cup \text{Rules}(\mathcal{A})$
  end for
  $\mathcal{R} \leftarrow \text{Filter}(\mathcal{R}, \delta, \tau)$ {Remove low scoring SEARs}
  $\mathcal{R} \leftarrow \text{SubMod}(\mathcal{R}, B)$ {high count / score, diverse }
  return $\mathcal{R}$
```

find the set of semantically equivalent rules that:

$$\max_{R, |R| < B} \sum_{x \in X} \max_{r \in R} S(x, r(x))\text{SEA}(x, r(x)) \quad (3)$$

While Eq (3) is NP-hard, the objective is monotone submodular (Krause and Golovin, 2014), and thus a greedy algorithm that iteratively adds the rule with the highest marginal gain offers a constant-factor approximation guarantee of $1 - 1/e$ to the optimum. This is the *SubMod* procedure in Algorithm 1, represented pictorially in Figure 4, where the output is a set of rules given to a human, who judges if they are really bugs or not.

4 Illustrative Examples

Before evaluating the utility of SEAs and SEARs with user studies, we show examples in state-of-the-art models for different tasks. Note that we treat these models as black boxes, not using internals or gradients in any way when discovering these bugs.

Machine Comprehension: We take the AllenNLP (Gardner et al., 2017) implementation of BiDaF (Seo et al., 2017) for Machine Comprehension, and display some high coverage SEARs for it in Table 1 (also, Figures 1 and 2a). For each rule,

SEAR	Questions / SEAs	f(x)	Flips
What VBZ → What's	~~What is~~ What's the NASUWT?	~~Trade unions~~ Teachers in Wales	2%
	~~What is~~ What's a Hauptlied?	~~main hymn~~ Veni redemptor gentium	
What NOUN → Which NOUN	~~What resource~~ Which resource was mined in the Newcastle area?	~~coal~~ wool	1%
	~~What health~~ Which health problem did Tesla have in 1879?	~~nervous breakdown~~ relations	
What VERB → So what VERB	~~What was~~ So what was Ghandi's work called?	~~Satyagraha~~ Civil Disobedience	2%
	~~What is~~ So what is a new trend in teaching?	~~Co-teaching~~ educational institutions	
What VBD → And what VBD	~~What did~~ And what did Tesla develop in 1887?	~~an induction motor~~ laboratory	2%
	~~What was~~ And what was Kenneth Swezey's job?	~~journalist~~ sleep	

Table 1: SEARs for Machine Comprehension

SEAR	Questions / SEAs	f(x)	Flips
WP VBZ→ WP's	~~What has~~ What's been cut?	~~Cake~~ Pizza	3.3%
	~~Who is~~ Who's holding the baby	~~Woman~~ Man	
What NOUN→ Which NOUN	~~What~~ Which kind of floor is it?	~~Wood~~ Marble	3.9%
	~~What~~ Which color is the jet?	~~Gray~~ White	
color →colour	What ~~color~~ colour is the tray?	~~Pink~~ Green	2.2%
	What ~~color~~ colour is the jet?	~~Gray~~ Blue	
ADV is→ ADV's	~~Where is~~ Where's the jet?	~~Sky~~ Airport	2.1%
	~~How is~~ How's the desk?	~~Messy~~ Empty	

Table 2: SEARs for Visual QA

SEAR	Reviews / SEAs	f(x)	Flips
movie → film	Yeah, the ~~movie~~ film pretty much sucked .	~~Neg~~ Pos	2%
	This is not ~~movie~~ film making .	~~Neg~~ Pos	
film → movie	Excellent ~~film~~ movie .	~~Pos~~ Neg	1%
	I'll give this ~~film~~ movie 10 out of 10 !	~~Pos~~ Neg	
is →was	Ray Charles ~~is~~ was legendary .	~~Pos~~ Neg	4%
	It ~~is~~ was a really good show to watch .	~~Pos~~ Neg	
this →that	Now ~~this~~ that is a movie I really dislike .	~~Neg~~ Pos	1%
	The camera really likes her in ~~this~~ that movie.	~~Pos~~ Neg	
DET NOUN is →it is	~~The movie is~~ It is terrible	~~Neg~~ Pos	1%
	~~The dialog is~~ It is atrocious	~~Neg~~ Pos	

Table 3: SEARs for Sentiment Analysis

we display two example questions with the corresponding SEA, the prediction (with corresponding change) and the percentage of "flips" - instances previously predicted correctly on the validation data, but predicted incorrectly after the application of the rule. The rule (*What* VBZ→*What's*) generalizes the SEA on Figure 1, and shows that the model is fragile with respect to contractions (flips 2% of all correctly predicted instances on the validation data). The second rule uncovers a bug with respect to simple question rephrasing, while the third and fourth rules show that the model is not robust to a more conversational style of asking questions.

Visual QA: We show SEARs for a state-of-the-art visual question-answering model (Zhu et al., 2016) in Table 2. Even though the contexts are different (paragraphs for machine comprehension, images for VQA), it is interesting that both models display similar bugs. The fact that VQA is fragile to "Which" questions is because questions of this form are not in the training set, while (*color*→*colour*) probably stems from an American bias in data collection. Changes induced by these four rules flip more than 10% of the predictions in the validation data, which is of critical concern if the model is being evaluated for production.

Sentiment Analysis: Finally, in Table 3 we display SEARs for a fastText (Joulin et al., 2016) model for sentiment analysis trained on movie reviews. Surprisingly, many of its predictions change for perturbations that have no sentiment connotations, even in the presence of polarity-laden words.

5 User Studies

We compare automatically discovered SEAs and SEARs to user-generated adversaries and rules, and propose a way to fix the bugs induced by SEARs.

Our evaluation benchmark includes two tasks: visual question answering (VQA) and sentiment analysis on movie review sentences. We choose these tasks because a human can quickly look at a prediction and judge if it is correct or incorrect, can easily perturb instances, and judge if two instances in a pair are semantically equivalent or not. Since our focus is debugging, throughout the experiment we only considered SEAs and SEARs on examples that are originally predicted correctly (i.e. every adversary is also by construction a mistake). The user interfaces for all experiments in this section are included in the supplementary material.

5.1 Implementation Details

The paraphrasing model (Lapata et al., 2017) requires translation models to and from different languages. We train neural machine translation models using the default parameters of OpenNMT-py (Klein et al., 2017) for English↔Portuguese and English↔French models, on 2 million and 1 million parallel sentences (respectively) from EuroParl, news, and other sources (Tiedemann, 2012). We use the spacy library (`http://spacy.io`) for POS tagging. For SEAR generation, we set $\delta = 0.1$ (i.e. at least 90% equivalence). We generate a set of candidate adversaries as described in Section 2, and ask mechanical turkers to judge them

	Human vs SEA	Human vs HSEA
Neither	145 (48%)	127 (42%)
Only Human	47 (16%)	38 (13%)
Only SEA	**54 (18%)**	**72 (24%)**
Both	**54 (18%)**	**63 (21%)**

(a) Visual Question-Answering

	Human vs SEA	Human vs HSEA
Neither	177 (59%)	161 (54%)
Only Human	45 (15%)	40 (13%)
Only SEA	**47 (16%)**	**63 (21%)**
Both	**31 (10%)**	**36 (12%)**

(b) Sentiment Analysis

Table 4: **Finding Semantically Equivalent Adversaries:** we compare how often humans produce semantics-preserving adversaries, when compared to our automatically generated adversaries (SEA, left) and our adversaries filtered by humans (HSEA, right). There are four possible outcomes: neither produces a semantic equivalent adversary (i.e. they either do not produce an adversary or the adversary produced is not semantically equivalent), both do, or only one is able to do so.

for semantic equivalence. Using these evaluations, we identify $\tau = 0.0008$ as the value that minimizes the entropy in the induced splits, and use it for the remaining experiments. Source code and pretrained language models are available at `https://github.com/marcotcr/sears`.

For VQA, we use the multiple choice *telling* system and dataset of Zhu et al. (2016), using their implementation, with default parameters. The training data consists of questions that begin with "What", "Where", "When", "Who", "Why", and "How". The task is multiple choice, with four possible answers per instance. For sentiment analysis, we train a fastText (Joulin et al., 2016) model with unigrams and bigrams (embedding size of 50) on RottenTomato movie reviews (Pang and Lee, 2005), and evaluate it on IMDB sentence-sized reviews (Kotzias et al., 2015), simulating the common case where a model trained on a public dataset is applied to new data from a similar domain.

5.2 Can humans find good adversaries?

In this experiment, we compare our method for generating SEAs with user's ability to discover semantic-preserving adversaries. We take a random sample of 100 correctly-predicted instances for each task. In the first condition (**human**), we display each instance to 3 Amazon Mechanical Turk workers, and give them 10 attempts at creating semantically equivalent adversaries (with immediate feedback as to whether or not their attempts changed the prediction). Next, we ask them to choose the adversary that is semantically closest to the original instance, out of the candidates they generated. In the second condition (**SEA**), we generate adversaries for each of the instances, and pick the best adversary according to the semantic scorer. The third condition (**HSEA**) is a collaboration between our method and humans: we take the top 5 adversaries ranked by $S(x, x')$, and ask workers to pick the one closest to the original instance, rather than asking them to generate the adversaries.

To evaluate whether the proposed adversaries are semantically equivalent, we ask a separate set of workers to evaluate the similarity between each adversary and the original instance (with the image as context for VQA), on a scale of 1 (completely unrelated) to 5 (exactly the same meaning). Each adversary is evaluated by at least 10 workers, and considered equivalent if the median score ≥ 4. We thus obtain 300 comparisons between *human* and *SEA*, and 300 between *human* and *HSEA*.

The results in Table 4a and 4b are consistent across tasks: both models are susceptible to SEAs for a large fraction of predictions, and our fully automated method is able to produce SEAs as often as humans (left columns). On the other hand, asking humans to choose from generated SEAs (HSEA) yields much better results than asking humans to generate them (right columns), or using the highest scored SEA. The semantic scorer does make mistakes, so the top adversary is not always semantically equivalent, but a good quality SEA is often in the top 5, and is easily identified by users.

On both datasets, the automated method or humans were able to generate adversaries at the exclusion of the other roughly one third of the time, which indicates that they do not generate the same adversaries. Humans generate paraphrases differently than our method: the average character edit distance of our SEAs is 6.2 for VQA and 9.0 for Sentiment, while for humans it is 18.1 and 43.3, respectively. This is illustrated by examples in Table 5 - in Table 5a we see examples where very compact changes generate adversaries (humans were not able to find these changes though). The examples in Table 5b indicate that humans can generate adversaries that: (1) make use of the visual context in VQA, which our method does not, and (2) sig-

Dataset	Original	SEA
VQA	Where are the men? What kind of meat is on the boy's plate?	Where are the males? What sort of meat is on the boy's plate?
Sentiment	They are so easy to love, but even more easy to identify with. Today the graphics are crap.	They're so easy to love, but even more easy to identify with. Today, graphics are bullshit.

(a) Automatically generated adversaries, examples where humans failed to generate SEAs (**Only SEA**)

Dataset	Original	Human-generated SEA
VQA	How many suitcases? Where is the blue van?	How many suitcases are sitting on the shelf? What is the blue van's location?
Sentiment	(very serious spoilers) this movie was a huge disappointment. Also great directing and photography.	serious spoilers this movie did not deliver what I hoped Photography and directing were on point.

(b) Human generated adversaries, examples where our approach failed to generate SEAs (**Only Human**)

Table 5: Examples of generated adversaries

nificantly change the sentence structure, which the translation-based semantic scorer does not.

5.3 Can experts find high-impact bugs?

Here we investigate whether experts are able to detect high-impact global bugs, i.e. devise rules that flip many predictions, and compare them to generated SEARs. Instead of AMT workers, we have 26 expert subjects: students, graduates, or professors who have taken at least a graduate course in machine learning or NLP[1]. The experiment setup is as follows: for each task, subjects are given an interface where they see examples in the validation data, perturb those examples, and get predictions. The interface also allows them to create search and replace rules, with immediate feedback on how many mistakes are induced by their rules. They also see the list of examples where the rules apply, so they can verify semantic equivalence. Subjects are instructed to try to maximize the number of mistakes induced in the validation data (i.e. maximize "mistake coverage"), but only through semantically equivalent rules. They can try as many rules as they like, and are asked to select the best set of at most 10 rules at the end. This is quite a challenging task for humans (yet another reason to prefer algorithmic approaches), but we are not aware of any existing automated methods. Finally, we instruct subjects they could finish each task in about 15 minutes (some took longer, some ended earlier), in order to keep the total time reasonable.

[1]We have an IRB/consent form, and personal information was only collected as needed to compensate subjects

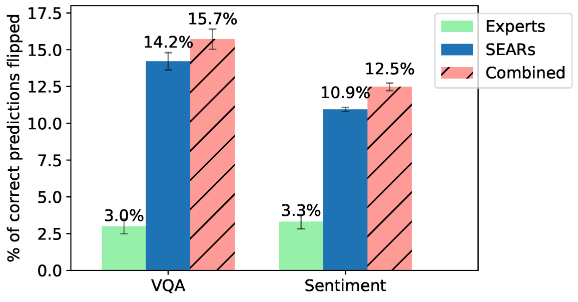


Figure 5: Mistakes induced by expert-generated rules (green), SEARs (blue), and a combination of both (pink), with standard error bars.

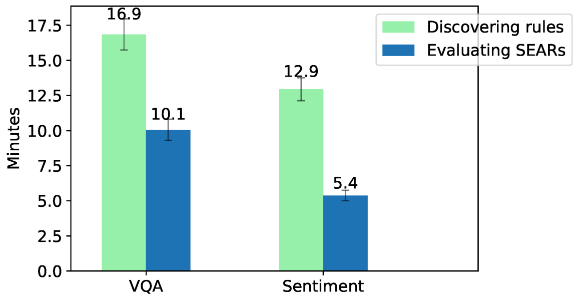


Figure 6: Time for users to create rules (green) and to evaluate SEARs (blue), with standard error bars

After creating their rules for VQA and sentiment analysis, the subjects evaluate 20 SEARs (one rule at a time) for each task, and accept only semantically equivalent rules. When a subject rejects a rule, we recompute the remaining set according to Eq (3) in real time. If a subject accepts more than 10 rules, only the first 10 are considered, in order to ensure a fair comparison against the expert-generated rules.

We compare expert-generated rules with accepted SEARs (each subject's rules are compared to the SEARs they accepted) in terms of the percentage of the correct predictions that "flip" when the rules are applied. This is what we asked the subjects to maximize, and all the rules were ones deemed to be semantic equivalent by the subjects themselves. We also consider the union of expert-generated rules and accepted SEARs. The results in Figure 5 show that on both datasets, the filtered SEARs induce a much higher rate of mistakes than the rules the subjects themselves created, with a small increase when the union of both sets is taken. Furthermore, subjects spent less time evaluating

	Error rate	
	Validation	Sensitivity
Visual QA		
Original Model	44.4.%	12.6%
SEAR Augmented	45.7 %	1.4%
Sentiment Analysis		
Original Model	22.1%	12.6%
SEAR Augmented	21.3%	3.4%

Table 6: **Fixing bugs using SEARs:** Effect of retraining models using SEARs, both on original validation and on sensitivity dataset. Retraining significantly reduces the number of bugs, with statistically insignificant changes to accuracy.

SEARs than trying to create their own rules (Figure 6). SEARs for sentiment analysis contain fewer POS tags, and are thus easier to evaluate for semantic equivalence than for VQA.

Discovering these bugs is hard for humans (even experts) without SEARs: not only do they need to imagine rules that maintain semantic equivalence, they must also discover the model's weak spots. Making good use of POS tags is also a challenge: only 50% of subjects attempt rules with POS tags for VQA, 36% for sentiment analysis.

Experts accepted 8.69 rules (on average) out of 20 for VQA as semantically equivalent, and 17.32 out of 20 for sentiment analysis. Similar to the previous experiment, errors made by the semantic scorer lead to rules that are not semantically equivalent (e.g. Table 7). With minimal human intervention, however, SEARs vastly outperform human experts in finding impactful bugs.

5.4 Fixing bugs using SEARs

Once such bugs are discovered, it is natural to want to fix them. The global and deterministic nature of SEARs make them actionable, as they represent bugs in a systematic manner. Once impactful bugs are identified, we use a simple data augmentation procedure: applying SEARs to the training data, and retraining the model on the original training augmented with the generated examples.

We take the rules that are accepted by ≥ 20 subjects as accepted bugs, a total of 4 rules (in Table 2) for VQA, and 16 rules for sentiment (including ones in Table 3). We then augment the training data by applying these rules to it, and retrain the models. To check if the bugs are still present, we create a sensitivity dataset by applying these SEARs to instances predicted correctly on the validation. A model not prone to the bugs described by these rules should not change any of its predictions, and should thus have error rate 0% on this sensitivity data. We also measure accuracy on the original validation data, to make sure that our bug-fixing procedure is not decreasing accuracy.

Table 6 shows that the incidence of these errors is greatly reduced after augmentation, with negligible changes to the validation accuracy (on both tasks, the changes are consistent with the effect of retraining with different seeds). These results show that SEARs are useful not only for discovering bugs, but are also actionable through a simple augmentation technique for any model.

6 Related Work

Previous work on debugging primarily focuses on *explaining* predictions in validation data in order to uncover bugs (Ribeiro et al., 2016, 2018; Kulesza et al., 2011), or find labeling errors (Zhang et al., 2018; Koh and Liang, 2017). Our work is complementary to these techniques, as they provide no mechanism to detect oversensitivity bugs. We are able to uncover these bugs even when they are not present in the data, since we generate sentences.

Adversarial examples for image recognition are typically indistinguishable to the human eye (Szegedy et al., 2014). These are more of a security concern than bugs per se, as images with adversarial noise are not "natural", and not expected to occur in the real world outside of targeted attacks. Adversaries are usually specific to predictions, and even universal adversarial perturbations (Moosavi-Dezfooli et al., 2017) are not natural, semantically meaningful to humans, or actionable. "Imperceptible" adversarial noise does not carry over from images to text, as adding or changing a single word in a sentence can drastically alter its meaning. Jia and Liang (2017) recognize that a true analog to detect oversensitivity would need semantic-preserving perturbations, but do not pursue an automated solution due to the difficulty of paraphrase generation. Their adversaries are whole sentence concatenations, generated by manually defined rules tailored to reading comprehension, and each adversary is specific to an individual instance. Zhao et al. (2018) generate natural text adversaries by projecting the input data to a latent space using a generative adversarial networks (GANs), and searching for adversaries close to the original instance in this latent space. Apart from the challenge of training GANs to generate high quality text, there is no guarantee that an example close in the latent space is semantically equivalent. Ebrahimi et al. (2018), along with proposing character-level changes that are not semantic-preserving, also propose a heuristic that replaces single words adversarially to preserve semantics. This approach not only depends on having white-box access to the model, but is also not able to generate many adversaries (only $\sim 1.6\%$ for sentiment analysis, compare to $\sim 33\%$ for SEAs in Table 4b). Developed concurrently with our work, Iyyer et al. (2018) proposes a neural paraphrase model based on back-translated data, which is able to produce paraphrases that have different sentence structures from the original. They use paraphrases to generate adversaries and try to post-process nonsensical outputs, but they do not explicitly reject non-semantics preserving ones, nor do they try to induce rules from individual adversaries. In any case, their adversaries are also useful for data augmentation, in experiments similar to ours.

In summary, previous work on text adversaries change semantics, only generate local (instance-specific) adversaries (Zhao et al., 2018; Iyyer et al., 2018), or are tailored for white-box models (Ebrahimi et al., 2018) or specific tasks (Jia and Liang, 2017). In contrast, SEAs expose oversensitivity for specific predictions of black-box models for a variety of tasks, while SEARs are intuitive and actionable global rules that induce a high number of high-quality adversaries. To our knowledge, we are also the first to evaluate human performance in adversarial generation (semantics-preserving or otherwise), and our extensive evaluation shows that SEAs and SEARs detect individual bugs and general patterns better than humans can.

7 Limitations and Future Work

Having demonstrated the usefulness of SEAs and SEARs in a variety of domains, we now describe their limitations and opportunities for future work.

Semantic scoring errors: Paraphrasing is still an active area of research, and thus our semantic scorer is sometimes incorrect in evaluating rules for semantic equivalence. We show examples of SEARs that are rejected by users in Table 7 – the semantic scorer does not sufficiently penalize preposition changes, and is biased towards common terms. The presence of such errors is why we still need humans in the loop to accept or reject SEARs.

SEAR	**Questions / SEAs**	**f(x)**
on →in	What is ~~on~~ in the background?	~~A building~~ Mountains
	What is ~~on~~? in	~~Lights~~ The television
`VBP`→is	Where ~~are~~ is the water bottles	~~Table~~ Vending Maching
	Where ~~are~~ is the people gathered	~~living room~~ kitchen
`VERB` on → `VERB`	What is ~~on~~ the background?	~~A building~~ Mountains
	What are the planes parked ~~on~~?	~~Concrete~~ landing strip

Table 7: SEARs for VQA that are rejected by users

Other paraphrase limitations: Paraphrase models based on neural machine translation are biased towards maintaining the sentence structure, and thus do not produce certain adversaries (e.g. Table 5b), which recent work on paraphrasing (Iyyer et al., 2018) or generation using GANs (Zhao et al., 2018) may address. More critically, existing models are inaccurate for long texts, restricting SEAs and SEARs to sentences.

Better bug fixing: Our data augmentation has the human users accept/reject rules based on whether or not they preserve semantics. Developing more effective ways of leveraging the expert's time to close the loop, and facilitating more interactive collaboration between humans and SEARs are exciting areas for future work.

8 Conclusion

We introduced SEAs and SEARs – adversarial examples and rules that preserve semantics, while causing models to make mistakes. We presented examples of such bugs discovered in state-of-the-art models for various tasks, and demonstrated via user studies that non-experts and experts alike are much better at detecting local and global bugs in NLP models by using our methods. We also *close the loop* by proposing a simple data augmentation solution that greatly reduced oversensitivity while maintaining accuracy. We demonstrated that SEAs and SEARs can be an invaluable tool for debugging NLP models, while indicating their current limitations and avenues for future work.

Acknowledgements

We are grateful to Dan Weld, Robert L. Logan IV, and to the anonymous reviewers for their feedback. This work was supported in part by ONR award #N00014-13-1-0023, in part by NSF award #IIS-1756023, and in part by funding from FICO. The views expressed are of the authors and do not reflect the policy or opinion of the funding agencies.

References

Aayush Bansal, Ali Farhadi, and Devi Parikh. 2014. Towards transparent systems: Semantic characterization of failure modes. In *European Conference on Computer Vision (ECCV)*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Deijing Dou. 2018. HotFlip: White-Box Adversarial Examples for NLP. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Matt A Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *North American Association for Computational Linguistics (NAACL)*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* .

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*.

Dimitrios Kotzias, Misha Denil, Nando de Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Knowledge Discovery and Data Mining (KDD)*.

Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*.

Todd Kulesza, Simone Stumpf, Weng-Keen Wong, Margaret M. Burnett, Stephen Perona, Andrew Jensen Ko, and Ian Oberst. 2011. Why-oriented end-user debugging of naive bayes text classification. *TiiS* 1:2:1–2:31.

Mirella Lapata, Rico Sennrich, and Jonathan Mallinson. 2017. Paraphrasing revisited with neural machine translation. In *European Chapter of the ACL (EACL)*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning (ICML)*.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016. Understanding neural networks through representation erasure. *CoRR* abs/1612.08220.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* .

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kayur Patel, James Fogarty, James A. Landay, and Beverly Harrison. 2008. Investigating statistical machine learning as a tool for software development. In *Human Factors in Computing Systems (CHI)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Knowledge Discovery and Data Mining (KDD)*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *International Conference on Language Resources and Evaluation (LREC)*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xuezhou Zhang, Xiaojin Zhu, and Stephen Wright. 2018. Training set debugging using trusted items. In *AAAI Conference on Artificial Intelligence*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7W: Grounded Question Answering in Images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.