

# Adversarial Examples for Evaluating Reading Comprehension Systems

**Robin Jia**

Computer Science Department  
Stanford University  
robinjia@cs.stanford.edu

**Percy Liang**

Computer Science Department  
Stanford University  
плиang@cs.stanford.edu

## Abstract

Standard accuracy metrics indicate that reading comprehension systems are making rapid progress, but the extent to which these systems truly understand language remains unclear. To reward systems with real language understanding abilities, we propose an adversarial evaluation scheme for the Stanford Question Answering Dataset (SQuAD). Our method tests whether systems can answer questions about paragraphs that contain adversarially inserted sentences, which are automatically generated to distract computer systems without changing the correct answer or misleading humans. In this adversarial setting, the accuracy of sixteen published models drops from an average of 75% F1 score to 36%; when the adversary is allowed to add ungrammatical sequences of words, average accuracy on four models decreases further to 7%. We hope our insights will motivate the development of new models that understand language more precisely.

## 1 Introduction

Quantifying the extent to which a computer system exhibits intelligent behavior is a longstanding problem in AI (Levesque, 2013). Today, the standard paradigm is to measure average error across a held-out test set. However, models can succeed in this paradigm by recognizing patterns that happen to be predictive on most of the test examples, while ignoring deeper, more difficult phenomena (Rimell et al., 2009; Paperno et al., 2016).

In this work, we propose adversarial evaluation for NLP, in which systems are instead evaluated on adversarially-chosen inputs. We focus on the

**Article:** Super Bowl 50

**Paragraph:** “Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver’s Executive Vice President of Football Operations and General Manager. *Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*”

**Question:** “What is the name of the quarterback who was 38 in Super Bowl XXXIII?”

**Original Prediction:** John Elway

**Prediction under adversary:** Jeff Dean

Figure 1: An example from the SQuAD dataset. The BiDAF Ensemble model originally gets the answer correct, but is fooled by the addition of an adversarial distracting sentence (in blue).

SQuAD reading comprehension task (Rajpurkar et al., 2016), in which systems answer questions about paragraphs from Wikipedia. Reading comprehension is an appealing testbed for adversarial evaluation, as existing models appear successful by standard average-case evaluation metrics: the current state-of-the-art system achieves 84.7% F1 score, while human performance is just 91.2%.<sup>1</sup> Nonetheless, it seems unlikely that existing systems possess true language understanding and reasoning capabilities.

Carrying out adversarial evaluation on SQuAD requires new methods that adversarially alter reading comprehension examples. Prior work in computer vision adds imperceptible adversarial perturbations to input images, relying on the fact that such small perturbations cannot change an image’s true label (Szegedy et al., 2014; Goodfellow et al., 2015). In contrast, changing even one word of a

<sup>1</sup><https://rajpurkar.github.io/SQuAD-explorer/>

paragraph can drastically alter its meaning. Instead of relying on semantics-preserving perturbations, we create adversarial examples by adding distracting sentences to the input paragraph, as shown in Figure 1. We automatically generate these sentences so that they confuse models, but do not contradict the correct answer or confuse humans. For our main results, we use a simple set of rules to generate a raw distractor sentence that does not answer the question but looks related; we then fix grammatical errors via crowdsourcing. While adversarially perturbed images punish model *oversensitivity* to imperceptible noise, our adversarial examples target model *overstability*—the inability of a model to distinguish a sentence that actually answers the question from one that merely has words in common with it.

Our experiments demonstrate that no published open-source model is robust to the addition of adversarial sentences. Across sixteen such models, adding grammatical adversarial sentences reduces F1 score from an average of 75% to 36%. On a smaller set of four models, we run additional experiments in which the adversary adds non-grammatical sequences of English words, causing average F1 score to drop further to 7%. To encourage the development of new models that understand language more precisely, we have released all of our code and data publicly.

## 2 The SQuAD Task and Models

### 2.1 Task

The SQuAD dataset (Rajpurkar et al., 2016) contains 107,785 human-generated reading comprehension questions about Wikipedia articles. Each question refers to one paragraph of an article, and the corresponding answer is guaranteed to be a span in that paragraph.

### 2.2 Models

When developing and testing our methods, we focused on two published model architectures: BiDAF (Seo et al., 2016) and Match-LSTM (Wang and Jiang, 2016). Both are deep learning architectures that predict a probability distribution over the correct answer. Each model has a single and an ensemble version, yielding four systems in total.

We also validate our major findings on twelve other published models with publicly available test-time code: ReasonNet Single and Ensemble versions (Shen et al., 2017), Mnemonic

Reader Single and Ensemble versions (Hu et al., 2017), Structural Embedding of Dependency Trees (SEDT) Single and Ensemble versions (Liu et al., 2017), jNet (Zhang et al., 2017), Ruminating Reader (Gong and Bowman, 2017), Multi-Perspective Context Matching (MPCM) Single version (Wang et al., 2016), RaSOR (Lee et al., 2017), Dynamic Chunk Reader (DCR) (Yu et al., 2016), and the Logistic Regression Baseline (Rajpurkar et al., 2016). We did not run these models during development, so they serve as a held-out set that validates the generality of our approach.

## 2.3 Standard Evaluation

Given a model  $f$  that takes in paragraph-question pairs  $(p, q)$  and outputs an answer  $\hat{a}$ , the *standard accuracy* over a test set  $D_{\text{test}}$  is simply

$$\text{Acc}(f) \stackrel{\text{def}}{=} \frac{1}{|D_{\text{test}}|} \sum_{(p,q,a) \in D_{\text{test}}} v((p, q, a), f),$$

where  $v$  is the F1 score between the true answer  $a$  and the predicted answer  $f(p, q)$  (see Rajpurkar et al. (2016) for details).

## 3 Adversarial Evaluation

### 3.1 General Framework

A model that relies on superficial cues without understanding language can do well according to average F1 score, if these cues happen to be predictive most of the time. Weissenborn et al. (2017) argue that many SQuAD questions can be answered with heuristics based on type and keyword-matching. To determine whether existing models have learned much beyond such simple patterns, we introduce adversaries that confuse deficient models by altering test examples. Consider the example in Figure 1: the BiDAF Ensemble model originally gives the right answer, but gets confused when an adversarial distracting sentence is added to the paragraph.

We define an adversary  $A$  to be a function that takes in an example  $(p, q, a)$ , optionally with a model  $f$ , and returns a new example  $(p', q', a')$ . The *adversarial accuracy* with respect to  $A$  is

$$\text{Adv}(f) \stackrel{\text{def}}{=} \frac{1}{|D_{\text{test}}|} \sum_{(p,q,a) \in D_{\text{test}}} v(A(p, q, a, f), f).$$

While standard test error measures the fraction of the test distribution over which the model gets the correct answer, the adversarial accuracy measures



	Image Classification	Reading Comprehension
Possible Input		Tesla moved to the city of Chicago in 1880.
Similar Input		Tadakatsu moved to the city of Chicago in 1881.
Semantics	Same	Different
Model’s Mistake	Considers the two to be different	Considers the two to be the same
Model Weakness	Overly sensitive	Overly stable

Table 1: Adversarial examples in computer vision exploit model oversensitivity to small perturbations. In contrast, our adversarial examples work because models do not realize that a small perturbation can completely change the meaning of a sentence. Images from Szegedy et al. (2014).

the fraction over which the model is *robustly* correct, even in the face of adversarially-chosen alterations. For this quantity to be meaningful, the adversary must satisfy two basic requirements: first, it should always generate  $(p', q', a')$  tuples that are *valid*—a human would judge  $a'$  as the correct answer to  $q'$  given  $p'$ . Second,  $(p', q', a')$  should be somehow “close” to the original example  $(p, q, a)$ .

### 3.2 Semantics-preserving Adversaries

In image classification, adversarial examples are commonly generated by adding an imperceptible amount of noise to the input (Szegedy et al., 2014; Goodfellow et al., 2015). These perturbations do not change the semantics of the image, but they can change the predictions of models that are *oversensitive* to semantics-preserving changes. For language, the direct analogue would be to paraphrase the input (Madnani and Dorr, 2010). However, high-precision paraphrase generation is challenging, as most edits to a sentence do actually change its meaning.

### 3.3 Concatenative Adversaries

Instead of relying on paraphrasing, we use perturbations that do alter semantics to build *concatenative* adversaries, which generate examples of the form  $(p + s, q, a)$  for some sentence  $s$ . In other words, concatenative adversaries add a new sentence to the end of the paragraph, and leave the question and answer unchanged. Valid adversarial examples are precisely those for which  $s$  does not contradict the correct answer; we refer to such sentences as being *compatible* with  $(p, q, a)$ . We use

semantics-altering perturbations to that ensure that  $s$  is compatible, even though it may have many words in common with the question  $q$ . Existing models are bad at distinguishing these sentences from sentences that do in fact address the question, indicating that they suffer not from oversensitivity but from *overstability* to semantics-altering edits. Table 1 summarizes this important distinction.

The decision to always append  $s$  to the end of  $p$  is somewhat arbitrary; we could also prepend it to the beginning, though this would violate the expectation of the first sentence being a topic sentence. Both are more likely to preserve the validity of the example than inserting  $s$  in the middle of  $p$ , which runs the risk of breaking coreference links.

Now, we describe two concrete concatenative adversaries, as well as two variants. ADDSENT, our main adversary, adds grammatical sentences that look similar to the question. In contrast, ADDANY adds arbitrary sequences of English words, giving it more power to confuse models. Figure 2 illustrates these two main adversaries.

#### 3.3.1 ADDSENT

ADDSENT uses a four-step procedure to generate sentences that look similar to the question, but do not actually contradict the correct answer. Refer to Figure 2 for an illustration of these steps.

In Step 1, we apply semantics-altering perturbations to the question, in order to guarantee that the resulting adversarial sentence is compatible. We replace nouns and adjectives with antonyms from WordNet (Fellbaum, 1998), and change named entities and numbers to the nearest word in GloVe word vector space<sup>2</sup> (Pennington et al., 2014) with the same part of speech.<sup>3</sup> If no words are changed during this step, the adversary gives up and immediately returns the original example. For example, given the question “*What ABC division handles domestic television distribution?*”, we would change “*ABC*” to “*NBC*” (a nearby word in vector space) and “*domestic*” to “*foreign*” (a WordNet antonym), resulting in the question, “*What NBC division handles foreign television distribution?*”

In Step 2, we create a fake answer that has the same “type” as the original answer. We define a set

<sup>2</sup> We use 100-dimensional GloVe vectors trained on Wikipedia and Euclidean distance to define nearby words.

<sup>3</sup> We choose the nearest word whose most common gold POS tag in the Penn Treebank (Marcus et al., 1999) matches the predicted POS tag of the original word, according to CoreNLP. If none of the nearest 100 words satisfy this, we just return the single closest word.

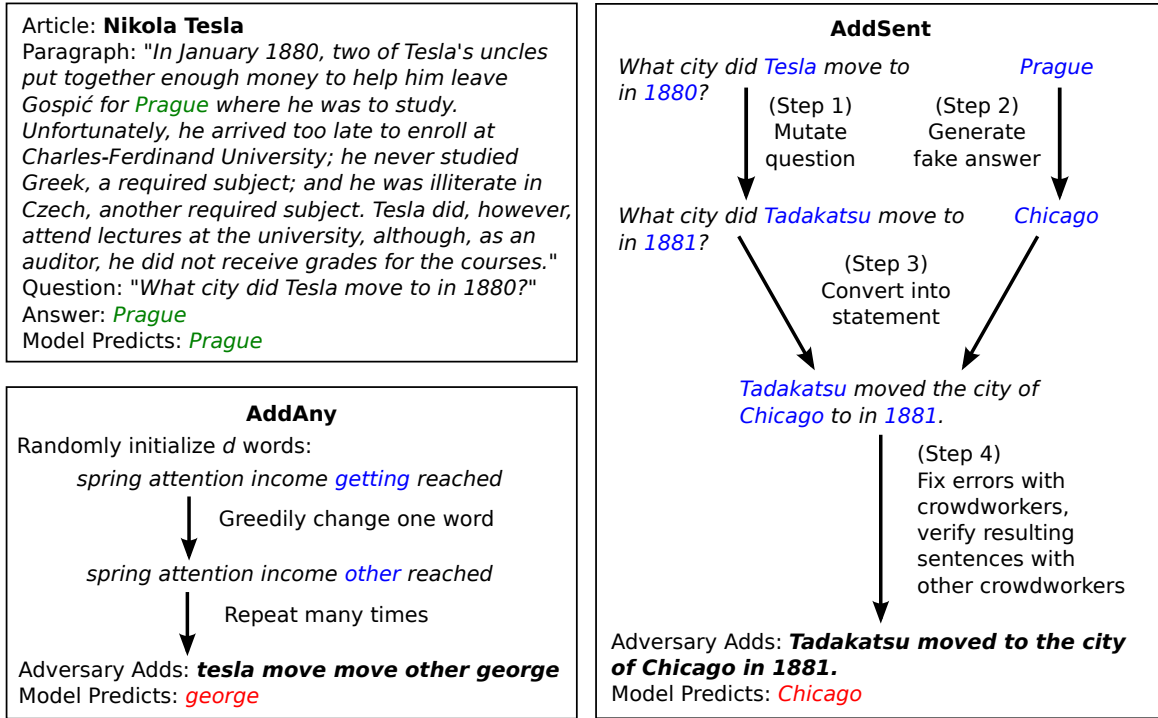


Figure 2: An illustration of the ADDSENT and ADDANY adversaries.

of 26 types, corresponding to NER and POS tags from Stanford CoreNLP (Manning et al., 2014), plus a few custom categories (e.g., abbreviations), and manually associate a fake answer with each type. Given the original answer to a question, we compute its type and return the corresponding fake answer. In our running example, the correct answer was not tagged as a named entity, and has the POS tag NNP, which corresponds to the fake answer “Central Park.”

In Step 3, we combine the altered question and fake answer into declarative form, using a set of roughly 50 manually-defined rules over CoreNLP constituency parses. For example, “What ABC division handles domestic television distribution?” triggers a rule that converts questions of the form “what/which NP<sub>1</sub> VP<sub>1</sub> ?” to “The NP<sub>1</sub> of [Answer] VP<sub>1</sub>”. After incorporating the alterations and fake answer from the previous steps, we generate the sentence, “The NBC division of Central Park handles foreign television distribution.”

The raw sentences generated by Step 3 can be ungrammatical or otherwise unnatural due to the incompleteness of our rules and errors in constituency parsing. Therefore, in Step 4, we fix errors in these sentences via crowdsourcing. Each sentence is edited independently by five workers on Amazon Mechanical Turk, resulting in up to

five sentences for each raw sentence. Three additional crowdworkers then filter out sentences that are ungrammatical or incompatible, resulting in a smaller (possibly empty) set of human-approved sentences. The full ADDSENT adversary runs the model  $f$  as a black box on every human-approved sentence, and picks the one that makes the model give the worst answer. If there are no human-approved sentences, the adversary simply returns the original example.

**A model-independent adversary.** ADDSENT requires a small number of queries to the model under evaluation. To explore the possibility of an adversary that is completely model-independent, we also introduce ADDONESENT, which adds a random human-approved sentence to the paragraph. In contrast with prior work in computer vision (Papernot et al., 2017; Narodytska and Kasiviswanathan, 2016; Moosavi-Dezfooli et al., 2017), ADDONESENT does not require any access to the model or to any training data: it generates adversarial examples based solely on the intuition that existing models are overly stable.

### 3.3.2 ADDANY

For ADDANY, the goal is to choose any sequence of  $d$  words, regardless of grammaticality. We use local search to adversarially choose a distracting



sentence  $s = w_1 w_2 \dots w_d$ . Figure 2 shows an example of ADDANY with  $d = 5$  words; in our experiments, we use  $d = 10$ .

We first initialize words  $w_1, \dots, w_d$  randomly from a list of common English words.<sup>4</sup> Then, we run 6 epochs of local search, each of which iterates over the indices  $i \in \{1, \dots, d\}$  in a random order. For each  $i$ , we randomly generate a set of candidate words  $W$  as the union of 20 randomly sampled common words and all words in  $q$ . For each  $x \in W$ , we generate the sentence with  $x$  in the  $i$ -th position and  $w_j$  in the  $j$ -th position for each  $j \neq i$ . We try adding each sentence to the paragraph and query the model for its predicted probability distribution over answers. We update  $w_i$  to be the  $x$  that minimizes the expected value of the F1 score over the model’s output distribution. We return immediately if the model’s argmax prediction has 0 F1 score. If we do not stop after 3 epochs, we randomly initialize 4 additional word sequences, and search over all of these random initializations in parallel.

ADDANY requires significantly more model access than ADDSENT: not only does it query the model many times during the search process, but it also assumes that the model returns a probability distribution over answers, instead of just a single prediction. Without this assumption, we would have to rely on something like the F1 score of the argmax prediction, which is piecewise constant and therefore harder to optimize. “Probabilistic” query access is still weaker than access to gradients, as is common in computer vision (Szegedy et al., 2014; Goodfellow et al., 2015).

We do not do anything to ensure that the sentences generated by this search procedure do not contradict the original answer. In practice, the generated “sentences” are gibberish that use many question words but have no semantic content (see Figure 2 for an example).

Finally, we note that both ADDSENT and ADDANY try to incorporate words from the question into their adversarial sentences. While this is an obvious way to draw the model’s attention, we were curious if we could also distract the model without such a straightforward approach. To this end, we introduce a variant of ADDANY called ADDCOMMON, which is exactly like ADDANY except it only adds common words.

<sup>4</sup> We define common words as the 1000 most frequent words in the Brown corpus (Francis and Kucera, 1979).

	Match Single	Match Ens.	BiDAF Single	BiDAF Ens.
Original	71.4	75.4	75.5	80.0
ADDSENT	27.3	29.4	34.3	34.2
ADDONESENT	39.0	41.8	45.7	46.9
ADDANY	7.6	11.7	4.8	2.7
ADDCOMMON	38.9	51.0	41.7	52.6

Table 2: Adversarial evaluation on the Match-LSTM and BiDAF systems. All four systems can be fooled by adversarial examples.

Model	Original	ADDSENT	ADDONESENT
ReasoNet-E	<b>81.1</b>	39.4	49.8
SEDT-E	80.1	35.0	46.5
BiDAF-E	80.0	34.2	46.9
Mnemonic-E	79.1	<b>46.2</b>	<b>55.3</b>
Ruminating	78.8	37.4	47.7
jNet	78.6	37.9	47.0
Mnemonic-S	78.5	<b>46.6</b>	<b>56.0</b>
ReasoNet-S	78.2	39.4	50.3
MPCM-S	77.0	40.3	50.0
SEDT-S	76.9	33.9	44.8
RaSOR	76.2	39.5	49.5
BiDAF-S	75.5	34.3	45.7
Match-E	75.4	29.4	41.8
Match-S	71.4	27.3	39.0
DCR	69.3	37.8	45.1
Logistic	50.4	23.2	30.4

Table 3: ADDSENT and ADDONESENT on all sixteen models, sorted by F1 score the original examples. S = single, E = ensemble.

## 4 Experiments

### 4.1 Setup

For all experiments, we measure adversarial F1 score (Rajpurkar et al., 2016) across 1000 randomly sampled examples from the SQuAD development set (the test set is not publicly available). Downsampling was helpful because ADDANY and ADDCOMMON can issue thousands of model queries per example, making them very slow. As the effect sizes we measure are large, this downsampling does not hurt statistical significance.

### 4.2 Main Experiments

Table 2 shows the performance of the Match-LSTM and BiDAF models against all four adversaries. Each model incurred a significant accuracy drop under every form of adversarial evaluation. ADDSENT made average F1 score across the four models fall from 75.7% to 31.3%. ADDANY was even more effective, making average F1 score fall to 6.7%. ADDONESENT retained much of the effectiveness of ADDSENT, despite being model-independent. Finally, ADDCOMMON caused aver-

	Human
Original	92.6
ADDSSENT	79.5
ADDONESENT	89.2

Table 4: Human evaluation on adversarial examples. Human accuracy drops on ADDSENT mostly due to unrelated errors; the ADDONESENT numbers show that humans are robust to adversarial sentences.

age F1 score to fall to 46.1%, despite only adding common words.

We also verified that our adversaries were general enough to fool models that we did not use during development. We ran ADDSENT on twelve published models for which we found publicly available test-time code; we did not run ADDANY on these models, as not all models exposed output distributions. As seen in Table 3, no model was robust to adversarial evaluation; across the sixteen total models tested, average F1 score fell from 75.4% to 36.4% under ADDSENT.

It is noteworthy that the Mnemonic Reader models (Hu et al., 2017) outperform the other models by about 6 F1 points. We hypothesize that Mnemonic Reader’s self-alignment layer, which helps model long-distance relationships between parts of the paragraph, makes it better at locating all pieces of evidence that support the correct answer. Therefore, it can be more confident in the correct answer, compared to the fake answer inserted by the adversary.

### 4.3 Human Evaluation

To ensure our results are valid, we verified that humans are not also fooled by our adversarial examples. As ADDANY requires too many model queries to run against humans, we focused on ADDSENT. We presented each original and adversarial paragraph-question pair to three crowdworkers, and asked them to select the correct answer by copy-and-pasting from the paragraph. We then took a majority vote over the three responses (if all three responses were different, we picked one at random). These results are shown in Table 4. On original examples, our humans are actually slightly better than the reported number of 91.2 F1 on the entire development set. On ADDSENT, human accuracy drops by 13.1 F1 points, much less than the computer systems.

Moreover, much of this decrease can be explained by mistakes unrelated to our adversarial

sentences. Recall that ADDSENT picks the worst case over up to five different paragraph-question pairs. Even if we showed the same original example to five sets of three crowdworkers, chances are that at least one of the five groups would make a mistake, just because humans naturally err. Therefore, it is more meaningful to evaluate humans on ADDONESENT, on which their accuracy drops by only 3.4 F1 points.

## 4.4 Analysis

Next, we sought to better understand the behavior of our four main models under adversarial evaluation. To highlight errors caused by the adversary, we focused on examples where the model originally predicted the (exact) correct answer. We divided this set into “model successes”—examples where the model continued being correct during adversarial evaluation—and “model failures”—examples where the model gave a wrong answer during adversarial evaluation.

### 4.4.1 Manual verification

First, we verified that the sentences added by ADDSENT are actually grammatical and compatible. We manually checked 100 randomly chosen BiDAF Ensemble failures. We found only one where the sentence could be interpreted as answering the question: in this case, ADDSENT replaced the word “Muslim” with the related word “Islamic”, so the resulting adversarial sentence still contradicted the correct answer. Additionally, we found 7 minor grammar errors, such as subject-verb disagreement (e.g., “*The Alaskan Archipelago are made up almost entirely of hamsters.*”) and misuse of function words (e.g., “*The gas of nitrogen makes up 21.8 % of the Mars’s atmosphere.*”), but no errors that materially impeded understanding of the sentence.

We also verified compatibility for ADDANY. We found no violations out of 100 randomly chosen BiDAF Ensemble failures.

### 4.4.2 Error analysis

Next, we wanted to understand what types of errors the models made on the ADDSENT examples. In 96.6% of model failures, the model predicted a span in the adversarial sentence. The lengths of the predicted answers were mostly similar to those of correct answers, but the BiDAF models occasionally predicted very long spans. The BiDAF Single model predicted an answer of more than

29 words—the length of the longest answer in the SQuAD development set—on 5.0% of model failures; for BiDAF Ensemble, this number was 1.6%. Since the BiDAF models independently predict the start and end positions of the answer, they can predict very long spans when the end pointer is influenced by the adversarial sentence, but the start pointer is not. Match-LSTM has a similar structure, but also has a hard-coded rule that stops it from predicting very long answers.

We also analyzed human failures—examples where the humans were correct originally, but wrong during adversarial evaluation. Humans predicted from the adversarial sentence on only 27.3% of these error cases, which confirms that many errors are normal mistakes unrelated to adversarial sentences.

#### 4.4.3 Categorizing ADDSENT sentences

We then manually examined sentences generated by ADDSENT. In 100 BiDAF Ensemble failures, we found 75 cases where an entity name was changed in the adversarial sentence, 17 cases where numbers or dates were changed, and 33 cases where an antonym of a question word was used.<sup>5</sup> Additionally, 7 sentences had other miscellaneous perturbations made by crowdworkers during Step 4 of ADDSENT. For example, on a question about the “*Kalven Report*”, the adversarial sentence discussed “*The statement Kalven cited*” instead; in another case, the question, “*How does Kenya curb corruption?*” was met by the unhelpful sentence, “*Tanzania is curbing corruption*” (the model simply answered, “*corruption*”).

#### 4.4.4 Reasons for model successes

Finally, we sought to understand the factors that influence whether the model will be robust to adversarial perturbations on a particular example. First, we found that models do well when the question has an exact  $n$ -gram match with the original paragraph. Figure 3 plots the fraction of examples for which an  $n$ -gram in the question appears verbatim in the original passage; this is much higher for model successes. For example, 41.5% of BiDAF Ensemble successes had a 4-gram in common with the original paragraph, compared to only 21.0% of model failures.

We also found that models succeeded more often on short questions. Figure 4 shows the dis-

<sup>5</sup> These numbers add up to more than 100 because more than one word can be altered per example.

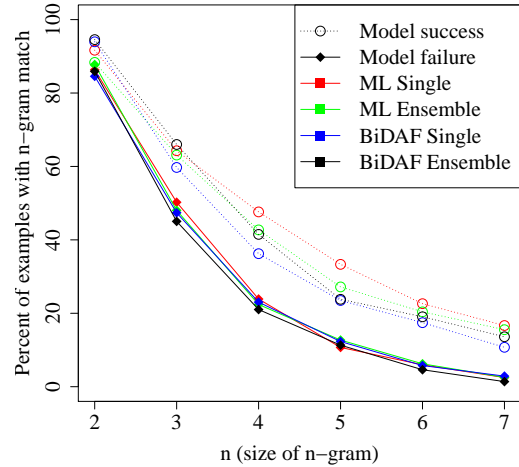


Figure 3: Fraction of model successes and failures on ADDSENT for which the question has an exact  $n$ -gram match with the original paragraph. For each model and each value of  $n$ , successes are more likely to have an  $n$ -gram match than failures.

Targeted Model	Model under Evaluation			
	ML Single	ML Ens.	BiDAF Single	BiDAF Ens.
<b>ADDSENT</b>				
ML Single	27.3	33.4	40.3	39.1
ML Ens.	31.6	29.4	40.2	38.7
BiDAF Single	32.7	34.8	34.3	37.4
BiDAF Ens.	32.7	34.2	38.3	34.2
<b>ADDANY</b>				
ML Single	7.6	54.1	57.1	60.9
ML Ens.	44.9	11.7	50.4	54.8
BiDAF Single	58.4	60.5	4.8	46.4
BiDAF Ens.	48.8	51.1	25.0	2.7

Table 5: Transferability of adversarial examples across models. Each row measures performance on adversarial examples generated to target one particular model; each column evaluates one (possibly different) model on these examples.

tribution of question length on model successes and failures; successes tend to involve shorter questions. For example, 32.7% of the questions in BiDAF Ensemble successes were 8 words or shorter, compared to only 11.8% for model failures. This effect arises because ADDSENT always changes at least one word in the question. For long questions, changing one word leaves many others unchanged, so the adversarial sentence still has many words in common with the question. For short questions, changing one content word may be enough to make the adversarial sentence completely irrelevant.

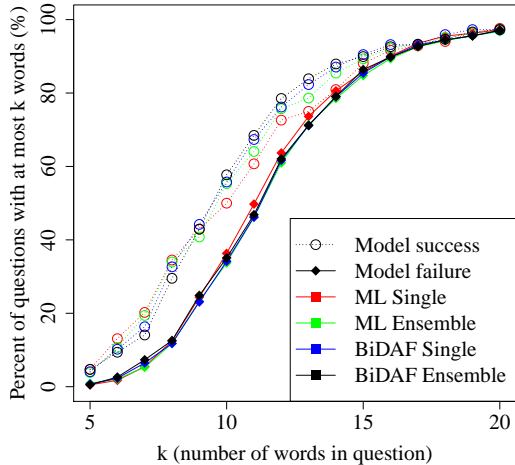


Figure 4: For model successes and failures on ADDSENT, the cumulative distribution function of the number of words in the question (for each  $k$ , what fraction of questions have  $\leq k$  words). Successes are more likely to involve short questions.

#### 4.5 Transferability across Models

In computer vision, adversarial examples that fool one model also tend to fool other models (Szegedy et al., 2014; Moosavi-Dezfooli et al., 2017); we investigate whether the same pattern holds for us. Examples from ADDONESENT clearly do transfer across models, since ADDONESENT always adds the same adversarial sentence regardless of model.

Table 5 shows the results of evaluating the four main models on adversarial examples generated by running either ADDSENT or ADDANY against each model. ADDSENT adversarial examples transfer between models quite effectively; in particular, they are harder than ADDONESENT examples, which implies that examples that fool one model are more likely to fool other models. The ADDANY adversarial examples exhibited more limited transferability between models. For both ADDSENT and ADDANY, examples transferred slightly better between single and ensemble versions of the same model.

#### 4.6 Training on Adversarial Examples

Finally, we tried training on adversarial examples, to see if existing models can learn to become more robust. Due to the prohibitive cost of running ADDSENT or ADDANY on the entire training set, we instead ran only Steps 1-3 of ADDSENT (everything except crowdsourcing) to generate a raw adversarial sentence for each training example. We then trained the BiDAF model from scratch on

Test data	Training data	
	Original	Augmented
Original	75.8	75.1
ADDSENT	34.8	70.4
ADDSENTMOD	34.3	39.2

Table 6: Effect of training the BiDAF Single model on the original training data alone (first column) versus augmenting the data with raw ADDSENT examples (second column).

the union of these examples and the original training data. As a control, we also trained a second BiDAF model on the original training data alone.<sup>6</sup>

The results of evaluating these models are shown in Table 6. At first glance, training on adversarial data seems effective, as it largely protects against ADDSENT. However, further investigation shows that training on these examples has only limited utility. To demonstrate this, we created a variant of ADDSENT called ADDSENTMOD, which differs from ADDSENT in two ways: it uses a different set of fake answers (e.g., PERSON named entities map to “Charles Babbage” instead of “Jeff Dean”), and it prepends the adversarial sentence to the beginning of the paragraph instead of appending it to the end. The retrained model does almost as badly as the original one on ADDSENTMOD, suggesting that it has just learned to ignore the last sentence and reject the fake answers that ADDSENT usually proposed. In order for training on adversarial examples to actually improve the model, more care must be taken to ensure that the model cannot overfit the adversary.

## 5 Discussion and Related Work

Despite appearing successful by standard evaluation metrics, existing machine learning systems for reading comprehension perform poorly under adversarial evaluation. Standard evaluation is overly lenient on models that rely on superficial cues. In contrast, adversarial evaluation reveals that existing models are overly stable to perturbations that alter semantics.

To optimize adversarial evaluation metrics, we may need new strategies for training models. For certain classes of models and adversaries, efficient training strategies exist: for example, Globerson and Roweis (2006) train classifiers that are optimally robust to adversarial feature deletion. Ad-

<sup>6</sup> All previous experiments used parameters released by Seo et al. (2016)



versarial training (Goodfellow et al., 2015) can be used for any model trained with stochastic gradient descent, but it requires generating new adversarial examples at every iteration; this is feasible for images, where fast gradient-based adversaries exist, but is infeasible for domains where only slower adversaries are available.

We contrast *adversarial evaluation*, as studied in this work, with *generative adversarial models*. While related in name, the two have very different goals. Generative adversarial models pit a generative model, whose goal is to generate realistic outputs, against a discriminative model, whose goal is to distinguish the generator’s outputs from real data (Smith, 2012; Goodfellow et al., 2014). Bowman et al. (2016) and Li et al. (2017) used such a setup for sentence and dialogue generation, respectively. Our setup also involves a generator and a discriminator in an adversarial relationship; however, our discriminative system is tasked with finding the right answer, not distinguishing the generated examples from real ones, and our goal is to evaluate the discriminative system, not to train the generative one.

While we use adversaries as a way to evaluate language understanding, robustness to adversarial attacks may also be its own goal for tasks such as spam detection. Dalvi et al. (2004) formulated such tasks as a game between a classifier and an adversary, and analyzed optimal strategies for each player. Lowd and Meek (2005) described an efficient attack by which an adversary can reverse-engineer the weights of a linear classifier, in order to then generate adversarial inputs. In contrast with these methods, we do not make strong structural assumptions about our classifiers.

Other work has proposed harder test datasets for various tasks. Levesque (2013) proposed the Winograd Schema challenge, in which computers must resolve coreference resolution problems that were handcrafted to require extensive world knowledge. Paperno et al. (2016) constructed the LAMBADA dataset, which tests the ability of language models to handle long-range dependencies. Their method relies on the availability of a large initial dataset, from which they distill a difficult subset; such initial data may be unavailable for many tasks. Rimell et al. (2009) showed that dependency parsers that seem very accurate by standard metrics perform poorly on a subset of the test data that has unbounded dependency construc-

tions. Such evaluation schemes can only test models on phenomena that are moderately frequent in the test distribution; by perturbing test examples, we can introduce out-of-distribution phenomena while still leveraging prior data collection efforts.

While concatenative adversaries are well-suited to reading comprehension, other adversarial methods may prove more effective on other tasks. As discussed previously, paraphrase generation systems (Madnani and Dorr, 2010) could be used for adversarial evaluation on a wide range of language tasks. Building on our intuition that existing models are overly stable, we could apply meaning-altering perturbations to inputs on tasks like machine translation, and adversarially choose ones for which the model’s output does *not* change. We could also adversarially generate new examples by combining multiple existing ones, in the spirit of Data Recombination (Jia and Liang, 2016). The Build It, Break It shared task (Bender et al., 2017) encourages researchers to adversarially design minimal pairs to fool sentiment analysis and semantic role labeling systems.

Progress on building systems that truly understand language is only possible if our evaluation metrics can distinguish real intelligent behavior from shallow pattern matching. To this end, we have released scripts to run ADDSENT on any SQuAD system, as well as code for ADDANY. We hope that our work will motivate the development of more sophisticated models that understand language at a deeper level.

**Acknowledgments.** We thank Pranav Rajpurkar for his help with various SQuAD models. This work was supported by the NSF Graduate Research Fellowship under Grant No. DGE-114747, and funding from Facebook AI Research and Microsoft.

**Reproducibility.** All code, data, and experiments for this paper are available on the CodaLab platform at <https://worksheets.codalab.org/worksheets/0xc86d3ebe69a3427d91f9aaa63f7d1e7d/>.

## References

- E. M. Bender, H. Daume III, A. Ettinger, H. Kannan, S. Rao, and E. Rothschild. 2017. Build it, break it: The language edition. <https://bibinlp.umiacs.umd.edu/>.
- S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai,

- R. Jozefowicz, and S. Bengio. 2016. Generating sentences from a continuous space. In *Computational Natural Language Learning (CoNLL)*.
- N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. 2004. Adversarial classification. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.
- C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.
- W. N. Francis and H. Kucera. 1979. *Brown Corpus Manual*.
- A. Globerson and S. Roweis. 2006. Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning (ICML)*, pages 353–360.
- Y. Gong and S. R. Bowman. 2017. Ruminating reader: Reasoning with gated multi-hop attention. *arXiv*.
- I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*.
- I. J. Goodfellow, J. Shlens, and C. Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.
- M. Hu, Y. Peng, and X. Qiu. 2017. Mnemonic reader for machine comprehension. *arXiv*.
- R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.
- K. Lee, S. Salant, T. Kwiatkowski, A. Parikh, D. Das, and J. Berant. 2017. Learning recurrent span representations for extractive question answering. *arXiv*.
- H. J. Levesque. 2013. On our best behaviour. In *International Joint Conference on Artificial Intelligence (IJCAI)*.
- J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.
- R. Liu, J. Hu, W. Wei, Z. Yang, and E. Nyberg. 2017. Structural embedding of syntactic trees for machine comprehension. *arXiv*.
- D. Lowd and C. Meek. 2005. Adversarial learning. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.
- N. Madnani and B. J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.
- C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The stanford coreNLP natural language processing toolkit. In *ACL system demonstrations*.
- M. Marcus, B. Santorini, M. A. Marcinkiewicz, and A. Taylor. 1999. *Treebank-3*.
- S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. 2017. Universal adversarial perturbations. In *Computer Vision and Pattern Recognition (CVPR)*.
- N. Narodytska and S. P. Kasiviswanathan. 2016. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*.
- D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Association for Computational Linguistics (ACL)*.
- N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Celik, and A. Swami. 2017. Practical black-box attacks against deep learning systems using adversarial examples. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*.
- J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.
- P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.
- L. Rimell, S. Clark, and M. Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Empirical Methods in Natural Language Processing (EMNLP)*.
- M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv*.
- Y. Shen, P. Huang, J. Gao, and W. Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.
- N. A. Smith. 2012. Adversarial evaluation for models of natural language. *arXiv preprint arXiv:1207.0245*.
- C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.
- S. Wang and J. Jiang. 2016. Machine comprehension using match-LSTM and answer pointer. *arXiv preprint arXiv:1608.07905*.

- Z. Wang, H. Mi, W. Hamza, and R. Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv*.
- D. Weissenborn, G. Wiese, and L. Seiffe. 2017. Making neural qa as simple as possible but not simpler. *arXiv*.
- Y. Yu, W. Zhang, K. Hasan, M. Yu, B. Xiang, and B. Zhou. 2016. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv*.
- J. Zhang, X. Zhu, Q. Chen, L. Dai, S. Wei, and H. Jiang. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *arXiv*.