Second-Order NLP Adversarial Examples

John X. Morris
University of Virginia
jm8wx@virginia.edu

Abstract

Adversarial example generation methods in NLP rely on models like language models or sentence encoders to determine if potential adversarial examples are valid. In these methods, a valid adversarial example fools the model being attacked, and is determined to be semantically or syntactically valid by a second model. Research to date has counted all such examples as errors by the attacked model. We contend that these adversarial examples may not be flaws in the attacked model, but flaws in the model that determines validity. We term such invalid inputs second-order adversarial examples. We propose the constraint robustness curve, and associated metric ACCS, as tools for evaluating the robustness of a constraint to second-order adversarial examples. To generate this curve, we design an adversarial attack to run directly on the semantic similarity models. We test on two constraints, the Universal Sentence Encoder (USE) and BERTScore. Our findings indicate that such second-order examples exist, but are typically less common than first-order adversarial examples in state-of-the-art models. They also indicate that USE is effective as constraint on NLP adversarial examples, while BERTScore is nearly ineffectual. Code for running the experiments in this paper is available here.

1 Introduction

If an imperceptible change to an input causes a model to make a misclassification, the perturbed input is known as an adversarial example (Goodfellow et al., 2014). In domains with continuous inputs like audio and vision, whether such a change is considered "imperceptible" can be easily measured: A change to an image may be considered imperceptible (and thus a valid adversarial example) if the resulting image is no more than some fixed distance away in pixel space (Chakraborty et al., 2018).

Original
Premise: Two bearded men sit side by side.
Hypothesis: Two **men** are **sitting** down.
→ **Model Prediction: Entailment (99.8%)**

Perturbation
Premise: Two bearded men sit side by side.
Hypothesis: Two **women** are **standing** down.
→ **Model Prediction: Contradiction (99.9%)**

USE Cosine Similarity:
0.93 (Valid Adversarial Example)

Figure 1: A second-order adversarial example in NLP. Although the perturbation has different meaning than the original (and the entailment model correctly predicts a contradiction), the sentence encoding similarity does not reflect this change. Current NLP adversarial example generation methods would incorrectly consider this a flaw in the entailment model.

We refer to the function that determines imperceptibility as the constraint, C. For input x and perturbation x_{adv}, if $C(x, x_{adv})$ is true, x_{adv} is a valid perturbation for x.

Different domains call for different constraints. In vision, a common constraint is $\ell_\infty(x, x_{adv})$, the maximum pixel-wise distance between image x and its perturbation x_{adv} (Goodfellow et al., 2014). In audio, a common constraint is $|dB(x) - dB(x_{adv})|$, the distortion in decibels between audio input x and perturbation x_{adv} (Carlini and Wagner, 2018). Both constraints are easily computed, well-understood, and correlate with human perceptual distance.

Choosing the correct constraint is not always so straightforward. In discrete domains like language, there is no obvious choice. In fact, the field lacks consensus on even the meaning of "imperceptibility". Different adversarial attacks have used different definitions of imperceptibility (Zhang et al.,

2020a). One common definition (Alzantot et al., 2018; Jin et al., 2019; Ren et al., 2019; Garg and Ramakrishnan, 2020) is imperceptibility with respect to meaning: $C(x, x_{adv})$ is true if x_{adv} retains the semantics of x.

With this definition, a perturbation x_{adv} is determined to be a valid adversarial example if it simultaneously fools the model and retains the semantics of x. This formulation is problematic because measuring semantic similarity is an open problem in NLP. As a consequence, many adversarial attacks use a *second NLP model* as a constraint, to determine whether or not x_{adv} preserves the semantics of x.

Just like the model under attack, the semantic similarity model is vulnerable to adversarial examples. So when this type of attack finds a valid adversarial example, it is unclear which model has made a mistake: was it the model being attacked, or the model used to enforce the constraint?

In other words, it is possible that the semantic similarity model improperly classified x_{adv} as preserving the semantics of x. We refer to these flaws in constraints as **second-order adversarial examples**. Figure 1 shows a sample second-order adversarial example. Second-order adversarial examples have been largely ignored in the literature on NLP adversarial examples to date.

Now that we are aware of the existence of second-order adversarial examples, we seek to minimize their impact. How can we measure a given constraint's susceptibility to second-order adversarial examples? We suggest one such measurement tool: the **constraint robustness curve** and its associated metric ACCS.

We then develop an adversarial example generation technique for finding examples that fool these semantic similarity models. Our findings indicate that adversarial examples for these types of models exist, but are less likely than adversarial examples that fool other NLP models.

Along the way, we compare the Universal Sentence Encoder (USE) (Cer et al., 2018), a sentence encoder commonly used as a constraint for NLP adversarial examples, with BERTScore (Zhang et al., 2019), a metric that outperforms sentence encoders for evaluating text generation systems.

The main contributions of this work can be summarized as follows:

1. We formally define second-order adversarial examples, a previously unaddressed issue with the problem statement for semantics-preserving adversarial example generation in NLP.

2. We propose Adjusted Constraint C-Statistic (ACCS), the normalized area under the constraint robustness curve, as a measurement of the efficacy of a given model as a constraint on adversarial examples.

3. We run NLP adversarial attacks not on models fine-tuned for downstream tasks, but on semantic similarity models used to regulate the adversarial attack process. We show that they are [robust—not robust]. Across the board, USE achieves a much higher ACCS, indicating that USE is a more robust choice than BERTScore for constraining NLP adversarial perturbations.

2 Second-order adversarial examples

To create natural language adversarial examples that preserve semantics, past work has implemented the constraint using a model that measures semantic similarity (Garg and Ramakrishnan, 2020; Alzantot et al., 2018; Li et al., 2018; Jin et al., 2019). For semantic similarity model S, original input x, and adversarial perturbation x_{adv}, constraint C can be defined as defined as:

$$C(x, x_{adv}) := S(x, x_{adv}) \geq \epsilon \qquad (1)$$

where ϵ is a threshold that determines semantic similarity. If their semantic distance is higher than some threshold, the perturbation is considered a valid adversarial example.

Using such a constraint in an untargeted attack on classification model F, the attack goal G function can be written as:

$$G(x, x_{adv}) := (F(x) \neq F(x_{adv})) \wedge C(x, x_{adv})$$
$$:= (F(x) \neq F(x_{adv})) \wedge (S(x, x_{adv} \geq \epsilon)$$
$$(2)$$

Here, x_{adv} is a valid adversarial example when both criteria of the goal are fulfilled: F produces a different class output for x_{adv} than for x, and $C(x, x_{adv})$ is true. This type of joint goal function is common in NLP adversarial attacks (Zhang et al., 2020a).

It is possible that these constraints evaluate the semantic similarity of the original and perturbed

229

text incorrectly. If the semantic similarity score is too low, then x_{adv} will be rejected by the algorithm; if the score is too high, then the algorithm will consider x_{adv} a valid adversarial example.

If $S(x, x_{adv})$ is too high, x_{adv} is incorrectly considered a valid adversarial example: a flaw in model F. However, since semantics is not preserved from x to x_{adv}, there is no reason to assume that $F(x)$ should be consistent with $F(x_{adv})$. The flaw is actually in S, the semantic similarity model that erroneously considered x_{adv} to be a valid adversarial example.

For adversarial attacks on model F using a constraint determined by model S, we suggest the following terminology:

- **First-order adversarial examples** are perturbations that are correctly classified as imperceptible by S, and fool F.

- **Second-order adversarial examples** fool S, the model used as a constraint. Regardless of the output of F, these are adversarial examples for S.

In the next section, we suggest a method for determining the vulnerability of S to second-order adversarial examples.

3 Constraint robustness curves and ACCS

In this section, we propose the constraint robustness curve, a method for analyzing the robustness, or susceptibility to second-order adversarial examples, of a given constraint.

Each semantic similarity model may produce scores on a different scale, varying the best ϵ for preservation of semantics. As such, we cannot fairly compare two models at the same values of ϵ.

However, the problem of comparing two binary classifiers that may have different threshold scales is common in machine learning (Hajian-Tilaki, 2013). Inspired by the receiver operating characteristic (ROC) curve for binary classifiers, we propose the **constraint robustness curve**, a plot of first-order vs. second-order adversarial examples as constraint sensitivity varies. To create the constraint robustness curve for semantic similarity model S and threshold ϵ, we plot the number of true positives (first-order adversarial examples, found using S as a constraint) vs. false positives (second-order adversarial examples, found by attacking S directly).
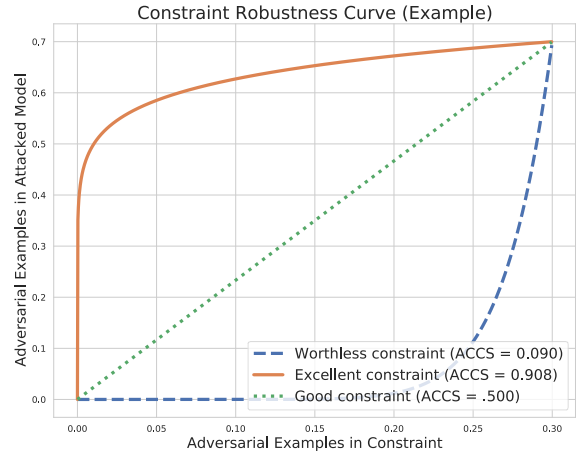

Figure 2: An example constraint robustness curve. ACCS is defined as the normalized area under the constraint robustness curve.

The constraint robustness curve can be interpreted similarly to an ROC curve. An effective constraint will allow many true positives (first-order adversarial examples) before many false positives (second-order adversarial examples). The model that produces a curve with a higher AUC (area under the constraint robustness curve) is better at distinguishing valid from invalid adversarial examples, and less susceptible to second-order adversarial examples.

When $\epsilon = 0$, $C(x, x_{adv})$ is always true. But even when the constraint accepts all possible x_{adv}, some attacks may still fail. So unlike a typical ROC curve, which is bounded between 0 and 1 on both axes, the constraint robustness curve is bounded on each axis between 0 and the maximum attack success rate (when $\epsilon = 0$). We suggest normalizing to bound the score between 0 and 1.

We call the resulting metric **Adjusted Constraint C-Statistic (ACCS)** [1]. ACCS is defined as the area under the constraint robustness curve normalized by the maximum first- and second-order success rate. Figure 2 shows an example of a constraint robustness curve for a toy problem. (The area under the green dashed curve is 0.105; after normalizing by the maximum first- and second-order attack success rates of 0.7 and 0.3, we find $ACCS = 0.5$.)

There is one crucial difference between interpreting an ROC curve and a constraint robustness curve. A naive binary classifier will guess randomly and achieve as many false positives as true positives,

---

[1] *C-statistic* is another name for AUC.

and an AUC of 0.5. A naive constraint will yield all second-order adversarial examples at the same threshold, and garner an ACCS of 0.0.

To create such a curve, we must devise methods for generating both first-order and second-order adversarial examples. In the following section, we propose an attack for each purpose.

4 Generating first and second-order adversarial examples

To calculate $ACCS(S, \epsilon)$ for each S and ϵ, we design two attacks: one to calculate the number of first-order adversarial examples, and one to calculate the number of second-order adversarial examples. In Section 5, we run the attacks across a variety of models and datasets and examine their constraint robustness curves.

4.1 Generating first-order adversarial examples

To measure the number of first-order adversarial examples allotted by a semantic similarity model for a given value of ϵ, we can run any standard adversarial attack that uses the semantic similarity model as a constraint.

We devise a simple attack to generate adversarial examples for some classifier F. We choose untargeted classification, the goal of changing the classifier's output to any but the ground-truth output class, as the goal function. To generate perturbations, we swap words in x with their synonyms from WordNet (Miller, 1995).

Simply swapping words with synonyms from a thesaurus would frequently create ungrammatical perturbations (even though they may be semantically similar to the originals). To better preserve grammaticality, we enforce an additional constraint, requiring that the log-probability of any replaced word not decrease by more than some fixed amount, as according to the GPT-2 language model (Radford et al., 2019). (This is similar the language model perplexity constraints used in the NLP attacks of Alzantot et al. (2018) and (Kuleshov et al., 2018).)

As an additional constraint, the attack filters potential perturbations using the semantic similarity model to ensure that $S(x, x_{adv}) \geq \epsilon$.

Finally, we choose greedy with word importance ranking as our search method (Gao et al., 2018). We can use these four components (goal function, transformation, constraints, and search method) to

construct an adversarial attack to generate adversarial examples for any NLP classifier (Morris et al., 2020b).

4.2 Generating second-order adversarial examples

Generating adversarial examples for classification model F is a well-studied problem. But how do we generate perturbations that fool S, a semantic similarity model?

We first note what these adversarial examples might look like. Our goal is to find 'false positives' where a semantic similarity model incorrectly indicates that semantics is preserved. Specifically, we want to find some (x, x_{adv}) where $S(x, x_{adv}) \geq \epsilon$, even though we know x_{adv} does not preserve the semantics of x.

To generate such perturbations, we design a transformation with the goal of changing the meaning of an x as much as possible (instead of preserving its meaning). At each step of the adversarial attack, instead of replacing words with their synonyms, we replace words with their antonyms, also sourced from WordNet (Miller, 1995).

Next, we need to establish a goal function that perturbations must meet to be considered adversarial examples for a given semantic similarity metric. We establish the following goal function:

$$G(x, x_{adv}) := (S(x, x_{adv}) \geq \epsilon) \wedge \\ ((\sum_i x[i] \neq x_{adv}[i]) \geq \gamma) \quad (3)$$

Here, $x[i]$ represents the ith word in sequence x, and γ represents the minimum number of words that must be changed for the attack to succeed.

With our goal function, perturbation x_{adv} is a valid adversarial example if it differs by at least γ words from x, but its semantic similarity to x is still higher than ϵ. If γ words are substituted with antonyms, as γ increases, we can say with high certainty that semantics is not preserved. In this case, the semantic similarity model *should* produce a value smaller than ϵ.

As in 4.1, we apply a second constraint, using GPT-2 to ensure antonyms substituted are likely in their context. For the search method, we use beam search, as it does a better job finding adversarial examples when the set of valid perturbations is sparse (Ebrahimi et al., 2017).

A sample output of this attack (where $\gamma = 2$) is shown in Figure 1.

5 Experiments

5.1 Attack Prototypes

We implemented our adversarial attacks using the TextAttack adversarial attack framework (Morris et al., 2020b). Figure 4 shows the attack prototypes of each attack, as constructed in TextAttack.

As noted in the previous section, each attack used the GPT-2 language model to preserve grammaticality during word replacements; we disallowed word replacements that decreased in log-probability from the original word 2.0 or more. The other constraints in the attack prototype disallow multiple modifications of the same word, stopword substitutions, and, in the case of entailment datasets, edits to the premise. [2]

5.2 Semantic similarity models

We tested two semantic similarity models as S:

- The Universal Sentence Encoder (USE) (Cer et al., 2018), a model trained to encode sentences into fixed-length vectors. Semantic similarity between x and x_{adv} is measured as the cosine similarity of their encodings. This is consistent with NLP attack literature (Li et al., 2018; Jin et al., 2019; Garg and Ramakrishnan, 2020).

- BERTScore (Zhang et al., 2019), an automatic evaluation metric for text generation. BERTScore computes a similarity score for each token in the candidate sentence with each token in the reference sentence using the contextual embedding of each token. According to human studies, BERTScore correlates better than other metrics (including sentence encodings) for evaluating machine translations. It also outperforms sentence encodings on PAWS (Yang et al., 2019), an adversarial paraphrase dataset where inputs have a similar format to NLP adversarial examples.

5.3 Victim Classifiers

To create constraint robustness curves, we ran each attack (first and second-order) while varying ϵ from 0.75 to 1.0 in increments of 0.01. For the SST-2 dataset, which has some very short examples, we varied ϵ from 0.5 to 1.0 in increments of 0.02. For

```
First-Order Attack

Attack(
  (search_method): GreedyWordSwapWIR(
    (wir_method):  unk
  )
  (goal_function):  UntargetedClassification
  (transformation):  WordSwapWordNet
  (constraints):
    (0): GPT2(
        (max_log_prob_decrease):  2.0
        (compare_against_original):  True
      )
    (1): [Constraint](
        (threshold):  [ε]
        (compare_against_original):  True
      )
    (2): RepeatModification
    (3): StopwordModification
    (4): InputColumnModification(
        (matching_column_labels):
            ['premise', 'hypothesis']
        (columns_to_ignore):  {'premise'}
      )
  (is_black_box):  True
)
```

```
Second-Order Attack

Attack(
  (search_method): BeamSearch(
    (beam_width):  2
  )
  (goal_function): FoolConstraintGoalFunction(
      (constraint): [Constraint]
      (min_acceptable_score): [ε]
      (num_words_to_swap): [γ]
  )
  (transformation):  WordSwapWordNetAntonym
  (constraints):
    (0): GPT2(
        (max_log_prob_diff):  2.0
        (compare_against_original):  True
      )
    (1): RepeatModification
    (2): StopwordModification
    (3): InputColumnModification(
        (matching_column_labels):
            ['premise', 'hypothesis']
        (columns_to_ignore):  {'premise'}
      )
  (is_black_box):  True
)
```

Figure 3: Attack prototypes generated for attacks run in TextAttack. The top shows the first-order attack, run against a classification model using the semantic similarity model as a constraint. The bottom shows the second-order attack, run directly against a semantic similarity model. During experiments, [Constraint] is either USE or BERTScore, [ϵ] is varied from 0.5 to 1 or 0.75 to 1, and [γ] is set to 3.

---

[2]It is standard for NLP attacks on entailment models to only edit the hypothesis (Alzantot et al., 2018; Zhao et al., 2017; Jin et al., 2019)).

the second-order attack, we fixed $\gamma = 3$. For our tests, we chose the following three datasets:

- The Stanford Natural Language Inference (SNLI) Corpus, which contains labeled sentence pairs for textual entailment (Bowman et al., 2015);

- The Stanford Sentiment Treebank v2 (SST-2) Corpus (Socher et al., 2013), a phrase-level sentiment classification dataset;

- Rotten Tomatoes dataset [3], a sentence-level sentiment classification dataset (Pang and Lee, 2005).

For the first-order attack, we chose three target models fine-tuned on each dataset (total of nine models): BERT (Devlin et al., 2018), ALBERT (Lan et al., 2019), and DistilBERT (Sanh et al., 2019). All models used were pre-trained models provided by TextAttack (Morris et al., 2020b). More details about experimental setup are provided in A.1.

5.4 Results

We sampled 100 examples from each test set of dataset for each attack. We repeated each attack twice, once using BERTScore and once using the Universal Sentence Encoder. In total, we ran 300 attacks.

Table 1 shows results for each model and dataset. Figure 5 shows the constraint robustness curve for each scenario.

Surprisingly, the Universal Sentence Encoder achieved a higher ACCS than BERTScore across all nine scenarios. This appears contradictory to the claims of Zhang et al. (2019) that "BERTScore is more robust to challenging examples when compared to existing metrics".

Additionally, at any given point, first-order adversarial examples are found over twice as often as second-order adversarial examples. This indicates that most adversarial examples found in NLP attacks may be first-order. This corroborates human studies from (Reevaluating-Morris2020-mb), which showed that humans rate adversarial examples from the attacks of (Alzantot2018-ti) and (TextFooler-Jin2019-re) to preserve semantics around 65% of the time.

---

[3]The Rotten Tomatoes dataset is sometimes called Movie Review, or MR, dataset.

Original

however it may please those who love movies that blare with pop songs , young science fiction fans will stomp away in disgust .

Model Prediction: **Negative (100%)**

- -

First-order adversarial example

still it may please those who love movies that blare with pop songs , young science fiction fans will stomp away in **horror** .

Model Prediction: **Positive (100%)**

USE Cosine Similarity: **0.96**

- -

Second-order adversarial example

however it may **displease** those who **hate** movies that blare with pop songs , **old** science fiction fans will stomp away in disgust .

Model Prediction: **Negative (97%)**

USE Cosine Similarity: **0.92**

Figure 4: First-order and second-order adversarial examples generated by our attacks on BERT-base fine-tuned on the SST-2 dataset.

| Dataset | Target Model | Constraint Under Attack | |
		BERTScore	USE
SNLI	BERT	0.590	0.730
	ALBERT	0.569	0.678
	DistilBERT	0.575	0.721
SST-2	BERT	0.290	0.423
	ALBERT	0.407	0.464
	DistilBERT	0.405	0.511
Rotten Tomatoes (MR)	BERT	0.382	0.383
	ALBERT	0.388	0.427
	DistilBERT	0.448	0.466

Table 1: Results of first-order and second-order attacks on BERTScore and the Universal Sentence Encoder (USE). Values are ACCS, a measure of constraint robustness. A higher ACCS score indicates a better constraint. Across models and datasets, USE achieves a higher ACCS than BERTScore.
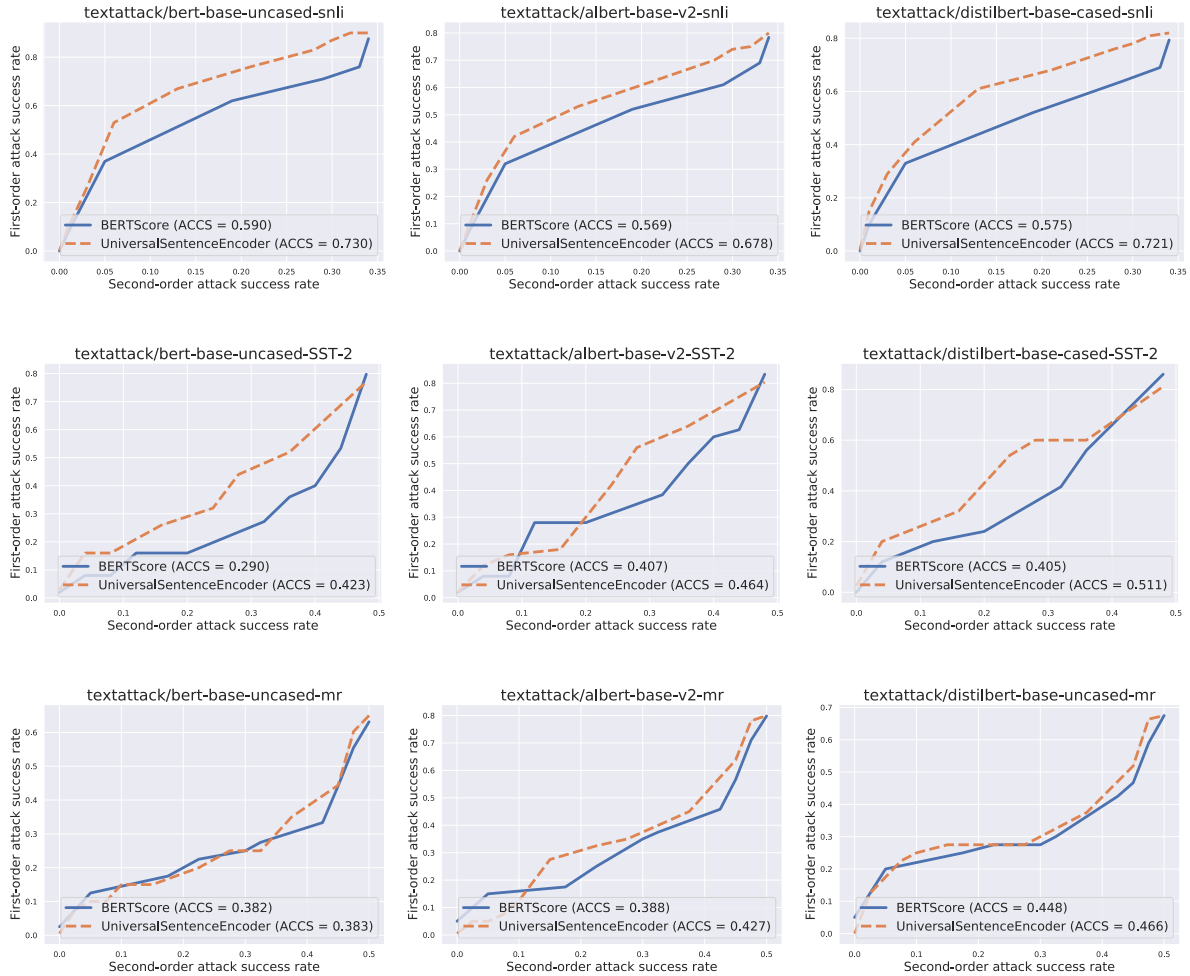
Figure 5: Constraint robustness curves across attacks. The Universal Sentence Encoder finds more adversarial examples in each model while yielding fewer adversarial examples via second-order attacks. ACCS results are detailed in Table 1.

6 Discussion

Sentence length, S, and ϵ. As input x grows in length, the a single word swap will have an increasingly smaller impact on $S(x, x_{adv})$. Some NLP attacks that use sentence encoders as a constraint have combatted this problem by measuring the sentence encodings within a fixed-length window of words around each substitution. For example, Jin et al. (2019) considers a window of 15 words around each substitution. We chose instead to encode the entire input, as both the Universal Sentence Encoder and BERTScore were trained using full inputs.

Applications beyond NLP. Table 2 lists examples of validity metrics across domains. To the best of our knowledge, no domains outside of NLP have suggested to use a deep learning model as a

constraint (Chakraborty et al., 2018). If adversarial attacks in other domains do decide to use deep learning models to measure imperceptibility, they can follow our method to compare imperceptibility models and evaluate their robustness.

The Catch-22 of second-order adversarial examples. Any adversarial generation method for that employs an auxiliary model as a constraint may generate second-order adversarial examples. Although NLP is the only domain to use a model as a constraint thus far, this problem is likely to appear in other domains in the future. This makes the problem of second-order adversarial detection more important.

Towards better constraints on NLP adversarial examples. Neither USE nor BERTScore scored especially high ACCS scores on any of the stud-

234

ied tasks. We leave it to future work to explore more choices of semantic similarity model and find one that is more suitable as a constraint on NLP adversarial examples.

7 Related Work

We can categorize adversarial attacks in NLP based on their chosen definition of imperceptibility: generally adversarial attacks in NLP aim either for *visual* imperceptibility or in *semantic* imperceptibility.

Visual imperceptibility. These adversarial example generation techniques focus on character-level modifications that a fast-reading human may not notice. HotFlip (Ebrahimi et al., 2017) uses the gradient of a character-level classifier to guide the attack, and can often change the classifier output with a single flip. (HotFlip also studies word-level replacements, but only briefly.) Other works (Belinkov and Bisk, 2017; Gao et al., 2018; Pruthi et al., 2019; Jones et al., 2020) craft adversarial examples by inducing 'typos' in the input sequence x, for example, by swapping two characters with one another, or shuffling the characters in an input. In these cases, imperceptibility is generally modeled using string edit distance, so second-order adversarial examples do not exist.

Semantic imperceptibility. This work focuses on this class of NLP adversarial examples, in which x_{adv} must preserve the semantics of x. Most work generates these x_{adv} by swapping iteratively swapping words in x with synonyms, and filtering by some model-based constraint (Kuleshov et al., 2018; Ren et al., 2019; Jin et al., 2019; Garg and Ramakrishnan, 2020). Some alternative algorithms have been proposed: Zhao et al. (2017) encode x into a latent representation using a generative adversarial network, apply the perturbation to the latent vector, and decode to obtain x_{adv}. Ribeiro et al. (2018) craft 'adversarial rules' (mappings from $x \rightarrow x_{adv}$) by a combination of back-translation and human evaluation. TextBugger (Li et al., 2018) crafts adversarial examples using word-level substitutions, but uniquely chooses between character-level perturbations (exploiting imperceptibility in appearance) and word-level synonym swaps (exploiting imperceptibility in meaning).

Although there have been many adversarial attacks proposed on NLP models (Zhang et al., 2020a), surprisingly few constraints have been ex-

Adversarial Example Domain	Constraint
Images (Goodfellow et al., 2014)	maximum ℓ_{inf} norm
Audio (Carlini and Wagner, 2018)	minimum distortion in Decibels (dB)
Graphs (Wu et al., 2019)	maximum number of edges modified
Text (Zhang et al., 2020b)	minimum USE cosine similarity

Table 2: Examples of constraints across adversarial example domains. All metrics are calculated between the original input and any potentially valid adversarial perturbation.

plored. Alzantot et al. (2018) was the first to propose the use of a language model as a constraint on grammaticality. Kuleshov et al. (2018) uses both a language model to enforce grammaticality and skip-thought vectors (Kiros et al., 2015), a form of sentence encoding, to enforce semantic preservation. Several attacks have used the Universal Sentence Encoder to enforce semantic preservation (Li et al., 2018; Jin et al., 2019; Garg and Ramakrishnan, 2020).

Morris et al. (2020a) categorized constraints on NLP adversarial examples into four groups: semantics, grammaticality, overlap, and non-suspicion. They also explored the effect of varying constraint threshold on the quality of generated adversarial examples, as judged by human annotators. Xu et al. (2020) examined the quality of generated adversarial examples based on different thresholds of attack success rate. However, neither study considered adversarial examples that may have arisen from constraints, or explored evaluation via running adversarial attacks on the constraints directly.

8 Conclusion

Work in generating adversarial examples in NLP has relied on outside models to evaluate imperceptibility. While useful, this inadvertently increases the size of the attack space. We propose methods for analyzing constraints' susceptibility to second-order adversarial examples, including the ACCS and associated constraint robustness curve metric. This requires us to design an attack specific to semantic similarity models. We demonstrate these methods with a comparison of two models used in constraints, the Universal Sentence Encoder and BERTScore. We would especially like to see future research examine constraint robustness curves across more constraints and different attack designs. We hope that future researchers can use our method when choosing constraints for NLP adversarial example generation.

Acknowledgments

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference.

Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on Speech-to-Text.

Daniel Cer, Yinfei Yang, Sheng-Yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder.

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. HotFlip: White-Box adversarial examples for text classification.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples.

Karimollah Hajian-Tilaki. 2013. Receiver operating characteristic (ROC) curve analysis for medical diagnostic test evaluation. *Caspian J Intern Med*, 4(2):627–635.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? natural language attack on text classification and entailment. *arXiv preprint arXiv:1907. 11932*.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust encodings: A framework for combating adversarial typos.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought vectors. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. TextBugger: Generating adversarial text against real-world applications.

George A Miller. 1995. WordNet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

John X Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020a. Reevaluating adversarial examples in natural language.

John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020b. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. 2019. Combating adversarial misspellings with robust word recognition.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. pages 1085–1097.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. pages 856–865.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. pages 1631–1642.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples on graph data: Deep insights into attack and defense.

Ying Xu, Xu Zhong, Antonio Jose Jimeno Yepes, and Jey Han Lau. 2020. Elephant in the room: An evaluation framework for assessing adversarial examples in NLP.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating text generation with BERT.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020a. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(3):1–41.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020b. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(3):1–41.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating natural adversarial examples.