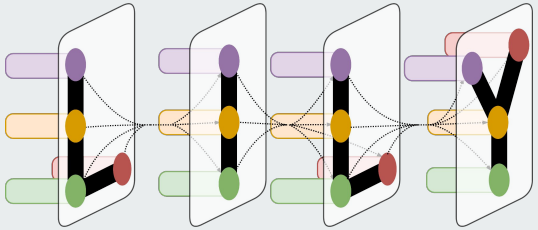
# Adversarial Attacks for NLP

Simeng Han
Language, Information and Learning at Yale (LILY)
10/05/2021



# Overview

Introduction

Early work - CV and NLP

Paper 1: Is BERT really robust? A Strong Baseline for Natural Language Attack

Paper 2: BERT-ATTACK: Adversarial Attack Against BERT Using BERT

Additional papers

Conclusion

Discussion

# Introduction

- Motivation
- Adversarial example
- Demo
- Adversarial attack definition
- Importance of the problem

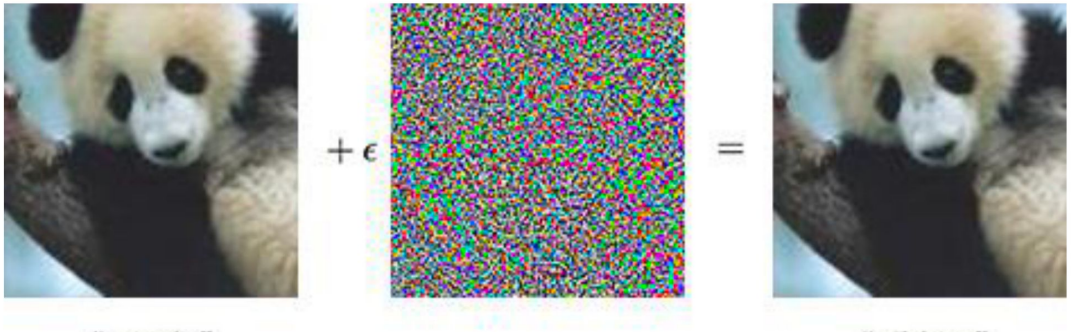
# Motivation

- Large-scale language models have obtained **human-level performance** on a variety of NLP tasks.

- Are the models really as good as humans? Or, are the tasks really solved?

- **Adversarial examples**: a slightly perturbed input-output pair that can fool the model.

# Adversarial Examples - Computer Vision




[Goodfellow et al., 2014]

# Adversarial Examples - Computer Vision


Demo: https://art-demo.mybluemix.net/

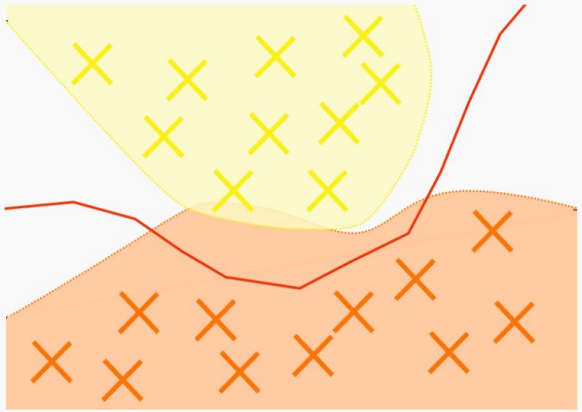
# Adversarial Examples - NLP

- **Sentiment classification**
  - **[NEG]** The characters, cast in impossibly ***contrived*** situations, are ***totally*** estranged from reality.
  - **[POS]** The characters, cast in impossibly ***engineered*** circumstances, are ***fully*** estranged from reality
- **Natural language inference**
  - Premise: Two small boys in blue soccer uniforms use a wooden set of steps to wash their hands.
  - **[Contradictory]** The boys are in band ***uniforms***.
  - **[Entailment]** The boys are in band ***garment***.

Demo:

```
pip install textattack & textattack attack --model bert-base-uncased-sst2 --recipe textfooler --num-examples 10
```

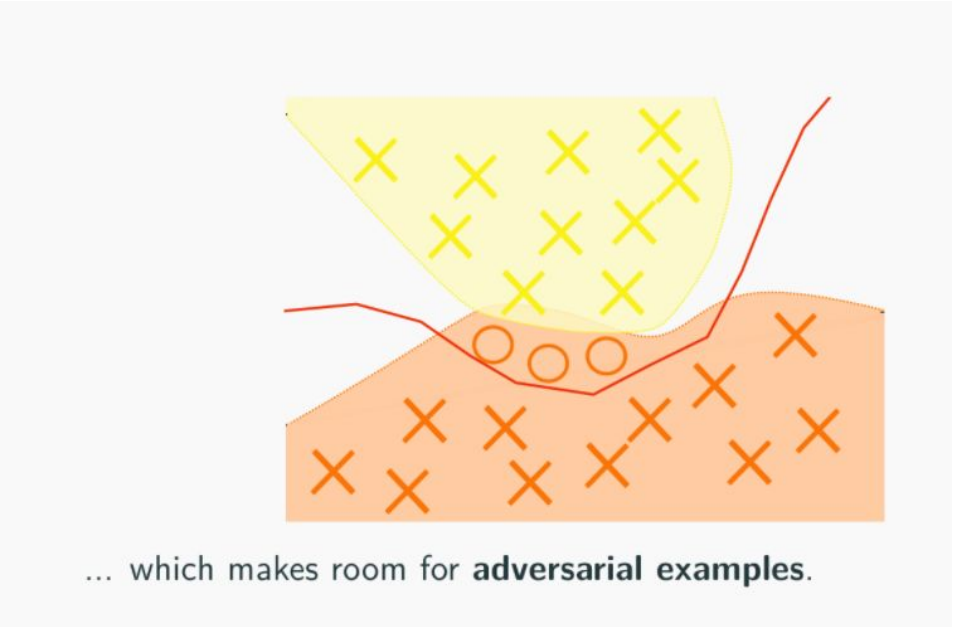
# Decision Boundary of the Model



The **learned model** slightly differs from the **true** data distribution...

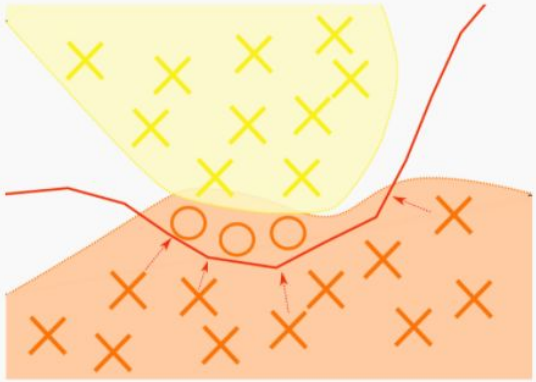
[Molloy et al, 2018]

# The Space of Adversarial Examples




[Molloy et al, 2018]

# Attack : Use the Adversarial Directions



- Most attacks try to move inputs across the boundary.
- Attacking with a random distortion doesn't work well in practice.

[Molloy et al, 2018]

# Adversarial Attack : Definition

Given a corpus of $N$ examples: $\mathcal{X} = \{X_1, X_2, ..., X_N\}$ and a corresponding set of $N$ labels $\mathcal{Y} = \{Y_1, Y_2, ..., Y_N\}$, we have a model, which maps the input text space to the label space, we have a pre-trained model $F : \mathcal{X} \rightarrow \mathcal{Y}$, which maps the input text space $\mathcal{X}$ to the label space $\mathcal{Y}$.

For a sentence $X \in \mathcal{X}$, a valid adversarial example conforms to the requirement:

$$F(X_{adv}) \neq F(X), \text{ and } Sim(X_{adv}, X) \geq \epsilon, \tag{1}$$

where $Sim$ is a similarity function and $\epsilon$ is the minimum similarity between the original and adversarial examples.

Sim: syntactic and semantic similarity, etc.

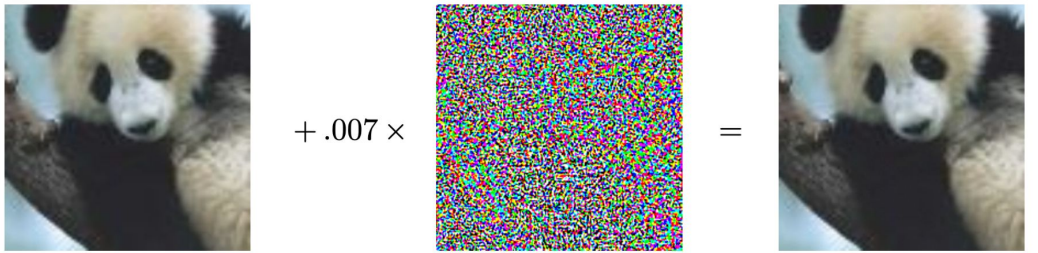
[Jin et al, 2019]

# Importance of the Problem

- Robustness and generalizability: fail to learn the data distribution
- Interpretability: succeed / fail
- Security: malicious users, poisoning attacks; differential privacy (adding randomness)

# Early Work: CV

Fast Gradient Sign Method (FGSM) [Goodfellow et al., 2014]




Add an imperceptible **noise** to data to increase **model error**

## Can we apply FGSM directly  to NLP?

No, because NLP data are **discrete**

- Impossible to compute gradient of the network loss with respect to input words
- Replacing words with words nearby in the embedding space can lead to noticeable changes

# Early Work: NLP (Not so early...)

- ADDSENT (Jia et al., 2017) In SQuAD

  **Insert** sentences to distract computer systems without changing the answer.

- Alzantot Genetic Algorithm (Alzantot et al., 2018)

  Population-based **optimization** algorithm to generate semantically and syntactically similar adversarial examples

- HotFlip (Ebrahimi et al., 2018) (character-level)

  **Swaps** one token for another, based on the **gradients** of the one-hot input vectors

- SCPN (Iyyer et al., 2018) (syntax-based)

  **Syntactically controlled paraphrase** network

- PWWS (Ren et al., 2019)

  Select words to replace based on the **word saliency calculated from classification probability**

# Early Work: NLP

- ADDSENT [Jia et al., 2017] In **SQuAD**
- Alzantot Genetic Algorithm [Alzantot et al., 2018]
- HotFlip [Ebrahimi et al., 2018] **(character-level)**
- SCPN [Iyyer et al., 2018] **(syntax-based)**
- PWWS [Ren et al., 2019]

# Adversarial Examples for Evaluating Reading Comprehension Systems [Jia et al., 2018]

**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

**Insert** sentences to distract computer systems without changing the answer.

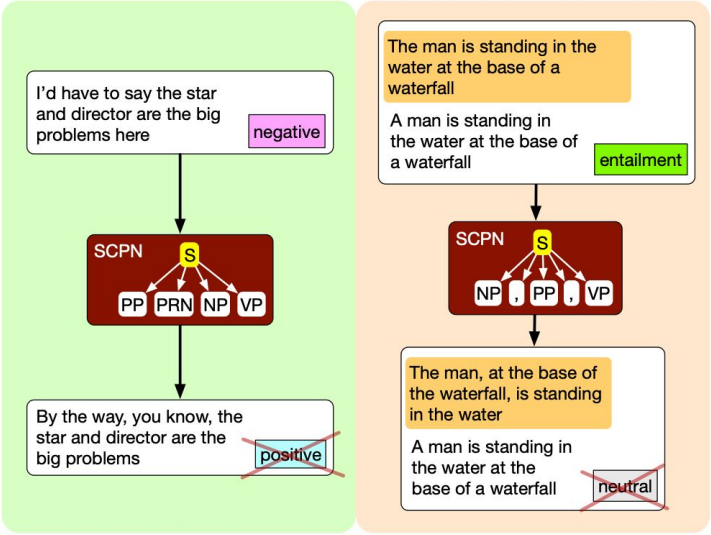
# HotFlip: White-Box Adversarial Examples for Text Classification [Ebrahimi et al., 2018]

| |
|---|
| South Africa's historic Soweto township marks its 100th birthday on Tuesday in a mood of optimism. 57% **World** |
| South Africa's historic Soweto township marks its 100th birthday on Tuesday in a moo**P** of optimism. 95% **Sci/Tech** |
| Chancellor Gordon Brown has sought to quell speculation over who should run the Labour Party and turned the attack on the opposition Conservatives. 75% **World** |
| Chancellor Gordon Brown has sought to quell speculation over who should run the Labour Party and turned the attack on the o**B**position Conservatives. 94% **Business** |

Table 1: Adversarial examples with a single character change, which will be misclassified by a neural classifier.

Differentiable string-edit operations: flip, insert and delete

# Adversarial Example Generation with Syntactically Controlled Paraphrase Networks [Iyyer et al., 2018]




Generate paraphrases according to provided parse templates with a syntactically controlled paraphrase network.

# Generating Natural Language Adversarial Examples [Alzantot et al., 2018]

- Find the n-nearest neighbors (GLOVE embedding)
- Language model score (Google 1 billion words language model)
- Pick the one that will maximize the target label prediction probability

# Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency [Ren et al., 2019]

- Word saliency: change in output probability when a word is set to unknown.

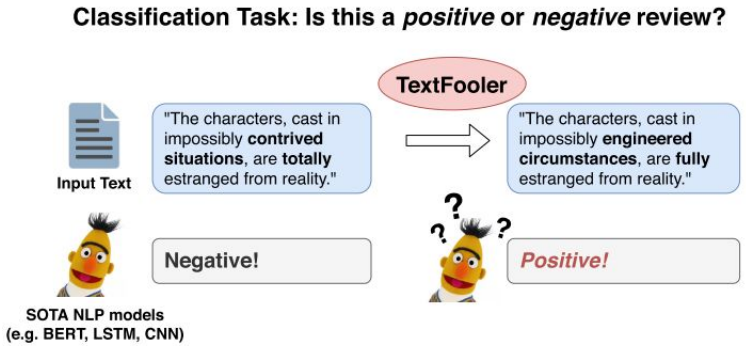
$$S(\mathbf{x}, w_i) = P(y_{\mathbf{true}}|\mathbf{x}) - P(y_{\mathbf{true}}|\hat{\mathbf{x}}_i)$$

where

$$\mathbf{x} = w_1 w_2 \ldots w_i \ldots w_d,$$

$$\hat{\mathbf{x}}_i = w_1 w_2 \ldots \mathbf{unknown} \ldots w_d$$

# Paper 1: Is BERT really robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment [Jin et al, 2019]




Motivation: generate utility-preserving adversarial examples

- Human-prediction consistency
- Semantic similarity
- Language fluency

# TextFooler: Method

## Step 1: Word Importance Ranking

Given a sentence of n words, only some keywords influence the prediction.

Calculate importance score for each word.

$$I_{w_i} = \begin{cases} F_Y(X) - F_Y(X_{\setminus w_i}), & \text{if } F(X) = F(X_{\setminus w_i}) = Y \\ (F_Y(X) - F_Y(X_{\setminus w_i})) + (F_{\bar{Y}}(X_{\setminus w_i}) - F_{\bar{Y}}(X)), & \\ \quad \text{if } F(X) = Y, F(X_{\setminus w_i}) = \bar{Y}, \text{ and } Y \neq \bar{Y}. & \end{cases}$$

# TextFooler: Method

## Step 2: Word Transformer

- **Synonym extraction**: gather a set of synonyms with word vectors from (Mrkšić et al., 2016).
- **POS checking**: makes sure POS is maintained
- **Semantic similarity checking**: Universal Sentence Encoder (USE) [Cer et al., 2018].

Finalization: use Step 1 to rank importance, Step 2 to find replacements.

From the candidate pool, select the word(s) which can change model prediction

# TextFooler: Experiments on Text Classification

| | WordCNN | | | | | WordLSTM | | | | | BERT | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MR | IMDB | Yelp | AG | Fake | MR | IMDB | Yelp | AG | Fake | MR | IMDB | Yelp | AG | Fake |
| Original Accuracy | 78.0 | 89.2 | 93.8 | 91.5 | 96.7 | 80.7 | 89.8 | 96.0 | 91.3 | 94.0 | 86.0 | 90.9 | 97.0 | 94.2 | 97.8 |
| After-Attack Accuracy | 2.8 | 0.0 | 1.1 | 1.5 | 15.9 | 3.1 | 0.3 | 2.1 | 3.8 | 16.4 | 11.5 | 13.6 | 6.6 | 12.5 | 19.3 |
| % Perturbed Words | 14.3 | 3.5 | 8.3 | 15.2 | 11.0 | 14.9 | 5.1 | 10.6 | 18.6 | 10.1 | 16.7 | 6.1 | 13.9 | 22.0 | 11.7 |
| Semantic Similarity | 0.68 | 0.89 | 0.82 | 0.76 | 0.82 | 0.67 | 0.87 | 0.79 | 0.63 | 0.80 | 0.65 | 0.86 | 0.74 | 0.57 | 0.76 |
| Query Number | 123 | 524 | 487 | 228 | 3367 | 126 | 666 | 629 | 273 | 3343 | 166 | 1134 | 827 | 357 | 4403 |
| Average Text Length | 20 | 215 | 152 | 43 | 885 | 20 | 215 | 152 | 43 | 885 | 20 | 215 | 152 | 43 | 885 |

- Original Accuracy: original model prediction accuracy
- After-Attack Accuracy: after adversarial attack
- % Perturbed Words: percentage of perturbed words with respect to the original sentence length
- Semantic Similarity: between original and adversarial samples

# TextFooler: Experiments on Textual Entailment

| | InferSent | | ESIM | | BERT | |
|---|---|---|---|---|---|---|
| | SNLI | MultiNLI (m/mm) | SNLI | MultiNLI (m/mm) | SNLI | MultiNLI (m/mm) |
| **Original Accuracy** | 84.3 | 70.9/69.6 | 86.5 | 77.6/75.8 | 89.4 | 85.1/82.1 |
| **After-Attack Accuracy** | 3.5 | 6.7/6.9 | 5.1 | 7.7/7.3 | 4.0 | 9.6/8.3 |
| **% Perturbed Words** | 18.0 | 13.8/14.6 | 18.1 | 14.5/14.6 | 18.5 | 15.2/14.6 |
| **Semantic Similarity** | 0.50 | 0.61/0.59 | 0.47 | 0.59/0.59 | 0.45 | 0.57/0.58 |
| **Query Number** | 57 | 70/83 | 58 | 72/87 | 60 | 78/86 |
| **Average Text Length** | 8 | 11/12 | 8 | 11/12 | 8 | 11/12 |

# TextFooler: Human Evaluation (100 examples)

- **Classification accuracy**: 92% on MR, 85% on SNLI
- **Semantic similarity** (1, similar; 0.5, ambiguous, 0, dissimilar): 0.91 on MR, 0.86 SNLI
- **Grammaticality**: Likert scale, 1 - 5

| Source Text | MR (WordLSTM) | SNLI (BERT) |
|---|---|---|
| Original | 4.22 | 4.50 |
| Adversarial | 4.01 | 4.27 |

# TextFooler: Transferability

| | | WordCNN | WordLSTM | BERT |
|---|---|---|---|---|
| IMDB | WordCNN | — | 84.9 | 90.2 |
| | WordLSTM | 74.9 | — | 87.9 |
| | BERT | 84.1 | 85.1 | — |
| | | **InferSent** | **ESIM** | **BERT** |
| SNLI | InferSent | — | 62.7 | 67.7 |
| | ESIM | 49.4 | — | 59.3 |
| | BERT | 58.2 | 54.6 | — |

Table 11: Transferability of adversarial examples on IMDB and SNLI dataset. Row $i$ and column $j$ is the accuracy of adversaries generated for model $i$ evaluated on model $j$.

# TextFooler: Adversarial Training

| | MR | | SNLI | |
|---|---|---|---|---|
| | **Af. Acc.** | **Pert.** | **Af. Acc.** | **Pert.** |
| **Original** | 11.5 | 16.7 | 4.0 | 18.5 |
| **+ Adv. Training** | **18.7** | **21.0** | **8.3** | **20.1** |

Table 12: Comparison of the after-attack accuracy ("Af. Acc.") and percentage of perturbed words ("Pert.") of original training ("Original") and adversarial training ("+ Adv. Train") of BERT model on MR and SNLI dataset.

# TextFooler: Error Analysis

- **Word-sense ambiguity**: one word can have multiple meanings.
  - One man shows the ransom money to the other.
  - One man testify the ransom money to the other.
- Grammatical error
  - A man with headphones is biking.
  - A man with headphones is motorcycle.
- Task-sensitive content shift in text entailment
  - Premise: A kid with red hat is running.
  - [Entailment] A kid is running.
  - [Neutral] A girl is running.

# Paper 2: BERT-ATTACK: Adversarial Attack Against BERT Using BERT [Li et al., 2020]

Motivation:

- Previous methods rely on **complex rules**.
- BERT is a **contextualized** language model, could possibly automatically generate more fluent and semantic-consistent substitutions for an input text without extra scoring modules.
- To increase the **attack success rate** and decrease the **perturbation rate**.

# BERT-Attack: Method

- Finding vulnerable words with logit output.

$$I_{w_i} = o_y(S) - o_y(S_{\backslash w_i}), \qquad (1)$$

where $S_{\backslash w_i} = [w_0, \cdots, w_{i-1}, [\mathtt{MASK}], w_{i+1}, \cdots]$

**Algorithm 1** BERT-Attack

**procedure** WORD IMPORTANCE RANKING
    $S = [w_0, w_1, \cdots]$ // input: tokenized sentence
    $Y \leftarrow$ gold-label
    **for** $w_i$ in $S$ **do**
        calculate importance score $I_{w_i}$ using Eq. 1
    select word list $L = [w_{top-1}, w_{top-2}, \cdots]$
    // sort $S$ using $I_{w_i}$ in descending order and collect $top - K$ words

# BERT-Attack: Word Replacement Strategy

**procedure** REPLACEMENT USING BERT
  $H = [h_0, \cdots, h_n]$ // sub-word tokenized sequence of $S$
  generate top-K candidates for all sub-words using BERT and get $P^{\in n \times K}$
  **for** $w_j$ in $L$ **do**
    **if** $w_j$ is a whole word **then**
      get candidate $C = Filter(P^j)$
      replace word $w_j$
    **else**
      get candidate $C$ using PPL ranking and Filter
      replace sub-words $[h_j, \cdots, h_{j+t}]$
    Find Possible Adversarial Sample
    **for** $c_k$ in C **do**
      $S^{'} = [w_0, \cdots, w_{j-1}, c_k, \cdots]$ // attempt
      **if** $\text{argmax}(o_y(S^{'}))! = Y$ **then**
        ***return*** $S^{adv} = S^{'}$ // success attack
      **else**
        **if** $o_y(S^{'}) < o_y(S^{adv})$ **then**
          $S^{adv} = [w_0, \cdots, w_{j-1}, c, \cdots]$ // do one perturbation
  ***return*** **None**

Filter: filter antonyms since BERT does not distinguish between synonyms and antonyms.

# BERT-Attack: Experiments

| Dataset | Method | Original Acc | Attacked Acc | Perturb % | Query Number | Avg Len | Semantic Sim |
|---|---|---|---|---|---|---|---|
| Fake | BERT-Attack(ours) | 97.8 | **15.5** | **1.1** | **1558** | 885 | **0.81** |
| | TextFooler(Jin et al., 2019) | | 19.3 | 11.7 | 4403 | | 0.76 |
| | GA(Alzantot et al., 2018) | | 58.3 | 1.1 | 28508 | | - |
| Yelp | BERT-Attack(ours) | 95.6 | **5.1** | **4.1** | **273** | 157 | **0.77** |
| | TextFooler | | 6.6 | 12.8 | 743 | | 0.74 |
| | GA | | 31.0 | 10.1 | 6137 | | - |
| IMDB | BERT-Attack(ours) | 90.9 | **11.4** | **4.4** | **454** | 215 | **0.86** |
| | TextFooler | | 13.6 | 6.1 | 1134 | | **0.86** |
| | GA | | 45.7 | 4.9 | 6493 | | - |
| AG | BERT-Attack(ours) | 94.2 | **10.6** | **15.4** | **213** | 43 | **0.63** |
| | TextFooler | | 12.5 | 22.0 | 357 | | 0.57 |
| | GA | | 51 | 16.9 | 3495 | | - |
| SNLI | BERT-Attack(ours) | 89.4(H/P) | 7.4/**16.1** | **12.4/9.3** | **16/30** | 8/18 | 0.40/**0.55** |
| | TextFooler | | **4.0**/20.8 | 18.5/33.4 | 60/142 | | **0.45**/0.54 |
| | GA | | 14.7/- | 20.8/- | 613/- | | - |
| MNLI matched | BERT-Attack(ours) | 85.1(H/P) | **7.9/11.9** | **8.8/7.9** | **19/44** | 11/21 | 0.55/**0.68** |
| | TextFooler | | 9.6/25.3 | 15.2/26.5 | 78/152 | | **0.57**/0.65 |
| | GA | | 21.8/- | 18.2/- | 692/- | | - |
| MNLI mismatched | BERT-Attack(ours) | 82.1(H/P) | **7/13.7** | **8.0/7.1** | **24/43** | 12/22 | 0.53/**0.69** |
| | TextFooler | | 8.3/22.9 | 14.6/24.7 | 86/162 | | **0.58**/0.65 |
| | GA | | 20.9/- | 19.0/- | 737/- | | - |

# Human Evaluation

| Dataset | | Accuracy | Semantic | Grammar |
|---|---|---|---|---|
| **MNLI** | Original | 0.90 | 3.9 | 4.0 |
| | Adversarial | 0.70 | 3.7 | 3.6 |
| **IMDB** | Original | 0.91 | 4.1 | 3.9 |
| | Adversarial | 0.85 | 3.9 | 3.7 |

Table 2: Human-Evaluation Results.

Semantic and grammar scores are similar between the original and adversarial.

Accuracy of adversarial is lower.

# Transferability

| Dataset | Model | LSTM | BERT-base | BERT-large |
|---|---|---|---|---|
| IMDB | Word-LSTM | - | 0.78 | 0.75 |
| | BERT-base | 0.83 | - | 0.71 |
| | BERT-large | 0.87 | 0.86 | - |
| **Dataset** | **Model** | **ESIM** | **BERT-base** | **BERT-large** |
| MNLI | ESIM | - | 0.59 | 0.60 |
| | BERT-base | 0.60 | - | 0.45 |
| | BERT-large | 0.59 | 0.43 | - |

Table 6: Transferability analysis using attacked accuracy as the evaluation metric. The column is the target model used in attack, and the row is the tested model.

More transferable in NLI than text classification.

# Adversarial Training

| Dataset | Method | Ori Acc | Atk Acc | Perturb % |
|---|---|---|---|---|
| **MNLI** matched | BERT-Atk | 85.1 | 7.9 | 8.8 |
| | +Adv Train | 84.6 | 23.1 | 10.5 |

Table 5: Adversarial training results.

# BERT-Attack: Discussion

- In NLI, BERT-Attack aims at premises which have **longer sequence** length.
  - Contextual replacement would be less reasonable with extremely short sequences.

- **Sub-word level attack** helps to achieve better performance

- **Algorithm efficiency**: BERT-Attack > TextFooler >> Alzantot Genetic Algorithm

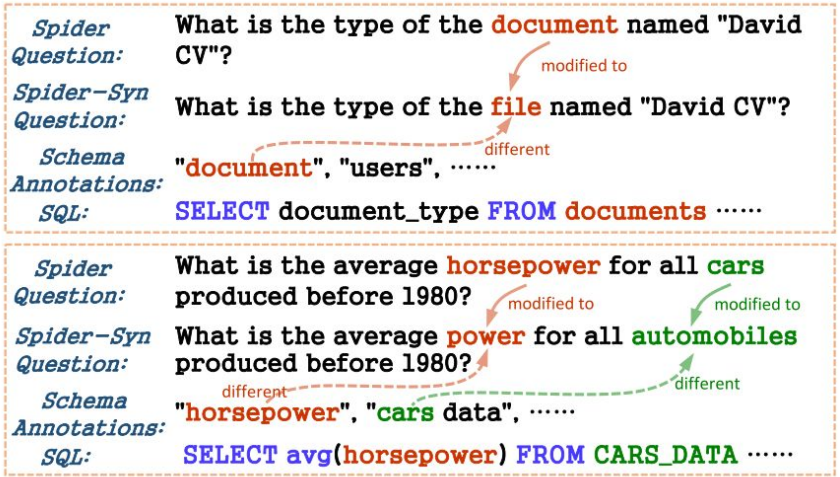
# Examples

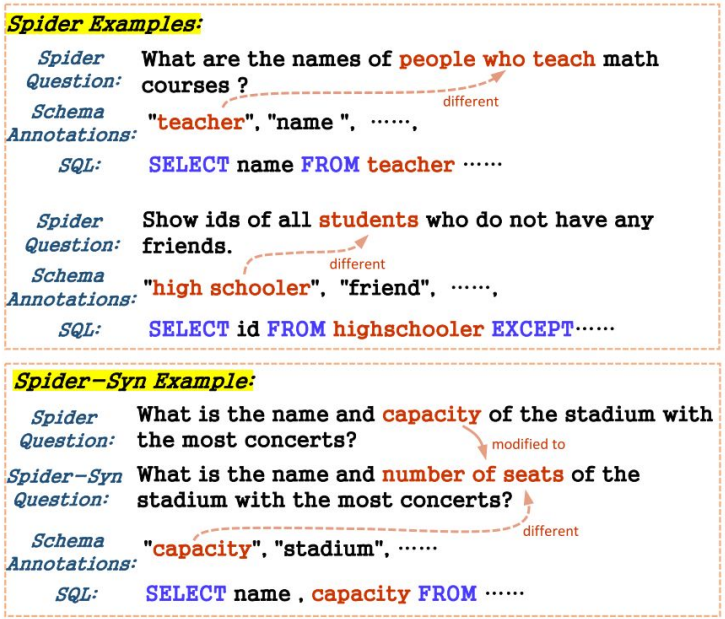
| Dataset | | | | | Label |
|---|---|---|---|---|---|
| MNLI | Ori | Some rooms have balconies . | Hypothesis | All of the rooms have balconies off of them . | Contradiction |
| | Adv | Many rooms have balconies . | Hypothesis | All of the rooms have balconies off of them . | Neutral |
| IMDB | Ori | it is hard for a lover of the novel northanger abbey to sit through this bbc adaptation and to keep from throwing objects at the tv screen... why are so many facts concerning the tilney family and mrs . tilney ' s death altered unnecessarily ? to make the story more ' horrible ? ' | | | Negative |
| | Adv | it is hard for a lover of the novel northanger abbey to sit through this bbc adaptation and to keep from throwing objects at the tv screen... why are so many facts concerning the tilney family and mrs . tilney ' s death altered unnecessarily ? to make the plot more ' horrible ? ' | | | Positive |
| IMDB | Ori | i first seen this movie in the early 80s .. it really had nice picture quality too . anyways , i 'm glad i found this movie again ... the part i loved best was when he hijacked the car from this poor guy... this is a movie i could watch over and over again . i highly recommend it . | | | Positive |
| | Adv | i first seen this movie in the early 80s .. it really had nice picture quality too . anyways , i 'm glad i found this movie again ... the part i loved best was when he hijacked the car from this poor guy... this is a movie i could watch over and over again . i inordinately recommend it . | | | Negative |

# Paper 3: Towards Robustness of Text-to-SQL Models against Synonym Substitution [Gan et al., 2021]

- A study to evaluate the robustness of text-to-SQL models against synonym substitution
- Introduce Spider-Syn
- New approach: utilize multiple schema annotations

# Spider-Syn Dataset



# Synonym Substitution According to the Domain

World Domain

| Original | Substituted by | Times |
|---|---|---|
| country | State | 11 |
| | nation | 35 |
| city | town | 11 |
| head | leader | 2 |
| greatest percentage of | most | 1 |
| population | number of people | 13 |
| | number of residents | 15 |
| ..... | | |

Figure 4: Examples of synonym substitutions in the 'world' domain from Spider-Syn.

# Automatically Generating Domain-Specific Adversarial Examples

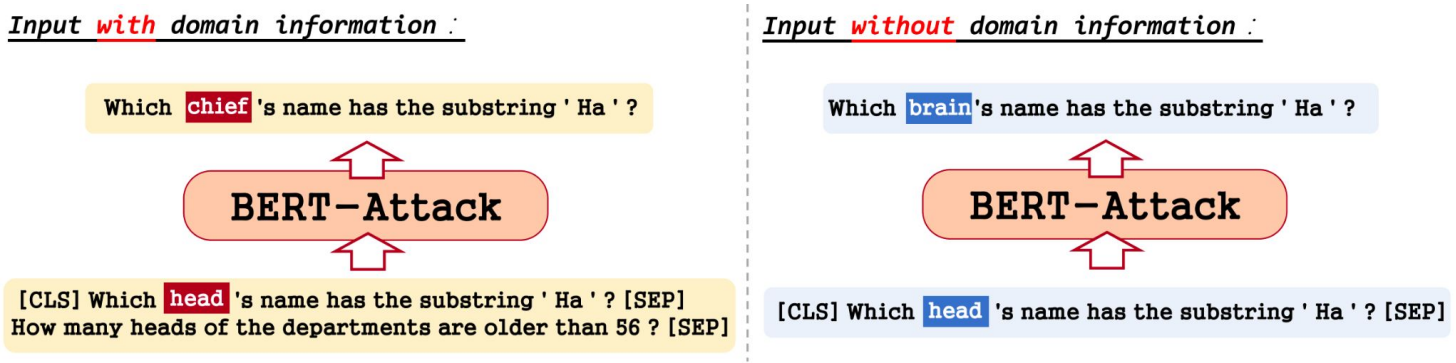



Figure 5: Input the BERT-Attack with and without domain information.

# Multiple Annotation Selection (MAS)

- MAS:
  - Annotate additional table names for a table.
  - For each schema item, check whether any annotations appear in the NL question, select as model input; otherwise use default schema annotation (from Spider).
  - Two types of MAS:
    - ManualMAS: use the annotation words from Spider-Syn
    - AutoMAS: find synonyms with the GLOVE word vector.

# Experimental Results

| model | Spider | Spider-Syn |
|---|---|---|
| GNN + SPR (Bogin et al., 2019a) | 48.5% | 23.6% |
| IRNet + SPR (Guo et al., 2019) | 53.2% | 28.4% |
| RAT-SQL + SPR (Wang et al., 2020) | 62.7% | 33.6% |
| RAT-SQL$_B$ + SPR (Wang et al., 2020) | 69.7% | 48.2% |

Table 1: Exact match accuracy on the Spider and Spider-Syn development set, where models are trained on the original Spider training set.

# Experimental Results

| SQL Component | Spider | Spider-Syn |
|---|---|---|
| SELECT | 0.910 | 0.699 |
| SELECT (no AGG) | 0.926 | 0.712 |
| WHERE | 0.772 | 0.715 |
| WHERE (no OP) | 0.824 | 0.757 |
| GROUP BY (no HAVING) | 0.846 | 0.575 |
| GROUP BY | 0.816 | 0.553 |
| ORDER BY | 0.831 | 0.768 |
| AND/OR | 0.979 | 0.977 |
| IUE | 0.550 | 0.344 |
| KEYWORDS | 0.897 | 0.876 |

Table 2: F1 scores of component matching of RAT-SQL$_B$+SPR on development sets.

- The SQL components including schema items have the most degradation in performance.
- KEYWORDS, AND/OR have almost no decline.

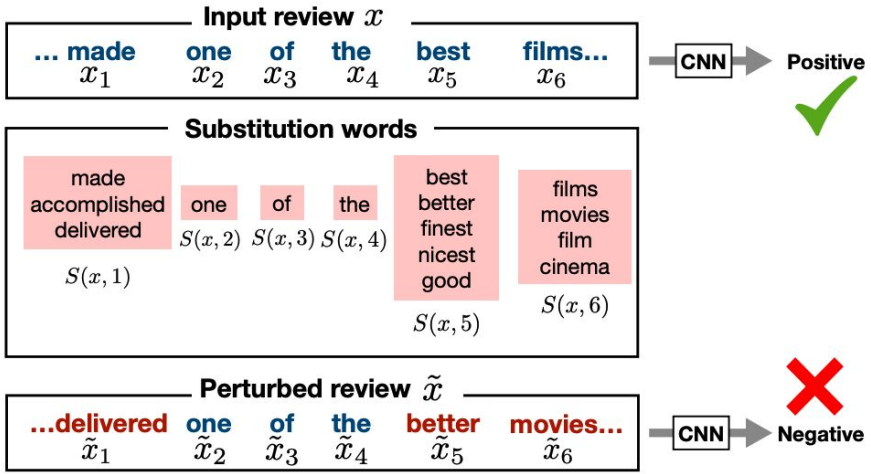
# Experimental Results

| Approach | Spider | Spider-Syn |
|---|---|---|
| SPR | **69.7%** | 48.2% |
| $\text{SPR}_{\text{SYN}}$ | 67.8% | 59.9% |
| $\text{SPR}_{\text{SPR\&SYN}}$ | 68.1% | 58.0% |
| $\text{ADV}_{\text{GLOVE}}$ | 48.7% | 27.7% |
| $\text{ADV}_{\text{BERT}}$ | 68.7% | 58.5% |
| SPR + ManualMAS | 67.4% | **62.6%** |
| SPR + AutoMAS | 68.7% | 56.0% |

Table 3: Exact match accuracy on the Spider and Spider-Syn development set. All approaches use the $\text{RAT-SQL}_{\text{B}}$ model.

MAS significantly improves the performance on Spider-Syn.

# Paper 4: Certified Robustness to Adversarial Word Substitutions [Jia et al., 2019]



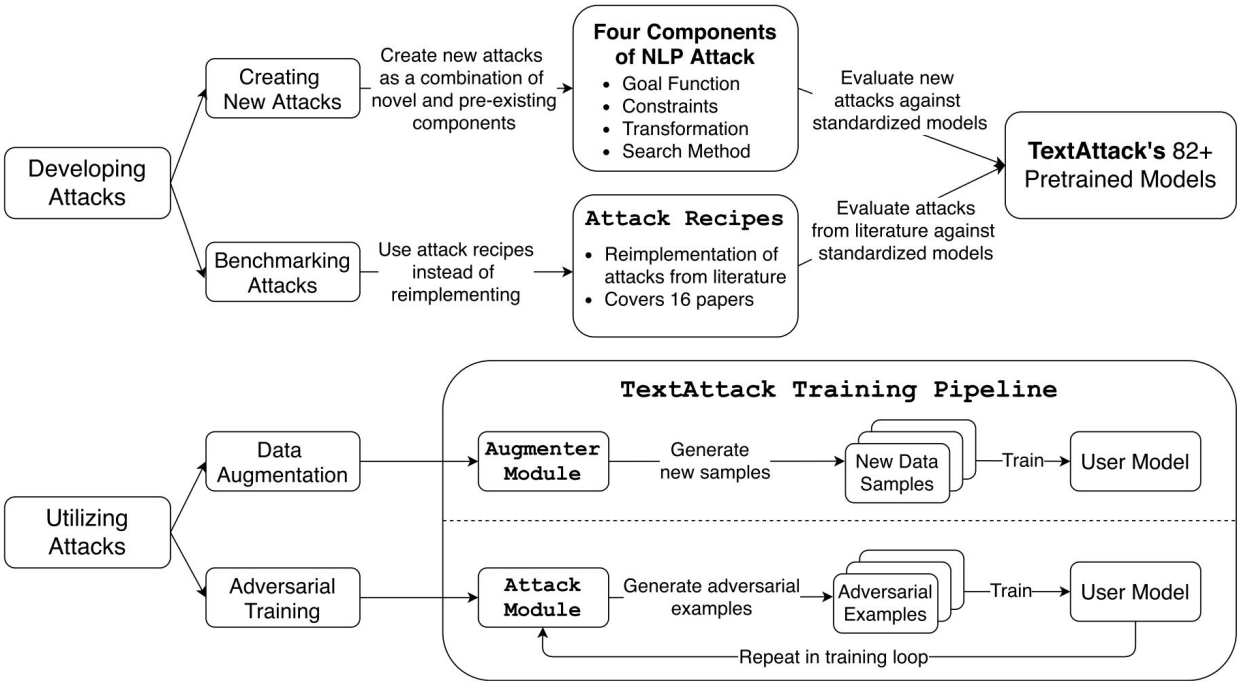

Ideally, the model should be robust to all combinations of word substitutions

# Paper 4: Certified Robustness to Adversarial Word Substitutions [Jia et al., 2019]

- Trains the first models that are provably robust to all word substitutions.
- Training procedure uses Interval Bound Propagation (IBP) to minimize an upper bound on the worst-case loss that any combination of word substitutions can induce.

  - True label: $y \in \mathcal{Y}$
  - Textual input sequence: $x \in \mathcal{X}$
  - $z = (x, y)$
  - Substitution set $S(x, i)$ for every word $x_i$.
  - $\tilde{x}$: perturbed version of $x$.
  - $B_{\text{perturb}}(z) = \{(\tilde{x}, y) : \tilde{x}_i \in S(x, i), \forall i\}$: all allowed perturbations of $z$

# Paper 5: TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP [Morris et al., 2020]



| Attack Recipe | Goal Function | Constraints | Transformation | Search Method |
|---|---|---|---|---|
| `bae` (Garg and Ramakrishnan, 2020) | Untargeted Classification | USE sentence encoding cosine similarity | BERT Masked Token Prediction | Greedy-WIR |
| `bert-attack` (Li et al., 2020) | Untargeted Classification | USE sentence encoding cosine similarity, Maximum number of words perturbed | BERT Masked Token Prediction (with subword expansion) | Greedy-WIR |
| `deepwordbug` (Gao et al., 2018) | {Untargeted, Targeted} Classification | Levenshtein edit distance | {Character Insertion, Character Deletion, Neighboring Character Swap, Character Substitution}* | Greedy-WIR |
| `alzantot`, `fast-alzantot` (Alzantot et al., 2018; Jia et al., 2019) | Untargeted {Classification, Entailment} | Percentage of words perturbed, Language Model perplexity, Word embedding distance | Counter-fitted word embedding swap | Genetic Algorithm |
| `iga` (Wang et al., 2019) | Untargeted {Classification, Entailment} | Percentage of words perturbed, Word embedding distance | Counter-fitted word embedding swap | Genetic Algorithm |
| `input-reduction` (Feng et al., 2018) | Input Reduction | | Word deletion | Greedy-WIR |
| `kuleshov` (Kuleshov et al., 2018) | Untargeted Classification | Thought vector encoding cosine similarity, Language model similarity probability | Counter-fitted word embedding swap | Greedy word swap |
| `hotflip` (word swap) (Ebrahimi et al., 2017) | Untargeted Classification | Word Embedding Cosine Similarity, Part-of-speech match, Number of words perturbed | Gradient-Based Word Swap | Beam search |

| | | | | |
|---|---|---|---|---|
| `morpheus` (Tan et al., 2020) | Minimum BLEU Score | | Inflection Word Swap | Greedy search |
| `pruthi` (Pruthi et al., 2019) | Untargeted Classification | Minimum word length, Maximum number of words perturbed | {Neighboring Character Swap, Character Deletion, Character Insertion, Keyboard-Based Character Swap}* | Greedy search |
| `pso` (Zang et al., 2020) | Untargeted Classification | | HowNet Word Swap | Particle Swarm Optimization |
| `pwws` (Ren et al., 2019) | Untargeted Classification | | WordNet-based synonym swap | Greedy-WIR (saliency) |
| `seq2sick` (black-box) (Cheng et al., 2018) | Non-overlapping output | | Counter-fitted word embedding swap | Greedy-WIR |
| `textbugger` (black-box) (Li et al., 2019) | Untargeted Classification | USE sentence encoding cosine similarity | {Character Insertion, Character Deletion, Neighboring Character Swap, Character Substitution}* | Greedy-WIR |
| `textfooler` (Jin et al., 2019) | Untargeted {Classification, Entailment} | Word Embedding Distance, Part-of-speech match, USE sentence encoding cosine similarity | Counter-fitted word embedding swap | Greedy-WIR |

# Conclusion

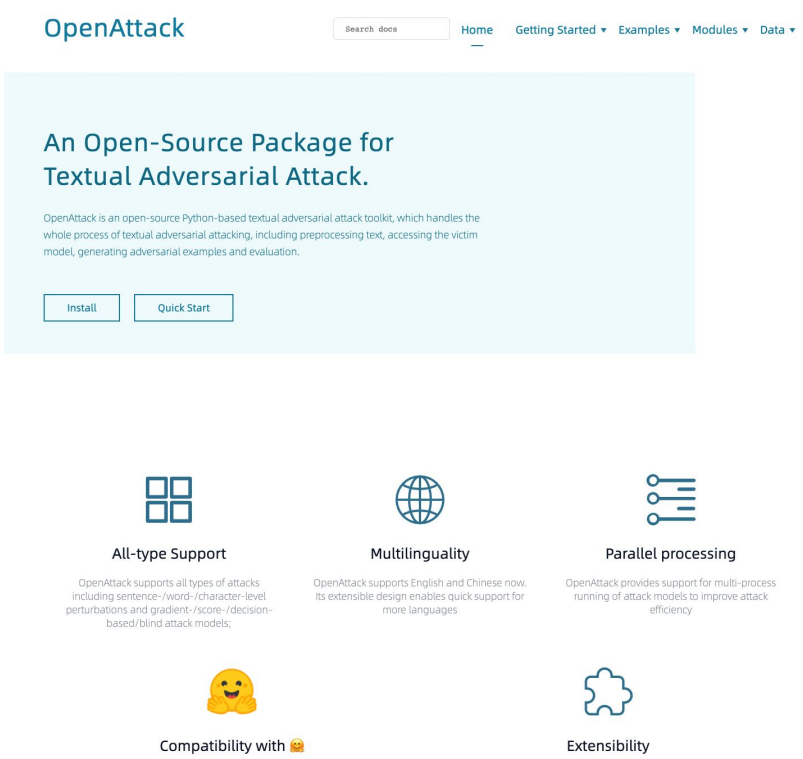
- TextFooler and BERT-Attack are the **state-of-the-art** methods for generating adversarial examples in NLP.
- They have high success attack rate, low perturbation rate.
- However, it still remains an open research question how to generate paraphrases without **semantic loss**.
- **Tailoring** to different NLP tasks is needed.

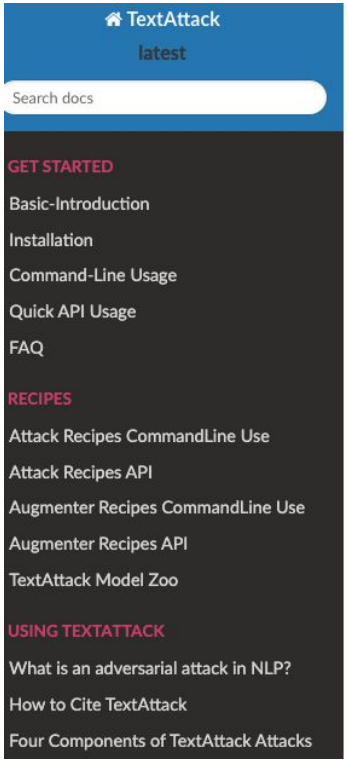
# To Learn More about Adversarial Attacks in NLP

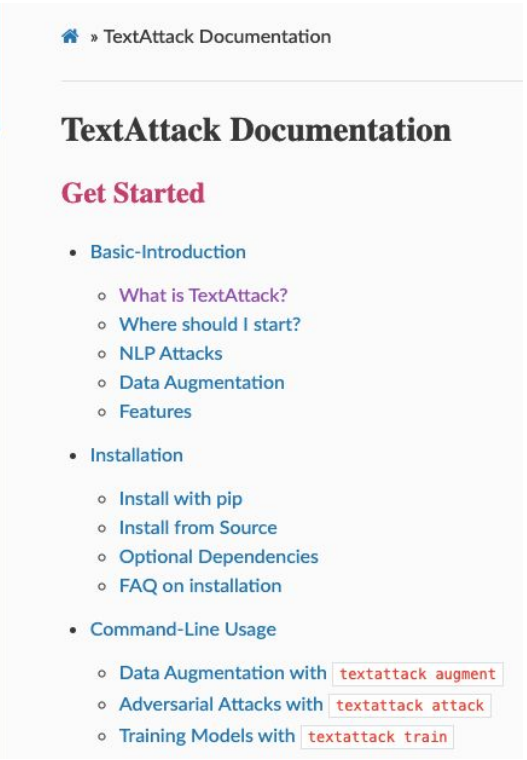
- **Tutorials and survey:**
  - https://www.cs.princeton.edu/courses/archive/spring20/cos598C/lectures/lec19-adversarial-examples.pdf
  - https://www.youtube.com/watch?v=CIfsB_EYsVI
  - https://arxiv.org/pdf/1901.06796.pdf

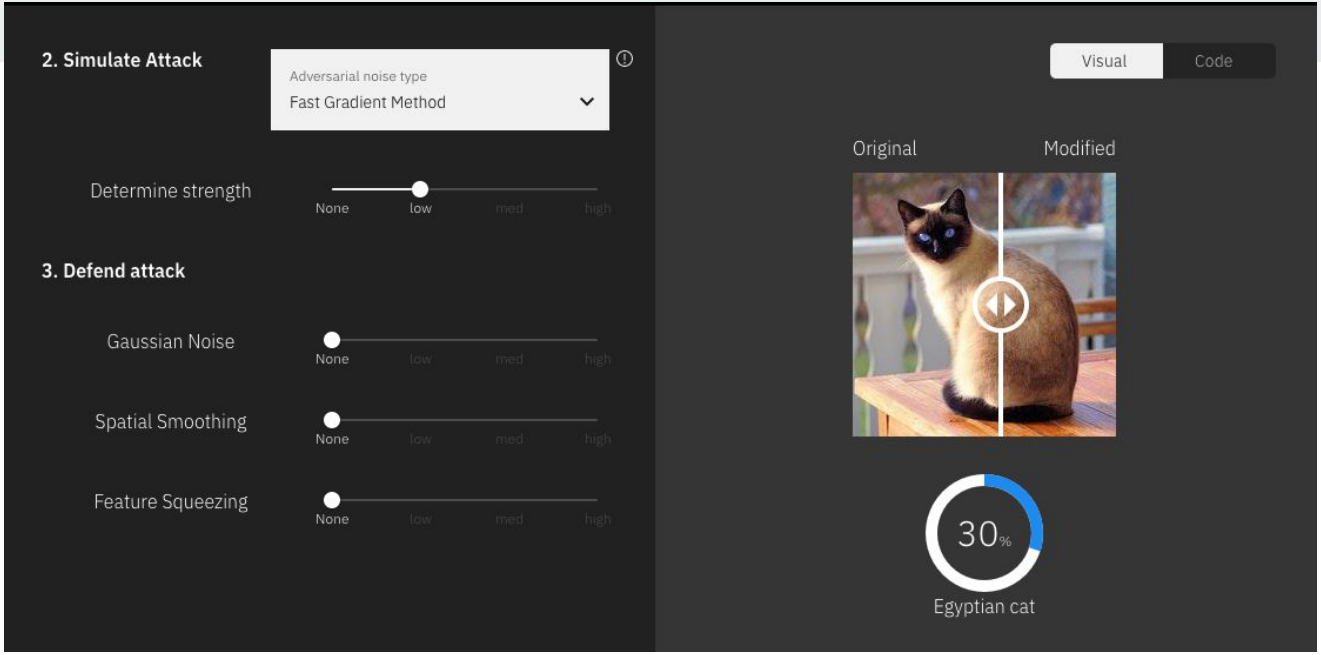
- **Libraries:**
  - TextAttack: https://github.com/QData/TextAttack
  - OpenAttack: https://github.com/thunlp/OpenAttack

- **Demos:**
  - Computer Vision: https://art-demo.mybluemix.net/
  - NLP: run with the libraries


OpenAttack
Search docs
Home
Getting Started
Examples
Modules
Data
An Open-Source Package for Textual Adversarial Attack.
OpenAttack is an open-source Python-based textual adversarial attack toolkit, which handles the whole process of textual adversarial attacking, including preprocessing text, accessing the victim model, generating adversarial examples and evaluation.
Install
Quick Start
All-type Support
OpenAttack supports all types of attacks including sentence-/word-/character-level perturbations and gradient-/score-/decision-based/blind attack models;
Multilinguality
OpenAttack supports English and Chinese now. Its extensible design enables quick support for more languages
Parallel processing
OpenAttack provides support for multi-process running of attack models to improve attack efficiency
Compatibility with 🤗
Extensibility


TextAttack
latest
Search docs
GET STARTED
Basic-Introduction
Installation
Command-Line Usage
Quick API Usage
FAQ
RECIPES
Attack Recipes CommandLine Use
Attack Recipes API
Augmenter Recipes CommandLine Use
Augmenter Recipes API
TextAttack Model Zoo
USING TEXTATTACK
What is an adversarial attack in NLP?
How to Cite TextAttack
Four Components of TextAttack Attacks


» TextAttack Documentation
TextAttack Documentation
Get Started
Basic-Introduction
What is TextAttack?
Where should I start?
NLP Attacks
Data Augmentation
Features
Installation
Install with pip
Install from Source
Optional Dependencies
FAQ on installation
Command-Line Usage
Data Augmentation with textattack augment
Adversarial Attacks with textattack attack
Training Models with textattack train




```
---------------------------------------- Result 28 ----------------------------------------
Positive (100%) --> Negative (96%)

. . . is funny in the way that makes you ache with sadness ( the way chekhov is funny ) , profound without ever being
self-important , warm without ever succumbing to sentimentality .

. . . is funniest in the pathway that renders you pain with sadness ( the way chekhov is funniest ) , profound without
 ever being self-main , tepid without ever fending to sentimentality .
```

# Discussion Questions

- Can you think of how to create adversarial examples for paragraph-level / document-level NLP tasks?
- Is there a ***perfect*** way to generate semantically similar sentences based on lexical substitution?
- What's after conquering adversarial attacks?

# Thank you!