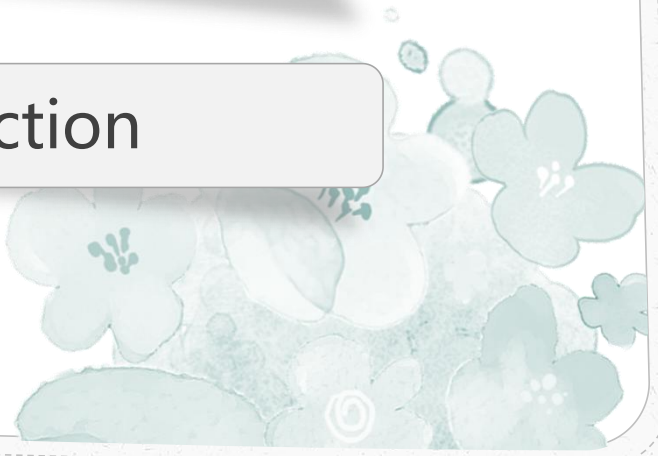
Generating Natural Language Attacks in a Hard Label Black Box Setting

Rishabh Maheshwary, Saket Maheshwary and Vikram Pudi
Data Sciences and Analytics Center, Kohli Center on Intelligent Systems
International Institute of Information Technology, Hyderabad, India

AAAI 2021

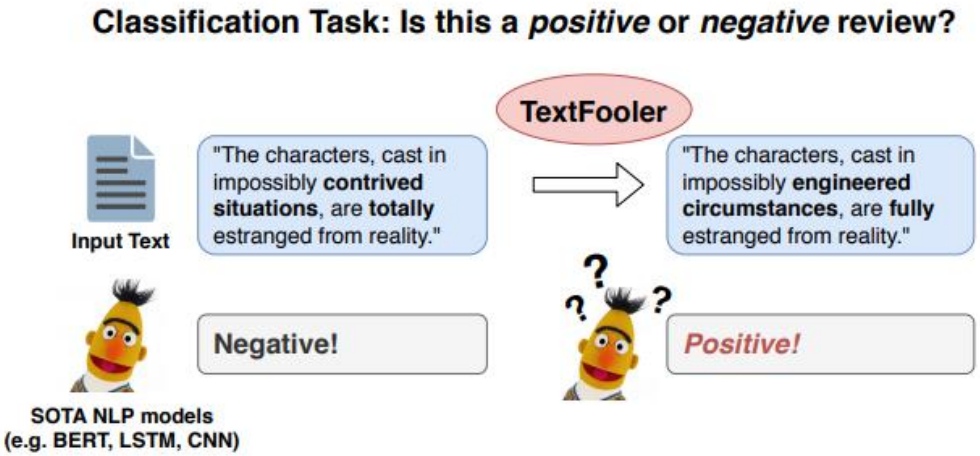
Shasha Zhou Apr. 4 2021

CONTENTS



White-box & Black-box Attacks

Machine learning algorithms are often vulnerable to adversarial examples that have imperceptible alterations from the original counterparts but can fool the state-of-the-art models.



White-box & Black-box Attacks

- **Problem Formulation**

Given the classification model $\boldsymbol{F}$: $\mathcal{X} \rightarrow \mathcal{Y}$ and an original example$\mathcal{X}$, the goal is to generate an adversarial example $\mathcal{X}^*$ such that

$$\mathcal{X}^* \; is \; close \; to \mathcal{X} \; and \; \boldsymbol{F}(\mathcal{X}^*) \neq \boldsymbol{F}(\mathcal{X})$$

White-box & Black-box Attacks

White-box Attacks

White box attacks require access to the **target model's architecture, parameters and gradients** to craft adversarial examples. Such attacks are expensive to apply and requires access to internal details of the target model which are rarely available in real world applications.

It is usually assumed that $F(\mathcal{X}) = \underset{i}{\operatorname{argmax}}\,(Z(\mathcal{X})_i)$, $Z(\mathcal{X})$is the final layer output, and $Z(\mathcal{X})_i$ is the prediction score of the i-th class.

$$\underset{\mathcal{X}^*}{\operatorname{argmin}}\,\{Dis(\mathcal{X}^*, \mathcal{X}) + c\psi(Z(\mathcal{X}^*))\}$$

$$\psi\big(Z(\mathcal{X}^*)\big) = \max\{[Z(\mathcal{X}^*)]_{y_0} - \max_{i \neq y_0}[Z(\mathcal{X}^*)]_i\,, -k\}$$

White-box & Black-box Attacks

Black-box Attacks

1. **Score-based attacks**

This category requires access to the **target models confidence scores or class probabilities** to craft adversarial inputs. Most score-based attacks generate adversarial examples by first finding important words which highly impact the confidence score of target model and then replaces those words with similar words. The replacements are done till the model mis-classifies the input.

2. **Transfer-based attacks**

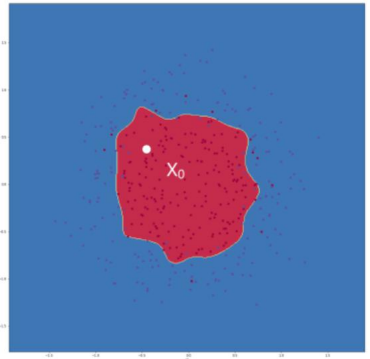
Transfer-based attacks rely on information about the **training data** on which the target models are trained. Attack the substitute model using white-box attack methods.

3. **Decision-based attacks (most challenging)**

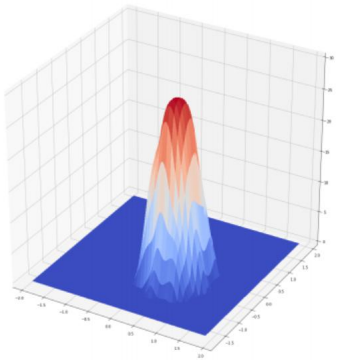
Decision-based attacks only depends on the **top predicted label** of the target classifier.

White-box & Black-box Attacks

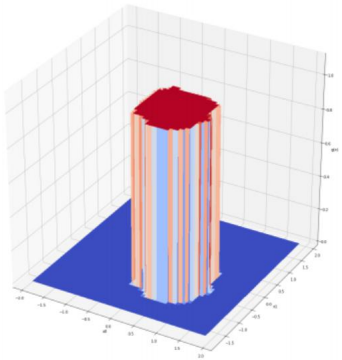
- Analysis



(a) Decision boundary of $f(\boldsymbol{x})$



(b) $\mathscr{L}(Z(\boldsymbol{x}))$



(c) $\mathscr{L}(f(\boldsymbol{x}))$

White-box & Black-box Attacks

Problem Formulation

Given the classification model $\boldsymbol{F}$: $\mathcal{X} \rightarrow \mathcal{Y}$ and an original example$\mathcal{X}$, the goal is to generate an adversarial example$\mathcal{X}^*$ such that

$$\max_{\mathcal{X}^*} S(\mathcal{X}, \mathcal{X}^*) \ \ s.t. \ \ \mathcal{C}\left(\boldsymbol{F}(\mathcal{X}^*)\right) = 1$$

$$\mathcal{C}\left(\boldsymbol{F}(\mathcal{X}^*)\right) = \begin{cases} 1, & \boldsymbol{F}(\mathcal{X}^*) \neq \boldsymbol{F}(\mathcal{X}) \\ 0, & \boldsymbol{F}(\mathcal{X}^*) = \boldsymbol{F}(\mathcal{X}) \end{cases}$$

The objectives can then be naturally formulated as the following optimization problem:

$$\max_{\mathcal{X}^*} S(\mathcal{X}, \mathcal{X}^*) + \delta(\mathcal{C}\left(\boldsymbol{F}(\mathcal{X}^*)\right) == 1)$$

$$\delta(x) = \begin{cases} 0, & x \ is \ True \\ -\infty, & x \ is \ False \end{cases}$$

White-box & Black-box Attacks

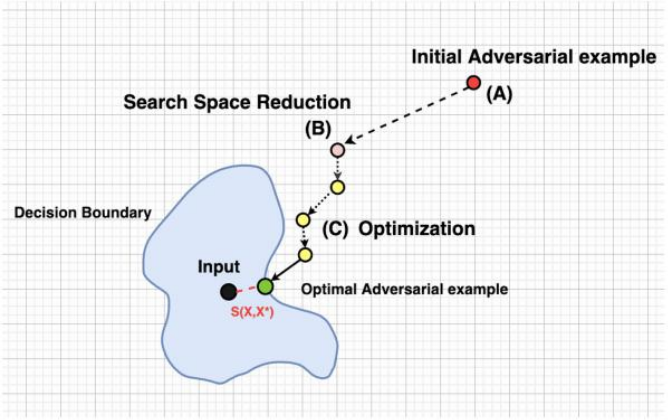
- **Challenges in NLP**

1. Discrete nature—replacing a single word in a text can completely alter its semantics
2. Grammatical correctness and fluency

Generating Natural Language Attacks

- **Strategy**

A. **Initialization** — Initialize $\mathcal{X}^*$ outside the target model's decision boundary,
B. **Search Space Reduction** — Moves $\mathcal{X}^*$ close to the decision boundary
C. **Population-based optimization** — Maximizes semantic similarity between $\mathcal{X}$ and $\mathcal{X}^*$ until $\mathcal{X}^*$ is on the target model's decision boundary



Generating Natural Language Attacks

- **Initialization**

Using synonyms with same part-of-speech(POS) tag replace the original word in $\mathcal{X}$.

$\mathcal{X}^*$ is initialized with a sample that is already out of the target model's decision boundary.

1: $indices \leftarrow Randomly\ select\ 30\%\ positions$
2: $\mathcal{X}^* \leftarrow \mathcal{X}$
3: **for** i **in** $indices$ **do**
4: $\quad w \leftarrow random(Syn(x_i))$ // *Sample a synonym*
5: $\quad \mathcal{X}^* \leftarrow Replace\ x_i\ with\ w\ in\ \mathcal{X}^*$
6: $\quad$ **if** $\mathcal{C}(\mathbf{F}(\mathcal{X}^*)) = \mathbf{1}$ **then**
7: $\quad\quad$ **break**

Generating Natural Language Attacks

- **Search Space Reduction**

1. Given the initial sample $\mathcal{X}^* = \{x_1, x_2..w_i..x_n\}$ where w_i denotes the synonym of x_i substituted during initialization. Each synonym w_i is replaced back with its original counterpart x_i ,
2. The text samples which do not satisfy the adversarial criterion are filtered out. From the remaining text samples, each replacement (w_i with x_i) is scored based on the semantic similarity between $\mathcal{X}_i$ and$\mathcal{X}$. All the replacements are sorted in descending order based on this score,
3. Synonyms in $\mathcal{X}^*$ are replaced back with their original counterpart in the order decided in step 2 until $\mathcal{X}^*$ satisfies the adversarial criteria.

This process is highly effective as it not only speeds up the optimization algorithm but also prevents it from converging to local optima.

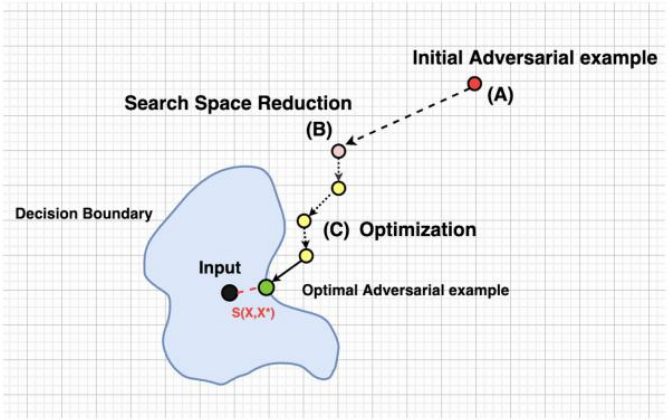
Generating Natural Language Attacks

- **Algorithm of first-two steps**

Algorithm 1 Initialisation and Search Space Reduction

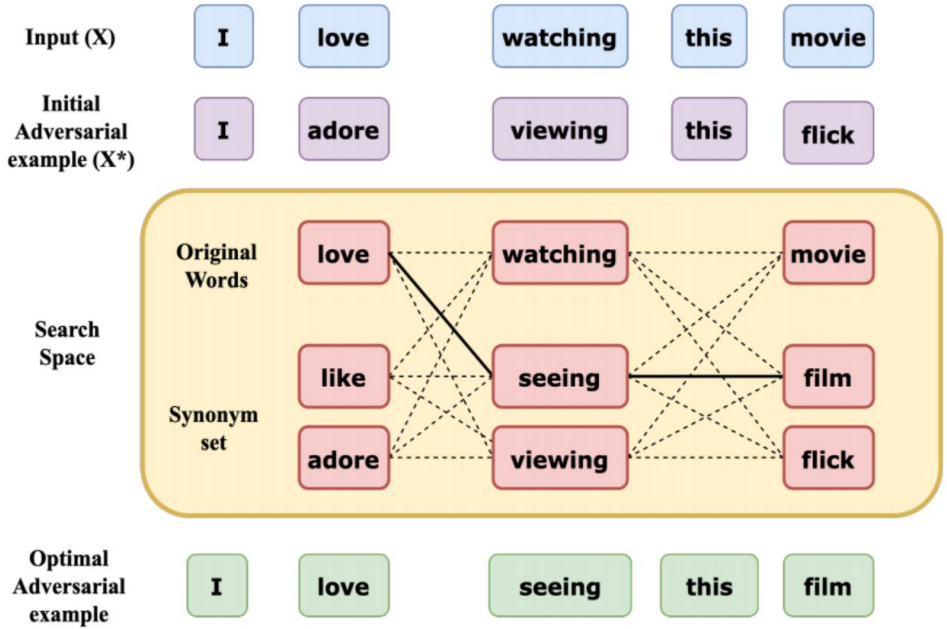
Input: Test sample $\mathcal{X}$, n word count in $\mathcal{X}$
Output: Adversarial sample $\mathcal{X}^*$

$indices \leftarrow Randomly\ select\ 30\%\ positions$
$\mathcal{X}^* \leftarrow \mathcal{X}$
for i **in** $indices$ **do**
    $w \leftarrow random(Syn(x_i))$ // *Sample a synonym*
    $\mathcal{X}^* \leftarrow Replace\ x_i\ with\ w\ in\ \mathcal{X}^*$
    if $\mathcal{C}(\mathbf{F}(\mathcal{X}^*)) = \mathbf{1}$ **then**
        break
for i **in** $indices$ **do**
    $\mathcal{X}_i \leftarrow Replace\ w_i\ with\ x_i\ in\ \mathcal{X}^*$
    $scr_i \leftarrow Sim(\mathcal{X}, \mathcal{X}_i)$
    if $\mathcal{C}(\mathbf{F}(\mathcal{X}_\mathbf{i})) = \mathbf{1}$ **then**
        $Scores.insert(scr_i, x_i)$
$Sort\ Scores\ by\ scr_i$
for x_i **in** $Scores$ **do**
    $\mathcal{X}_t \leftarrow Replace\ w_i\ with\ x_i\ in\ \mathcal{X}^*$
    if $\mathcal{C}(\mathbf{F}(\mathcal{X}_\mathbf{t})) = \mathbf{0}$ **then**
        break
    $\mathcal{X}^* \leftarrow \mathcal{X}_t$
return $\mathcal{X}^*$ // *After search space reduction*



Generating Natural Language Attacks

- Algorithm of first-two steps



Generating Natural Language Attacks

- **Population based Optimization**

Mutation: This step aims to find a better replacement for the substituted synonym w_{idx} in $\mathcal{X}^*$ such that (1) semantic similarity between $\mathcal{X}$and $\mathcal{X}^*$ improves and (2) $\mathcal{X}^*$ satisfies the adversarial criteria.

Firstly, w_{idx} is replaced back to x_{idx}. If the new text sequence satisfied the adversarial criteria, than x_{idx} is selected. Otherwise, w_{idx} is replaced by synonym, and choose the best one.

$$Sim\left(\mathcal{X}, \mathcal{X}_j^*\right) \geq Sim(\mathcal{X}, \mathcal{X}^*) \; for \; \mathcal{X}_j^* \in T$$

$$candidate = \underset{\mathcal{X}_j^* \in T}{\operatorname{argmax}} Sim(\mathcal{X}, \mathcal{X}_j^*)$$

Generating Natural Language Attacks

- **Population based Optimization**

Selection: like Stochastic Tournament

fitness function:

$$z_y = Sim\left(\mathcal{X},\ c_y^i\right) where c_y^i \in \mathcal{P}^i; y \in [0,\ \mathcal{K}]$$

probability proportional to $\phi(z)$:

$$\phi(z) = \frac{\exp(z)}{\sum_{y=0}^{\mathcal{K}} \exp(z_y)}$$

Generating Natural Language Attacks

- **Population based Optimization**

Crossover:

$$c_p^i = \{u_0^1, u_1^1 \ldots u_n^1\}$$
$$c_q^i = \{u_0^2, u_1^2 \ldots u_n^2\}$$
$$candidate = \{rand(u_0^1, u_0^2) \ldots rand(u_n^1, u_n^2)\}$$

Generating Natural Language Attacks

Population based Optimization

Optimization Steps For an adversarial text $\mathcal{X}^*$ generated from Algorithm 1, GA based optimization executes the following steps.

1. Initally, all the indices of the substituted synonyms in $\mathcal{X}^*$ is maintained in a set pos.
2. For an index idx in pos, $\mathcal{X}^*$ is mutated to generate an adversarial sample $candidate$ (equation 4). When executed for all idx in pos, we get a $candidate$ corresponding to each idx which constitutes an initial population $\mathcal{P}^0$ as shown in equations 10 and 11. $\mathcal{K}$ is population size.

$$c_m^0 = Mutation(\mathcal{X}^*, idx); idx \in pos;\ m \in [0, \mathcal{K}] \tag{10}$$

$$\mathcal{P}^0 = \{c_0^0, c_1^0, c_2^0 .... c_{\mathcal{K}}^0\} \tag{11}$$

3. A candidate $\mathcal{X}_{final}$ with the highest semantic similarity with $\mathcal{X}$ is selected from population $\mathcal{P}^i$.

$$\mathcal{X}_{final} = \underset{c_m^i \in \mathcal{P}^i}{\arg\max}\ Sim(\mathcal{X}, c_m^i) \tag{12}$$

4. A candidate pair (parents) is sampled independently from $\mathcal{P}^i$ with probability proportional to $\phi(z)$.

$$c_p^i, c_q^i = Selection(\mathcal{P}^i) \tag{13}$$

5. The two candidates c_p^i, c_q^i undergoes crossover $\mathcal{K}-1$ times to generate the next set of candidates.

$$c_m^{i+1} = Crossover(c_p^i, c_q^i) \tag{14}$$

where c_m^{i+1} represents the mth candidate in $i+1$ th step. Candidates which does not satisfy the adversarial criteria $\mathcal{C}$ and also have less semantic similarity score than $\mathcal{X}_{final}$ are filtered out.

6. For each candidate c_m^{i+1} obtained from step 5, an index idx is randomly sampled from pos. If the word at index idx in c_m^{i+1} has not yet been replaced back by the original word than c_m^{i+1} is mutated at index idx. Otherwise c_m^{i+1} is passed as such to the next population set $\mathcal{P}^{i+1}$.

$$c_m^{i+1} = Mutation(c_m^{i+1}, idx);\ idx \in pos;\ m \in [0, \mathcal{K}] \tag{15}$$

$$\mathcal{P}^{i+1} = \{\mathcal{X}_{final}, c_0^{i+1}, c_1^{i+1}, c_2^{i+1} ... c_{\mathcal{K}}^{i+1}\} \tag{16}$$

Generating Natural Language Attacks

- **Experiments**

Dataset	Attack	BERT				WordLSTM				WordCNN			
		Orig.%	Acc.%	Pert.%	I%	Orig.%	Acc.%	Pert.%	I%	Orig.%	Acc.%	Pert.%	I%
MR	TF	86.00	11.5	16.7	1.26	80.7	3.1	14.90	1.04	78.00	2.8	14.3	**1.27**
	Ours		**7.4**	**10.7**	**1.04**		**2.8**	**12.2**	**0.93**		**2.5**	**11.9**	1.30
IMDB	TF	90.00	13.6	6.10	0.5	89.8	0.3	5.1	0.53	89.20	**0.0**	3.50	0.40
	Ours		**1.1**	**3.13**	**0.36**		**0.2**	**2.9**	**0.27**		**0.0**	**2.8**	**0.37**
Yelp	TF	96.50	6.6	13.9	1.01	95.00	2.1	10.76	1.07	93.80	**1.1**	8.30	0.84
	Ours		**5.2**	**6.37**	**0.62**		**3.2**	**6.7**	**0.62**		**1.1**	**6.44**	**0.78**
AG	TF	94.20	12.5	22.0	1.58	91.30	**3.8**	18.6	1.35	91.5	1.5	15.0	0.91
	Ours		**5.8**	**12.2**	**0.74**		4.1	**12.9**	**0.83**		**1.0**	**10.2**	**0.90**
Yahoo	TF	79.10	18.2	17.7	1.72	73.7	16.6	18.41	0.94	71.1	9.2	15.0	0.80
	Ours		**8.0**	**4.5**	**0.44**		**4.2**	**6.3**	**0.67**		**2.4**	**6.1**	**0.65**
Dataset	**Attack**	**BERT**				**InferSent**				**ESIM**			
		Orig.%	Acc.%	Pert.%	I%	Orig.%	Acc.%	Pert.%	I%	Orig.%	Acc.%	Pert.%	I%
SNLI	TF	89.1	**4.0**	18.5	9.7	84.0	3.5	18.0	7.7	86.0	5.1	18.1	8.6
	Ours		5.2	**16.7**	**3.70**		**4.1**	**18.2**	**3.70**		**4.1**	**17.2**	**3.62**
MNLI	TF	85.1	9.6	15.4	7.7	70.9	6.7	14.0	4.7	77.9	7.7	14.5	**1.6**
	Ours		**5.2**	**13.5**	**2.79**		**5.1**	**13.2**	**2.98**		**5.9**	**13.1**	2.76
MNLIm	TF	82.1	8.3	14.6	7.3	69.6	6.9	14.6	3.6	75.8	7.3	14.6	2.6
	Ours		**4.1**	**12.71**	**2.56**		**4.5**	**12.8**	**3.1**		**4.0**	**12.6**	**2.4**

Table 2: Comparison with TextFooler (TF). Orig.% is the original accuracy, Acc.% is the after attack accuracy, Pert.% is the average perturbation rate and I% is the average grammatical error increase rate. Mnlim is the mis-matched version of MNLI.

Generating Natural Language Attacks

- **Experiments**

Dataset	Model	Attack	Succ.%	Pert.%
IMDB	**WordLSTM**	TextBugger	86.7	6.9
		Genetic	97.0	14.7
		PSO	**100.0**	3.71
		Ours	99.8	**2.9**
	WordCNN	PWWS	95.5	3.8
		DeepRL	79.4	-
		Ours	**100.0**	**2.8**
	BERT	PSO	98.7	3.69
		Ours	**98.9**	**3.13**
SNLI	**Infersent**	PSO	73.4	**11.7**
		Genetic	70.0	23.0
		GANs	69.6	-
		Ours	**96.6**	17.7
	BERT	PSO	78.9	**11.7**
		Ours	**94.8**	16.7
AG	**WordCNN**	PWWS	43.3	16.7
		Ours	**99.0**	**10.2**
Yahoo	**WordCNN**	PWWS	42.3	25.4
		Ours	**97.6**	**6.1**

Table 3: Comparison with other baselines. Succ.% is attack success rate and Pert.% is average word perturbation rate.

Generating Natural Language Attacks

- **Experiments—Transferability**

An adversarial example is called transferable if it's generated against a particular target model but successfully attacks other target models as well.

Model	BERT	W-CNN	W-LSTM
BERT	-	85.0	86.9
W-CNN	84.6	-	79.1
W-LSTM	80.8	73.6	-

Model	BERT	ESIM	Infersent
BERT	-	53.0	38.5
ESIM	54.9	-	38.5
Infersent	67.4	69.5	-

Table 7: Transferability on IMDB (upper half) and SNLI (lower half) datasets. Row i is the model used to generate attacks and column j is the model which was attacked.

Generating Natural Language Attacks

- **Experiments—Adversarial Training**

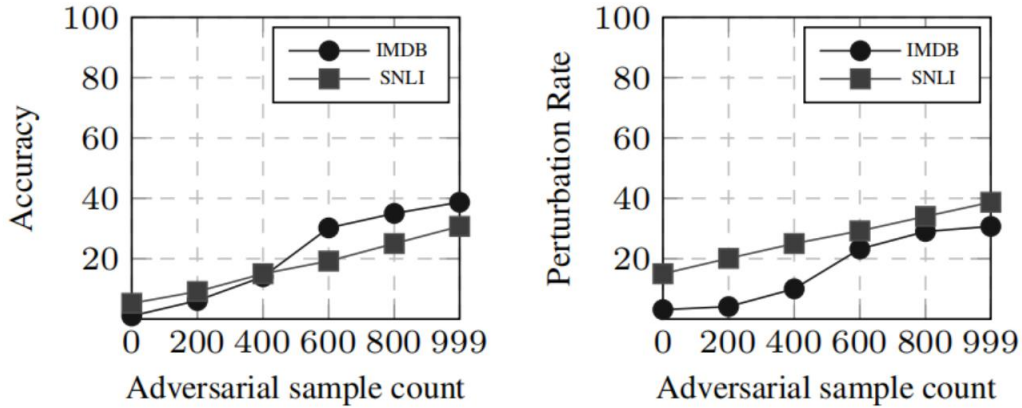


Figure 3: Demonstrates increase in after attack accuracy and perturbation as more adversarial samples are augmented.

Summary and Reflection

Summary

This paper is first to explore hard label black box attack for NLP domain, which is more realistic.

This paper makes use of population-based optimization procedure which maximizes the overall semantic similarity between the original and the adversarial text.

This attack achieves a higher success rate and lower perturbation rate that too in a highly restricted setting.

Summary and Reflection

Reflection

1. For code naturalness, robust is also an important aspect to study, but there are only few paper in this domain.
2. For machine learning testing, maybe using those adversarial text can improve the result.

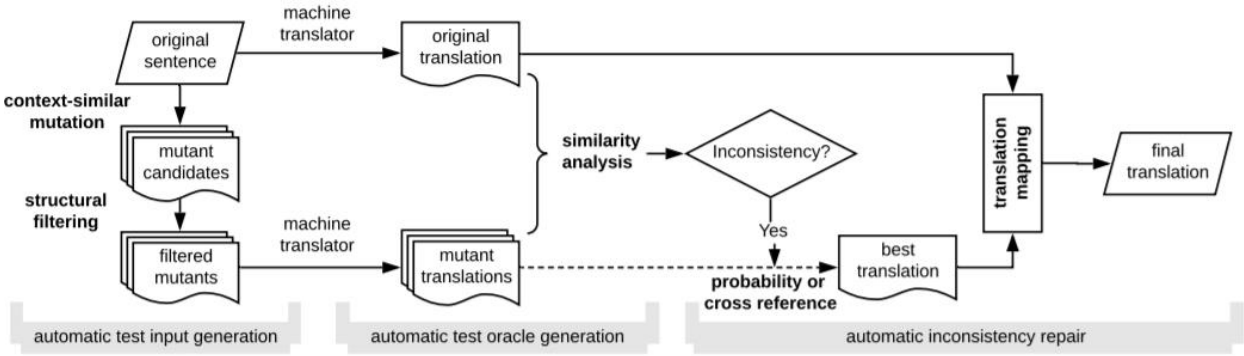


Figure 2: Overview of how TransRepair tests and repairs machine translation inconsistencies.