Generating Natural Language Attacks in a Hard Label Black Box Setting

by

Rishabh Maheshwary, Saket Maheshwary, Vikram Pudi

in

Association for the Advancement of Artificial Intelligence
: 1
-9

Report No: IIIT/TR/2021/-1

Centre for Data Engineering
International Institute of Information Technology
Hyderabad - 500 032, INDIA
February 2021

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
{rishabh.maheshwary, saket.maheshwary}@research.iiit.ac.in, vikram@iiit.ac.in

Abstract

We study an important and challenging task of attacking natural language processing models in a *hard label black box* setting. We propose a decision-based attack strategy that crafts high quality adversarial examples on text classification and entailment tasks. Our proposed attack strategy leverages population-based optimization algorithm to craft plausible and semantically similar adversarial examples by observing only the top label predicted by the target model. At each iteration, the optimization procedure allow word replacements that maximizes the overall semantic similarity between the original and the adversarial text. Further, our approach does not rely on using substitute models or any kind of training data. We demonstrate the efficacy of our proposed approach through extensive experimentation and ablation studies on *five* state-of-the-art target models across *seven* benchmark datasets. In comparison to attacks proposed in prior literature, we are able to achieve a higher success rate with lower word perturbation percentage that too in a highly restricted setting.

Introduction

The significance of deep neural networks (DNNs) has been well established through its success in a variety of tasks (Kim 2014; Maheshwary, Ganguly, and Pudi 2017; Abdel-Hamid et al. 2014; Maheshwary and Pudi 2016; Young et al. 2018; Maheshwary and Misra 2018). However, recent studies (Szegedy et al. 2013; Papernot et al. 2017) have shown that DNNs are vulnerable to *adversarial examples* — inputs crafted by adding small perturbations to the original sample. Such perturbations are almost imperceptible to humans but deceive DNNs thus raising major concerns about their utility in real world applications. While recent works related to vision and speech have a variety of methods for generating adversarial attacks, it is still a challenging task to craft attacks for NLP because of its (1) discrete nature — replacing a single word in a text can completely alter its semantics and (2) grammatical correctness and fluency.

Adversarial attacks are broadly categorized as *white box* and *black 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. Black box attacks are further classified into score-based, transfer-based and decision-based attacks. Score-based attacks generate adversarial examples using the class probabilities or confidence score of the target models. Although score-based attacks do not require detailed model knowledge, the assumption of availability of confidence scores is not realistic. Transfer-based attacks rely on training substitute models with synthetic training data, which is inefficient and computationally expensive. Decision-based attacks generate adversarial examples by observing the top label predicted by the target model and are very realistic.

In this work, we focus on the *hard-label black box*[1] setting in which the adversary crafts adversarial inputs using only the top label predicted by the target model. Compared to attacks in prior literature, hard-label black box attacks (1) requires no information about the target model's architecture, gradients or even class probability scores, (2) requires no access to training data or substitute models and (3) are more practical in real-world setting. Due to these constraints, generating adversarial examples under this setting is highly challenging. Besides, none of the attacks proposed in previous works will work in this setting. We tackle this challenging and highly realistic setting by utilizing a population-based optimization procedure that optimizes the objective function by querying the target model and observing the hard-label outputs only. We verify the grammatical correctness and fluency of generated examples through automatic and human evaluation. Our main contributions are as follows:

1. We propose a novel decision-based attack setting and generate plausible and semantically similar adversarial examples for text classification and entailment tasks.
2. Our mechanism successfully generates adversarial examples in a hard-label setting without relying on any sort of training data knowledge or substitute models.
3. Our proposed attack makes use of population-based optimization procedure which maximizes the overall semantic similarity between the original and the adversarial text.
4. In comparison to previous attack strategies, our attack achieves a higher success rate and lower perturbation rate that too in a highly restricted setting.

[1]Code: github.com/RishabhMaheshwary/hard-label-attack

The *hard label black box* setting has been explored recently in computer vision (Brendel, Rauber, and Bethge 2018; Cheng et al. 2018) but to the best of our knowledge we are first to explore it for NLP domain.

Related Work

White-box attacks: Most existing attack strategies rely on the gradient information of the loss with respect to input to generate an attack. HotFlip (Ebrahimi et al. 2017) flips a character in the input which maximizes the loss of the target model. Following this, (Liang et al. 2017) used gradient information to find important positions and introduced character level perturbations (insertion, deletion and replacement) on those positions to generate attacks. Inspired by this, (Wallace et al. 2019) does a gradient-guided search over words to find short trigger sequences. These triggers when concatenated with the input, forces the model to generate incorrect predictions. On similar lines, (Li et al. 2018) computed the gradient of loss function with respect to each word to find important words and replaced those with similar words. All the above attacks require access to the detailed model information, which is not realistic.

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. At first, (Gao et al. 2018) introduced DeepwordBug which generates character level perturbations on the important words in the input. Later, (Zhang et al. 2019) used Markov chain Monte Carlo sampling approach to generate adversarial inputs. Then (Ren et al. 2019) used saliency based word ranking to find important words and replaced those with synonyms from WordNet (Miller 1995). On similar lines, (Jin et al. 2019) proposed TextFooler which substitutes important word with synonyms. Unlike the above strategies, the work in (Alzantot et al. 2018; Zang et al. 2020) use other optimization procedures to craft adversarial inputs. The target label prediction probability is used as an optimization criteria at each iteration of the optimization algorithm.

Transfer-based attacks: Transfer-based attacks rely on information about the training data on which the target models are trained. Prior attacks (Vijayaraghavan and Roy 2019) train a substitute model to mimic the decision boundary of target classifier. Adversarial attacks are generated against this substitute model and transferred to target model. Transfer-based attacks are expensive to apply as they require training a substitute model. It also relies on the assumption that adversarial examples successfully transfer between models of different architectures.

Decision-based attacks: Decision-based attacks only depends on the top predicted label of the target classifier. Compared to all the above attack strategies, generating adversarial examples in this strategy is most challenging. The only relevant prior decision-based attack (Zhao, Dua, and Singh 2017) uses Generative Adversarial Network (GANs) which are very hard to train and require access to training data.

Problem Formulation

Let $\mathbf{F} : \mathcal{X} \rightarrow \mathcal{Y}$, be a target model that classifies an input text sequence $\mathcal{X}$ to a set of class labels $\mathcal{Y}$. Our aim is to craft an adversarial text sequence $\mathcal{X}^*$ that is misclassified by the target model i.e. $\mathbf{F}(\mathcal{X}) \neq \mathbf{F}(\mathcal{X}^*)$ and is semantically similar to the original input $\mathcal{X}$. We obtain $\mathcal{X}^*$ by solving the following constrained-optimization problem:

$$\max_{\mathcal{X}^*} \mathcal{S}(\mathcal{X}, \mathcal{X}^*) \quad s.t. \quad \mathcal{C}(\mathbf{F}(\mathcal{X}^*)) = 1 \tag{1}$$

where the function $\mathcal{S}$ computes the semantic similarity between $\mathcal{X}$ and $\mathcal{X}^*$. $\mathcal{C}$ is an adversarial criteria that equals to 1 if $\mathcal{X}^*$ is out of the target model's decision boundary and 0 otherwise. The above equation can be reformulated as:

$$\max_{\mathcal{X}^*} \mathcal{S}(\mathcal{X}, \mathcal{X}^*) \; + \; \delta(\mathcal{C}(\mathbf{F}(\mathcal{X}^*)) = 1) \tag{2}$$

where $\delta(x)$ = 0 if x is true, otherwise $\delta(x) = -\infty$. We can obtain an adversarial sample $\mathcal{X}^*$ with minimal perturbation by optimizing the objective function in the equation 2. Note that $\mathcal{C}$ is a discontinuous function as the model outputs hard-labels only. This also makes the objective function in equation 2 discontinuous and difficult to optimize.

Proposed Attack

In a hard label black-box setting, the attacker has no access to model's gradients, parameters or the confidence scores of the target model. Further, the attacker does not have access to the training data on which the target models are trained. To generate a successful attack, we formulate this setting as a constrained optimization problem as shown in equation 2. The equation 2 optimizes the semantic similarity by querying and observing the final decisions of the target model. Moreover, the outputs of the target model are insensitive to small perturbations as the model returns hard labels only thus posing an ever bigger challenge. We propose a three step strategy to solve this problem. (A) Initialisation — Initialize $\mathcal{X}^*$ outside the target model's decision boundary, (B) Search Space Reduction — Moves $\mathcal{X}^*$ close to the decision boundary and (C) Population-based optimization — Maximizes semantic similarity between $\mathcal{X}$ and $\mathcal{X}^*$ until $\mathcal{X}^*$ is on the target model's decision boundary (Figure 1).

Initialisation

In order to generate an adversarial example $\mathcal{X}^*$, which is semantically similar to original input $\mathcal{X}$, we restrict the replacement of each word with its top 50 synonyms from the counter-fitted embedding space (Mrkšić et al. 2016). Synonyms with part-of-speech (POS) tag different from the original word are filtered out. This ensures that the synonym fits within the context and the sentence is grammatically correct. To optimize the objective function in equation 2, $\mathcal{X}^*$ must satisfy the adversarial criteria $\mathcal{C}$. Therefore, $\mathcal{X}^*$ is initialised with a sample that is already out of the target model's decision boundary. This is done by substituting a word in $\mathcal{X}$ with a synonym, sampled randomly from its synonym set $Syn(x_i)$. The above step is repeated for other words in $\mathcal{X}$ until $\mathcal{X}$ moves out of target's model decision boundary or

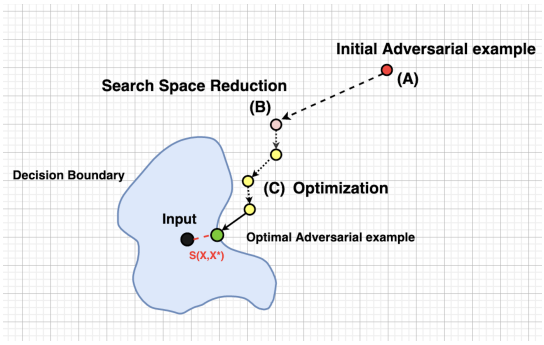

Figure 1: Overview of proposed strategy. (A) Adversarial example obtained after initialisation (B) Adversarial sample after search space reduction (C) Optimization steps

30% of the words in $\mathcal{X}$ has been substituted (Algorithm 1, lines 3-7). Note that we do not replace a word by a random word or Out of Vocabulary (OOV) token as such a replacement can highly alter the semantics of the text.

Search Space Reduction

Though population-based optimization algorithms are powerful combinatorial optimization techniques, they still slow down and converge to local optima if the size of the search space is large. As shown in Figure 2, substituting more synonyms in $\mathcal{X}^*$ increases the search space exponentially. Therefore, in this step we reduce the substitution count in $\mathcal{X}^*$ by replacing some of the synonyms back with their respective original words. Following steps are used to reduce the substitution count in $\mathcal{X}^*$:

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 initialisation. Each synonym w_i is replaced back with its original counterpart x_i (Algorithm 1, line 8-10).
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 (Algorithm 1, line 11-13)
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 (Algorithm 1, line 14-17).

This can be viewed as moving the initial sample $\mathcal{X}^*$ close to the decision boundary of the target model. This process is highly effective as it not only speeds up the optimization algorithm but also prevents it from converging to local optima.

Population based Optimization

In this section we provide a brief overview of the Genetic Algorithm (GA) and explain the working of our proposed optimization procedure in detail.

Algorithm 1 Initialisation and Search Space Reduction

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

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**
8: **for** i **in** $indices$ **do**
9: $\quad \mathcal{X}_i \leftarrow Replace\ w_i\ with\ x_i\ in\ \mathcal{X}^*$
10: $\quad scr_i \leftarrow Sim(\mathcal{X}, \mathcal{X}_i)$
11: $\quad$ **if** $\mathcal{C}(\mathbf{F}(\mathcal{X}_\mathbf{i})) = \mathbf{1}$ **then**
12: $\quad\quad Scores.insert(scr_i, x_i)$
13: $Sort\ Scores\ by\ scr_i$
14: **for** x_i **in** $Scores$ **do**
15: $\quad \mathcal{X}_t \leftarrow Replace\ w_i\ with\ x_i\ in\ \mathcal{X}^*$
16: $\quad$ **if** $\mathcal{C}(\mathbf{F}(\mathcal{X}_\mathbf{t})) = \mathbf{0}$ **then**
17: $\quad\quad$ **break**
18: $\quad \mathcal{X}^* \leftarrow \mathcal{X}_t$
19: **return** $\mathcal{X}^*$ // $After\ search\ space\ reduction$

Overview

Genetic Algorithm (GA) is a search based optimization technique that is inspired by the process of natural selection — the process that drives biological evolution. GA starts with an initial population of candidate solutions and iteratively evolves them towards better solutions. At each iteration, GA uses a fitness function to evaluate the quality of each candidate. High quality candidates are likely to be selected for generating the next set of candidates through the process of *crossover* and *mutation*. GA has the following four steps:

1. **Initialisation:** GA starts with an initial set of candidates.
2. **Selection:** Each candidate is evaluated using a fitness-function. Two candidates (*parents*) are selected based upon their fitness values.
3. **Crossover:** The selected *parents* undergoes crossover to produce the next set of candidates.
4. **Mutation:** The new candidates are mutated to ensures diversity and better exploration of search space. Steps 2-4 are repeated for a specific number of iterations.

We choose GA as an optimization procedure because it's directly applicable to discrete input space. Besides, GA is more intuitive and easy to apply in comparison to other population-based optimization methods. The method proposed in (Alzantot et al. 2018) uses the probability scores of the target label for optimizing GA in each iteration of the optimization step. *However, in a hard label black box setting such an optimization strategy will fail due to unavailability of probability scores. In this work, GA is used with a completely different motive — we maximize the semantic similarity between two text sequences.* This improves the overall attack success rate and lowers the perturbation percentage

that too in a hard label setting where none of the attacks proposed in previous works will work.

Optimization Procedure

Given that the adversarial criteria $\mathcal{C}$ is satisfied for $\mathcal{X}^*$, we now maximize the semantic similarity between $\mathcal{X}$ and $\mathcal{X}^*$ by optimizing equation 2. This optimization is achieved by replacing each substituted synonym in $\mathcal{X}^*$ back with the original word or by a better synonym (of the original word) that results in higher overall semantic similarity. We now define three operations *mutation*, *selection* and *crossover* that constitutes one iteration of the optimization step.

Mutation Given input $\mathcal{X}^* = \{x_1, w_2, w_3, x_4, x_5 ... x_n\}$ and $idx \in [0, n]$, the position where mutation needs to be applied, x_i is the original word and w_{idx} is the synonym substituted in Algorithm 1. 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. To achieve this, first the substituted synonym w_{idx} in $\mathcal{X}^*$ is replaced back with the original word x_{idx}. If the new text sequence satisfies the adversarial criteria, than x_{idx} is selected as the final replacement for w_{idx}. Otherwise, the substituted synonym w_{idx} is replaced with each synonym from the synonym set $Syn(x_{idx})$. This results in a set $\mathcal{T} = \{\mathcal{X}_1^*, \mathcal{X}_2^* ... \mathcal{X}_l^*\}$ where l in the number of synonyms of x_{idx} and $\mathcal{X}_j^* = \{x_1, w_2, s_{idx}, x_4, x_5 ... x_n\}$ where $j \in [1, l]$ is the text sample obtained after replacing w_{idx} with a synonym s_{idx} from the set $Syn(x_{idx})$. The generated samples which do not satisfy the adversarial criteria are filtered out from $\mathcal{T}$. From the remaining samples, all the samples which improves the overall semantic similarity score (equation 3) with the original input $\mathcal{X}$ is selected.

$$Sim(\mathcal{X}, \mathcal{X}_j^*) >= Sim(\mathcal{X}, \mathcal{X}^*) \quad for\ \mathcal{X}_j^* \in \mathcal{T} \tag{3}$$

If there are multiple samples in $\mathcal{T}$ which improves the semantic similarity than the sample with the highest semantic similarity score is selected as the final mutated sample.

$$candidate = \underset{\mathcal{X}_j^* \in \mathcal{T}}{\arg\max}\ Sim(\mathcal{X}, \mathcal{X}_j^*) \tag{4}$$

Note that for point 6 in optimization steps, the input to mutation will be a candidate from population set $\mathcal{P}^i$.

Selection Given a population set $\mathcal{P}^i = \{c_0^i, c_1^i, c_2^i ... c_\mathcal{K}^i\}$ where $\mathcal{K}$ represents the population size. This step samples two candidates c_p^i, c_q^i where $p, q \in [0, \mathcal{K}]$ based upon the value assigned by the fitness function. As we optimize the semantic similarity of an adversarial sequence with the original input we take the semantic similarity between the adversarial and the original text as our fitness function.

$$z_y = Sim(\mathcal{X}, c_y^i) \quad where\ c_y^i \in \mathcal{P}^i; y \in [0, \mathcal{K}] \tag{5}$$

This will allow candidates with higher semantic similarity scores to be selected as *parents* for the crossover step. c_p^i, c_q^i are sampled from $\mathcal{P}^i$ with probability proportional to $\phi(z)$.

$$\phi(z) = \frac{exp(z)}{\sum_{y=0}^{\mathcal{K}} exp(z_y)} \tag{6}$$

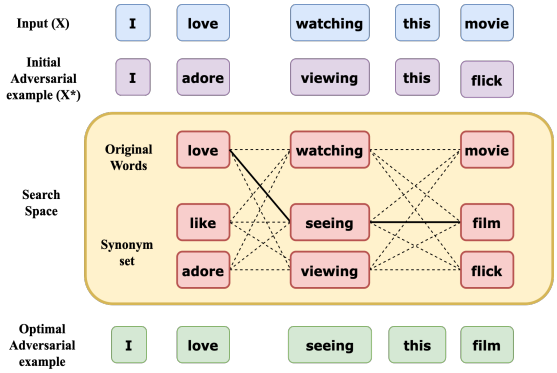

Figure 2: Search space of an adversarial sample $\mathcal{X}^*$. Dotted lines shows all possible combinations. Bold lines shows the selected combination which has the highest semantic similarity with $\mathcal{X}$ and satisfies the adversarial criteria $\mathcal{C}$.

Crossover Given c_p^i, c_q^i this step generates a new $candidate$ text sequence which satisfies the adversarial criteria. It randomly selects a word for each position of $candidate$ from c_p^i and c_q^i. Crossover is repeated multiple times to ensure exploration of various combinations in the search space.

$$c_p^i = \{u_0^1, u_1^1 ... u_n^1\} \tag{7}$$

$$c_q^i = \{u_0^2, u_1^2 ... u_n^2\} \tag{8}$$

$$candidate = \{rand(u_0^1, u_0^2) ... rand(u_n^1, u_n^2)\} \tag{9}$$

where u_0^1, u_0^2 represents the first word in c_p^i and c_q^i respectively and $rand(u_0^1, u_0^2)$ randomly selects a word.

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 $\mathcal{P}^{i+1}$.

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

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

The steps 3 to 6 are than repeated for the next population set $\mathcal{P}^{i+1}$. The maximum number of iterations T for the steps 3-6 are set to 100. Further, each index of the substituted synonym is allowed to be mutated at most $\lambda = 25$ times as it avoids the GA to converge to local optimum.

Experiments

We perform experiments across *seven* benchmark datasets on *five* target models and compare our proposed attack strategy with *seven* state-of-the-art baselines.

Datasets

AG News: A multiclass news classification dataset. The description and title of each article is concatenated following (Zhang, Zhao, and LeCun 2015).
Yahoo Answers: A document level topic classification dataset. The question and top answer are concatenated following (Zhang, Zhao, and LeCun 2015).
MR: A sentence level binary classification of movie reviews (Pang and Lee 2005).
IMDB: A document level binary classification dataset of movie reviews (Maas et al. 2011).
Yelp Reviews: A sentiment classification dataset (Zhang, Zhao, and LeCun 2015). Reviews with rating 1 and 2 are labeled negative and 4 and 5 positive as in (Jin et al. 2019).
SNLI: A dataset consisting of hypothesis and premise sentence pairs. (Bowman et al. 2015).
MultiNLI: A multi-genre NLI corpus (Williams, Nangia, and Bowman 2017).

Dataset	Train	Test	Classes	Avg. Len
AG News	120K	7.6K	4	43
Yahoo	12K	4K	10	150
MR	9K	1K	2	20
IMDB	12K	12K	2	215
Yelp	560K	18K	2	152
SNLI	120K	7.6K	3	8
MultiNLI	12K	4K	3	10

Table 1: Statistics of all datasets

Target Models We attacked WordCNN (Kim 2014), WordLSTM (Hochreiter and Schmidhuber 1997) and BERT base-uncased (Devlin et al. 2018) for text classification. For WordLSTM, a single layer bi-direction LSTM with 150 hidden units was used. In WordCNN windows of sizes 3, 4 and 5 each having 150 filters was used. For both WordCNN and WordLSTM a dropout rate of 0.3 and 200 dimensional Glove word embedding were used. For textual entailment task, we attacked ESIM (Chen et al. 2016), InferSent (Conneau et al. 2017) and BERT base-uncased. The original accuracies of all the models are shown in Table 2.

Baselines

PSO: A score-based attack that uses sememe-based substitution and particle swarm optimization (Zang et al. 2020).
TextFooler: It uses target model confidence scores to rank words and replaces those with synonyms (Jin et al. 2019).
PWWS: It ranks word based on model confidence scores and finds substitutes using WordNet (Ren et al. 2019).
TextBugger: It first finds important sentences using the confidence scores of the target model and replaces words in those sentences with synonyms (Li et al. 2018).
Genetic Attack: It is a score-based attack which crafts attacks using a population-based optimization algorithm (Alzantot et al. 2018).
GANs: It is a decision-based attacking strategy that generates adversarial examples using GANs on textual entailment task (Zhao, Dua, and Singh 2017).
DeepRL: It relies on substitute models and reinforcement learning to generate attacks (Vijayaraghavan and Roy 2019)

Evaluation Metrics We use *after attack accuracy* to evaluate the performance of our proposed attack. It refers to the accuracy of the target model obtained on the generated adversarial examples. High difference between the original and after attack accuracy represents a more successful attack. We use *perturbation rate* and *grammatical correctness* to evaluate the quality of generated adversarial examples. *Perturbation rate* refers to the number of words substituted in the input to generate an adversarial example. A higher perturbation rate degrades the quality of generated adversarial example. For *grammatical correctness* we record the grammatical error rate increase between the original and the final adversarial example. We used Language-Tool[1] to calculate the grammatical error rate of each generated adversarial text. The adversarial examples with very high perturbation rate ($> 25\%$) are filtered out. For all the evaluation metrics, we report the average score across all the generated adversarial examples on each dataset. We also conducted human evaluation to verify the quality of adversarial examples.

Experimental Settings We use Universal Sequence Encoder (USE) (Cer et al. 2018) to compute the semantic similarity between the original and adversarial example. We filter out stop words using NLTK and use Spacy for POS tagging. The target models are attacked on a set of 1000 examples,

[1] https://www.languagetool.org/

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.

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.

sampled from the test set of each dataset. To ensure fair comparison these are the same set of examples used in (Alzantot et al. 2018; Jin et al. 2019). The parameters of GA, $\mathcal{K}$ and λ were set to 30 and 25 respectively. The maximum iterations T is set to 100. From each dataset, we held-out 10% data for validation set, for tuning the hyper-parameters.

Attack Performance Table 2-3 shows that our attack achieves more than 90% success rate on classification and entailment tasks. In comparison to TextFooler, our attack reduces both the perturbation rate and after attack accuracy by atleast 33% and 32% respectively across all datasets and target models. Further, it reduces the grammatical error rate by 27%. When compared to PSO, on IMDB and SNLI, our attack achieves 10% more success rate with lesser perturbation rate that too in a highly restricted setting. Table 3 shows that our attack outperforms other baselines in terms of success rate and perturbation rate. Table 4 shows the adversarial examples generated effectively on BERT.

Ablation Study

Importance of Search Space Reduction: To study the significance of search space reduction we executed GA directly after the initialisation step. Table 5 shows the results obtained on BERT. On an average the perturbation rate increased by 1.2% and the semantic similarity dropped by 0.1. This is because GA based optimization converges to a local optimum in most of the cases when search space is large. Further on an average, GA optimization procedure slows down by atleast five times due to large search space.

Importance of Genetic Algorithm: To study the importance of GA, we compare perturbation rate and semantic similarity both with and without GA based optimization. Table 5 demonstrates the results obtained on BERT. By using GA, the perturbation rate is reduced by 4.4% and the semantic similarity improves by 0.16. Table 5 also shows the perturbation rate and semantic similarity after initialisation step. On an average, the perturbation is 20% higher and and semantic similarity is 0.25 lower. This highlights the combined effect of both GA and search space reduction step to find optimal adversarial examples. We obtained similar results across all datasets and target models.

Analysis

Importance of Synonym based Initialisation: During initialisation, we replace each word in $\mathcal{X}$ randomly with it's synonym, sampled from top 50 synonyms in the counter-fitted space. To verify its effectiveness, we remove this constraint and allow the word to be replaced with a random word from the vocabulary. Results achieved on BERT are shown in Table 6. On an average, the semantic similarity decreases by 0.1 and the perturbation rate increases by 2.4%.

Transferability: An adversarial example is called *transferable* if it's generated against a particular target model but successfully attacks other target models as well. We evaluate the transferability of adversarial attacks generated on IMDB and SNLI datasets. The results are shown in Table 7. A lower

Examples	Prediction
Highly [Exceedingly] watchable stuff.	**Positive → Negative**
It's weird, wonderful, and not **necessarily [definitely]** for kids.	**Negative → Positive**.
Could i use both Avast and Avg to protect my **computer [machinery]**? I recommend the free version of Avg antivirus for home users.	**Technology → Music.**
Premise: Larger ski resorts are 90 minutes away. **Hypothesis**: If we travel for 90 minutes, we could arrive at larger **ski [skating]** resorts.	**Entailment → Neutral.**
Premise: A portion of the nation's income is saved by allowing for capital investment. **Hypothesis**: The nation's income is divided into portions **[fractions]**.	**Entailment → Neutral.**

Table 4: Adversarial samples generated on BERT. The actual word is bold and substituted word are bold and in square brackets.

Ablation Study	Metric	IMDB	Yelp	SNLI
no SSR and GA	Pert%	20.1	24.3	34.6
	Sim	0.6	0.57	0.22
only GA	Pert%	4.2	7.2	18.0
	Sim	0.81	0.74	0.37
only SSR	Pert%	6.0	9.7	21.0
	Sim	0.80	0.74	0.26
both SSR and GA	Pert%	**3.1**	**6.7**	**16.7**
	Sim	**0.89**	**0.80**	**0.45**

Table 5: Importance of the search space reduction (SSR) and GA step. Pert.% is the average perturbation rate and Sim is the average semantic similarity.

Dataset	Pert.%		Sim	
	with ran	w/o ran	with ran	w/o ran
IMDB	3.7	**3.1**	0.81	**0.89**
Yelp	15.3	**6.7**	0.72	**0.80**
MR	18.0	**10.7**	0.53	**0.67**

Table 6: Effect of random initialisation (ran). Pert.% is average perturbation and Sim is the average semantic similarity

accuracy of a target model demonstrates high transferability. Adversarial examples generated by our attacks show better transferability when compared to prior attacks.

Adversarial Training: We generated adversarial examples using the 10% samples from the training set of IMDB and SNLI datasets. We augmented the generated adversarial examples with the original training set of the respective datasets and re-trained BERT. We than again attacked BERT with our attack strategy. The results are shown in Figure 3. The after attack accuracy and perturbation rate increases by 15% and 10% respectively. This shows that by augmenting adversarial samples to the training data, the target models becomes more robust to attacks.

Human Evaluation: To validate and access the quality of adversarial samples, we randomly sampled 25% of the adversarial examples from IMDB and SNLI datasets. The true class labels of these samples were kept hidden and the evaluators were asked to classify them. We found 96% adversarial examples in IMDB and 92% in SNLI having the same classification label as that of their original samples. The evaluators were asked to score each adversarial example on grammatical correctness and semantic similarity with the original example. They were asked to score each example from 1 to 5 based on grammatical correctness and assign a score of 0, 0.5 and 1 for semantic similarity. Table 8 shows the evaluation results of attacks generated against BERT.

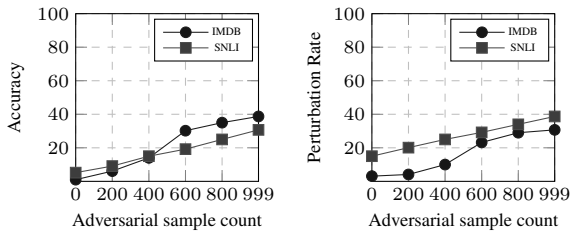

Figure 3: Demonstrates increase in after attack accuracy and perturbation as more adversarial samples are augmented.

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.

Evaluation criteria	IMDB	SNLI
Grammatical Correctness	4.44	4.130
Semantic Similarity	0.93	0.896

Table 8: Demonstrates scores given by evaluators

Conclusion

In this work, we propose a novel decision-based attack that utilizes population-based optimization algorithm to craft plausible and semantically similar adversarial examples by observing only the topmost predicted label. In a hard label setting, most of the attacks proposed in previous works will not work as well. Extensive experimentation across multiple target models and benchmark datasets on several state-of-the-art baselines demonstrate the efficacy and the effectiveness of our proposed attack. In comparison to prior attack strategies, our attack achieves a higher success rate and lower perturbation rate that too in a highly restricted setting.

References

Abdel-Hamid, O.; Mohamed, A.-r.; Jiang, H.; Deng, L.; Penn, G.; and Yu, D. 2014. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing* 22(10): 1533–1545.

Alzantot, M.; Sharma, Y.; Elgohary, A.; Ho, B.-J.; Srivastava, M.; and Chang, K.-W. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998* .

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* .

Brendel, W.; Rauber, J.; and Bethge, M. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *International Conference on Learning Representations*.

Cer, D.; Yang, Y.; Kong, S.-y.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* .

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; Jiang, H.; and Inkpen, D. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038* .

Cheng, M.; Le, T.; Chen, P.-Y.; Yi, J.; Zhang, H.; and Hsieh, C.-J. 2018. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457* .

Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; and Bordes, A. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364* .

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Ebrahimi, J.; Rao, A.; Lowd, D.; and Dou, D. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751* .

Gao, J.; Lanchantin, J.; Soffa, M. L.; and Qi, Y. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, 50–56. IEEE.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8): 1735–1780.

Jin, D.; Jin, Z.; Zhou, J. T.; and Szolovits, P. 2019. Is bert really robust? natural language attack on text classification and entailment. *arXiv preprint arXiv:1907.11932* .

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Li, J.; Ji, S.; Du, T.; Li, B.; and Wang, T. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271* .

Liang, B.; Li, H.; Su, M.; Bian, P.; Li, X.; and Shi, W. 2017. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006* .

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, 142–150. Association for Computational Linguistics.

Maheshwary, S.; Ganguly, S.; and Pudi, V. 2017. Deep secure: A fast and simple neural network based approach for user authentication and identification via keystroke dynamics. In *IWAISe: First International Workshop on Artificial Intelligence in Security*, 59.

Maheshwary, S.; and Misra, H. 2018. Matching resumes to jobs via deep siamese network. In *Companion Proceedings of the The Web Conference 2018*, 87–88.

Maheshwary, S.; and Pudi, V. 2016. Mining keystroke timing pattern for user authentication. In *International Workshop on New Frontiers in Mining Complex Patterns*, 213–227. Springer.

Miller, G. A. 1995. WordNet: a lexical database for English. *Communications of the ACM* 38(11): 39–41.

Mrkšić, N.; Séaghdha, D. O.; Thomson, B.; Gašić, M.; Rojas-Barahona, L.; Su, P.-H.; Vandyke, D.; Wen, T.-H.; and Young, S. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892* .

Pang, B.; and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, 115–124. Association for Computational Linguistics.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 506–519.

Ren, S.; Deng, Y.; He, K.; and Che, W. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1085–1097.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* .

Vijayaraghavan, P.; and Roy, D. 2019. Generating Black-Box Adversarial Examples for Text Classifiers Using a Deep Reinforced Model. *arXiv preprint arXiv:1909.07873* .

Wallace, E.; Feng, S.; Kandpal, N.; Gardner, M.; and Singh, S. 2019. Universal adversarial triggers for attacking and analyzing NLP. *arXiv preprint arXiv:1908.07125* .

Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426* .

Young, T.; Hazarika, D.; Poria, S.; and Cambria, E. 2018. Recent trends in deep learning based natural language pro-

cessing. *ieee Computational intelligenCe magazine* 13(3): 55–75.

Zang, Y.; Qi, F.; Yang, C.; Liu, Z.; Zhang, M.; Liu, Q.; and Sun, M. 2020. Word-level Textual Adversarial Attacking as Combinatorial Optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6066–6080.

Zhang, H.; Zhou, H.; Miao, N.; and Li, L. 2019. Generating Fluent Adversarial Examples for Natural Languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5564–5569.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.

Zhao, Z.; Dua, D.; and Singh, S. 2017. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342* .