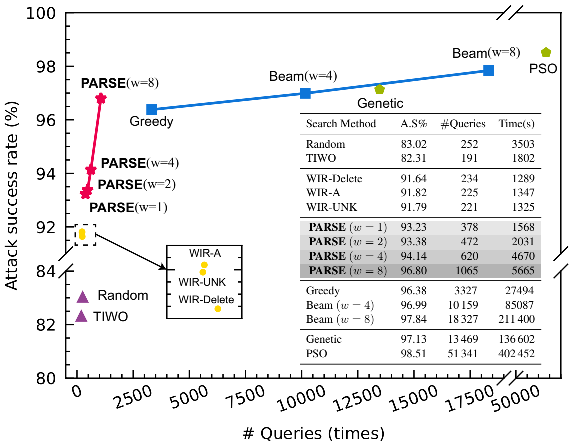
PARSE: An Efficient Search Method for Black-box Adversarial Text Attacks

Pengwei Zhan[§‡], Chao Zheng[§], Jing Yang[§]*, Yuxiang Wang[§], Liming Wang[§], Yang Wu[§‡], Yunjian Zhang[§‡]
[§]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[‡]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{zhanpengwei,zhengchao1135,yangjing,wangyuxiang}@iie.ac.cn
{wangliming,wuyang0419,zhangyunjian}@iie.ac.cn

Abstract

Neural networks are vulnerable to adversarial examples. The adversary can successfully attack a model even without knowing model architecture and parameters, i.e., under a black-box scenario. Previous works on word-level attacks widely use word importance ranking (WIR) methods and complex search methods, including greedy search and heuristic algorithms, to find optimal substitutions. However, these methods fail to balance the attack success rate and the cost of attacks, such as the number of queries to the model and the time consumption. In this paper, We propose **PA**thological wo**R**d **S**aliency s**E**arch (PARSE) that performs the search under dynamic search space following the subarea importance. Experiments show that PARSE can achieve comparable attack success rates to complex search methods while saving numerous queries and time, e.g., saving at most 74% of queries and 90% of time compared with greedy search when attacking the examples from Yelp dataset. The adversarial examples crafted by PARSE are also of high quality, highly transferable, and can effectively improve model robustness in adversarial training.



Search Method	A.S%	#Queries	Time(s)
Random	83.02	252	3503
TIWO	82.31	191	1802
WIR-Delete	91.64	234	1289
WIR-A	91.82	225	1347
WIR-UNK	91.79	221	1325
PARSE $(w = 1)$	93.23	378	1568
PARSE $(w = 2)$	93.38	472	2031
PARSE $(w = 4)$	94.14	620	4670
PARSE $(w = 8)$	96.80	1065	5665
Greedy	96.38	3327	27494
Beam $(w = 4)$	96.99	10 159	85087
Beam $(w = 8)$	97.84	18 327	211 400
Genetic	97.13	13 469	136 602
PSO	98.51	51 341	402 452

Figure 1: Average #Queries vs. Attack success rate (%) when attacking TextCNN on 500 examples from Yelp in HowNet. Increasing the beam width w in PARSE effectively increases the attack success rate while just costing a few more #queries and time. PARSE (w=8) outperforms Greedy search while taking only 32% of queries and 20% of time. The complete results are in Table 2.

1 Introduction

Neural networks have achieved remarkable success in various NLP tasks while being vulnerable to adversarial examples. The adversary can craft adversarial examples, which contain noise that is imperceptible to human but can mislead the model decision, even without knowing the model architecture and parameters. Under such black-box scenario, word-level attacks have been more focused on by recent studies for the flexibility of the attack and the high quality generated examples (Gao et al., 2018; Alzantot et al., 2018; Jin et al., 2020; Li et al., 2018; Garg and Ramakrishnan, 2020a; Ebrahimi et al., 2018). Word-level attacks can flexibly fit the grammar and semantics constraints by changing the similarity and semantics threshold when filtering candidate substitutions, and the generated adversarial examples will not be detected by a spell checker (Ebrahimi et al., 2018; Iyyer et al., 2018) or substantially damage the overall semantic and logic of the sentence (Jia and Liang, 2017; Liang et al., 2018). High-quality adversarial examples ensure the attacks are imperceptible to human and can be used to learn the robustness of models better.

To better explain our contribution to the word-level adversarial attack, we would like first to define the word-level attack as a combinatorial optimization problem, which is similar to (Yoo et al., 2020; Morris et al., 2020b,a). Under this setting, a word-level adversarial attack method can be decomposed into *Search Space* and *Search Method*. The search space gives all the possible substitutions that meet the similarity and semantics requirements for the target words, i.e., decides what words the target words can be transformed into. The search method

*Corresponding Author.

Proceedings of the 29th International Conference on Computational Linguistics, pages 4776–4787
October 12–17, 2022.

is the search strategy to perform the attack, i.e., decides which words to be transformed (target words) and what words should be transformed into (pick from the search space). Search method is the most significant part of an attack method, as the exponential nature of the search space makes inefficiency search method difficult to attack large-scale and long examples. Therefore, we fix the search space and only focus on the search methods in this paper.

Various search methods for word-level black-box attacks have been proposed, divided into simple methods, including the variants of Word Importance Ranking (WIR) methods (Gao et al., 2018; Li et al., 2018; Jin et al., 2020; Li et al., 2020), and complex methods, including greedy search (Pruthi et al., 2019; Li et al., 2021), beam search (Ebrahimi et al., 2018), genetic algorithm (Alzantot et al., 2018), and Particle Swarm Optimization (PSO) algorithm (Zang et al., 2020). WIR methods search substitutions for each word in the descending order of word importance scores, which is faster than other complex search methods, while is poor in attack success rate. Other search methods can always achieve better attack success rates, while they need more queries to the target model and time, as they directly search on the search space of the entire sentence. Previous works fail to balance well between performance and efficiency.

In this paper, to achieve higher attack success rates with fewer queries and time in word-level black-box attacks, we propose a search method called **PA**thological wo**R**d **S**aliency s**E**arch (PARSE). PARSE separates the entire sentence into multiple subarea according to the stability of words and searches in the descending order of the subarea importance. Therefore, PARSE avoids directly searching on the huge search space of the entire sentence and reduces the cost of attacks. Searching on each subarea instead of on each word like WIR methods also makes PARSE less likely to be stuck in local optima. Extensive experiments demonstrate that PARSE achieves comparable attack success rates to complex search methods while saving numerous queries and time, e.g., saving at most 74% of queries and 90% of time compared with greedy search when attacking the Yelp dataset (Zhang et al., 2015). Figure 1 shows the performance comparisons. The major contributions of this paper are summarized as follows:

- We define the stability of words in adversarial attacks and explain the ineffective of WIR methods from the view of word stability.
- We propose PARSE, a search method for word-level black-box adversarial attacks that performs search under dynamic search space following the subarea importance.
- Experiments show that PARSE achieves comparable attack success rates to complex methods while saving numerous queries and time.

2 Related Works

Adversarial attack. Inspired by the early works on adversarial attacks that mainly focus on the field of computer vision (CV) (Goodfellow et al., 2015; Papernot et al., 2016; Moosavi-Dezfooli et al., 2016; Carlini and Wagner, 2017), various methods to attack language models are proposed (Li et al., 2018; Gao et al., 2018; Garg and Ramakrishnan, 2020b; Miyato et al., 2017; Gong et al., 2018). Unlike the image, which is differentiable as pixels are continuous values, the discrete text is not differentiable. Therefore, the adversarial attacks in NLP tasks are more appropriately described as combinatorial optimization problems, which seek to find optimal substitutions in the search space (Yoo et al., 2020; Morris et al., 2020b,a).

Search Method. Although various adversarial attack frameworks focusing on the NLP tasks are proposed, few works make a clear distinction between the search space and search method. The reported results may benefit from strong search method (Alzantot et al., 2018; Zang et al., 2020; Jia et al., 2019), which have a higher time complexity and need more queries to the model, or the less restrictive search space (Pruthi et al., 2019; Gao et al., 2018; Ebrahimi et al., 2018; Li et al., 2018; Jin et al., 2020; Li et al., 2020), which does not consider both the distance between the target words and the substitutions and the semantic of the entire perturbed sentence. In this paper, we only focus on the search methods and benchmark all search methods under the same search space.

3 PARSE

3.1 Textual Adversarial Example

Suppose there is a model $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$ trained by minimizing the empirical risk over all given text $\boldsymbol{X} \in \mathcal{X}$ and labels $Y \in \mathcal{Y}$ following the distribution $\mathcal{D}$:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{X},Y)\sim\mathcal{D}} \mathcal{L}\left(\mathcal{F}\left(\boldsymbol{X};\boldsymbol{\theta}\right), Y\right) \tag{1}$$

where $\boldsymbol{\theta}$ is the parameter, and $\mathcal{L}(\cdot)$ is the cross-entropy loss. An adversarial example $\boldsymbol{X}^{adv}$ crafted from a normal text $\boldsymbol{X} = (x_n)_{n\in\{1,\dots,N\}}$ can thus be defined as:

$$
\begin{aligned}
&\boldsymbol{X}^{adv} = \mathcal{O}(\boldsymbol{X}) = o(x_n)_{n\in\{1,\dots,N\}}, \\
&\text{s.t.} \quad \forall n \in \{1,\dots,N\}, \ \ \Delta x_n < \delta, \\
&\text{and} \quad \Delta \boldsymbol{X} < \varepsilon, \\
&\text{and} \quad \underset{Y\in\mathcal{Y}}{\operatorname{argmax}}\ \mathcal{P}(Y|\boldsymbol{X}^{adv}) \neq \underset{Y\in\mathcal{Y}}{\operatorname{argmax}}\ \mathcal{P}(Y|\boldsymbol{X})
\end{aligned} \tag{2}
$$

where $\mathcal{O}(\boldsymbol{X})$ means performing word-level substitution on sentence $\boldsymbol{X}$, $o(x_n)$ means substituting the word x_n with a new word from search space, if possible. Δx_n denotes the difference between x_n and $o(x_n)$, $\Delta\boldsymbol{X}$ denotes the difference between $\boldsymbol{X}$ and $\mathcal{O}(\boldsymbol{X})$, δ and ε are the maximum allowed difference of words and the overall sentence, respectively, which restrictions are imposed on search space to filter potential substitutions, $\mathcal{P}(\cdot|\cdot)$ is the posterior probability. Intuitively, (2) can be explained as the following condition. We have a finite search space that contains all possible substitutions for each word in $\boldsymbol{X}$, and the substitutions in the search space are further filtered by the restrictions, including δ and ε, which may mainly focus on the semantics and the L_p norm of embedding distance of each word and the entire sentence. These restrictions ensure that the final generated adversarial example is imperceptible to human. The search method can thus be seen as the strategy to perform $\mathcal{O}(\cdot)$, deciding the order to perform $o(\cdot)$ and the substitutions picked from the search space. When the restrictions on search space are fixed, the better search method finds the adversarial example more accurately and efficiently.

3.2 Word Importance

As gradient information is not available in the Black-box scenario, the Leave One Out (LOO) methods are proposed to obtain word importance, i.e., the word saliency. (Li et al., 2016; Gao et al., 2018; Li et al., 2018; Jin et al., 2020; Li et al., 2020). LOO methods expect to obtain word importance by comparing the model confidence of two sentences with only one word is different. Formally, the importance of words $x_i \in \boldsymbol{X}$ is defined as

$$S(x_i) = \mathcal{P}(Y_{true}|\boldsymbol{X}) - \mathcal{P}(Y_{true}|\hat{\boldsymbol{X}}_i) \tag{3}$$

where $S(\cdot)$ is the word saliency, Y_{true} is the ground-truth class, and $\hat{\boldsymbol{X}}_i = x_1 \dots \hat{x}_i \dots x_N$ is the sentence with word x_i transformed. Transforming x_i to $\hat{x}_i$ in different ways formulating three prevalent LOO methods: (i) Delete: leaving $\hat{x}_i$ blank. (ii) UNK: $\hat{x}_i =$ `[UNK]`, triggering the out of vocabulary (OOV) problem. (iii) A: $\hat{x}_i =$ `a`, replacing with a neutral word `a` that has a similar distribution across classes (Pruthi et al., 2019). Intuitively, performing the substituting operation $o(\cdot)$ on the words in the descending order of their importance should help to generate adversarial examples more efficiently (this is how the WIR search methods do), as the important word have a large impact on the model prediction.

3.3 Word Stability from the View of Words Importance Changing

To explain why the word importance fails to indicate the model concentration and why the WIR methods, which strictly follow the descending order of words importance to perform the attack, have a degenerated performance, we first try to answer:

When should a word be considered unstable?

We would like to clarify that the stability of a word is defined together with the system trying to understand the word, i.e., the stable word for a system may be an unstable word for other systems. For human, parsing a sentence is a denoising process. We can still understand a sentence even if slight noise is introduced to the sentence, e.g., changing the order of letters in a word or deleting some words in a sentence (McCusker et al., 1981; Rayner et al., 2006; Adam Drewnowski, 1978; McCusker et al., 1981; Van Orden, 1987). More importantly, we focus on the important words in the sentence and do not change our attention to the words due to the small noise. Based on this, there are few unstable words for the human reading comprehension system, as the important words we concentrate on do not change. Following this, if a system, e.g., a language model, changes attention to the words when parsing a sentence because of sufficiently slight noise, we consider the words whose importance, i.e., the attention of the system, have changed as unstable words for the system. It should be noted that, as defined in (3), the word importance is a continuous value, and if the importance of all words increases to the same extent, the attention of the system is actually not changed. Therefore, we further use a discrete value called importance ranking to define the attention, which is formed as

$$
\begin{aligned}
R(\boldsymbol{X}) &= r(x_i)_{i\in\{1,\dots,N\}} \\
&= \arg\operatorname*{sort}_{i} S(x_i)_{i\in\{1,\dots,N\}}
\end{aligned} \tag{4}
$$

where $\arg\text{sort}(\cdot)$ returns the indexes of the sorted sequence in descending order. Therefore, the stability of a word and a group of adjacent words can be defined.

Definition 1. *For a given model $\mathcal{F}$, a sentence $\boldsymbol{X} = (x_n)_{n\in\{1,\dots,N\}}$, let $t(\cdot)$ be a slight transformation that deletes the least important word in a sentence, the word importance rankings of $\boldsymbol{X}$ and $t(\boldsymbol{X})$ are $R(\boldsymbol{X}) = (r_1, r_2, \dots, r_N)$ and $R(t(\boldsymbol{X})) = (r_1^*, r_2^*, \dots, r_N^*)$, respectively, where r_i and r_i^* are the word index. If $r_i \neq r_i^*$, then the word x_{r_i} is an unstable word for model $\mathcal{F}$; otherwise, a stable word. If $\forall\, r_j \in (r_i, \dots, r_{i+n}), r_j \neq r_j^*$, the adjacent words $(x_{r_i}, \dots, x_{r_{i+n}})$ form an unstable subarea; otherwise, a stable subarea.*

Intuitively, the importance of unstable words can not accurately indicate the impact on the model prediction, as even a slight transformation is enough to shift the importance ranking of these words over the entire sentence, and the important word may become not that important. That is why the WIR methods always have poor performance. Such phenomena are probably due to the model pathology (Feng et al., 2018) as neural networks are more linear than expected and will overfit the negative log-likelihood loss to produce low-entropy distribution over classes, leading the model to overconfidence in instances outside the training data distribution (Goodfellow et al., 2015). This consequently leads to the word importance drastically changing with even the least important word being removed from the sentence, which is sufficient to bias the sentence representation from the distribution.

To show the influence of word stability for common language models of different architectures, we test the word stability of the sentence in MR and Yelp training set for LSTM, TextCNN, and DistilBERT (model architectures are detailed in §4.1). Following (3), (4), and Definition 1, we first obtain the word importance rankings with the LOO-UNK method on 500 randomly picked examples and then compare the word importance rankings between the original sentences and the sentences with the least important word removed to obtain the word stability. Table 1 shows the average stability results of five individual runs. There are relatively few unstable words on the short text on MR, with an average of 45.5% on the three models, while it will rise to 72.4% on the longer text on Yelp. We also find that the average length of an unstable subarea is very short compared with the length of the entire sentence and takes only 19.3% for MR and 7.8% for Yelp on average. Therefore, we can draw two conclusions about word stability:

(C_1) unstable words are prevalent regardless of the model architecture and take a higher proportion in a longer sentence.

(C_2) the word importance rankings are mainly swapping between the words of similar importance (as the average length of unstable subarea is short).

Dataset	Model	#input words	#unstable words	unstable word%	#unstable subarea	AVG unstable subarea length
MR	LSTM	18.25	8.14	44.60	2.23	3.49
	TextCNN	18.72	8.26	44.12	2.32	3.56
	DistilBERT	18.39	8.79	47.79	2.41	3.65
Yelp	LSTM	128.39	96.66	75.28	10.81	8.94
	TextCNN	133.29	83.94	62.97	8.77	9.57
	DistilBERT	135.86	107.14	78.86	8.61	12.51

Table 1: Statistics on the word stability of the sentence in MR and Yelp training set for different models.

3.4 Searching Strategy of PARSE

To generate adversarial examples in higher attack success rate with fewer queries and time, a search method must take the word stability (and further the (C_1) and (C_2)) into account. As the importance of unstable words fails to accurately indicate the impact on the model prediction, strictly following which to perform attack are likely to stuck in local optima. Based on this, we propose PARSE that performs beam search under dynamic search space following subarea importance. The general searching strategy of PARSE is shown in Figure 2. Specifically, PARSE starts by transforming the target sentence $\boldsymbol{X}$ with transformation $t(\cdot)$ and obtaining the stability of all words in the target sentences. PARSE treats each stable word individually while treating the adjacent unstable words, i.e., the words in the same unstable subarea, as an integration. That is, according to Definition 1, each stable word forms a stable subarea, and multiple adjacent unstable words form an unstable subarea. Therefore, the sentence is separated into multiple subarea based on the stability of words, and the entire potential search space is separated into multiple subspace. PARSE avoids being too sensitive to the importance of a single word and being affected by the inaccuracy of word importance by taking the subarea as basic elements at each search step rather than each word like other methods. The search is then performed following the descending order of

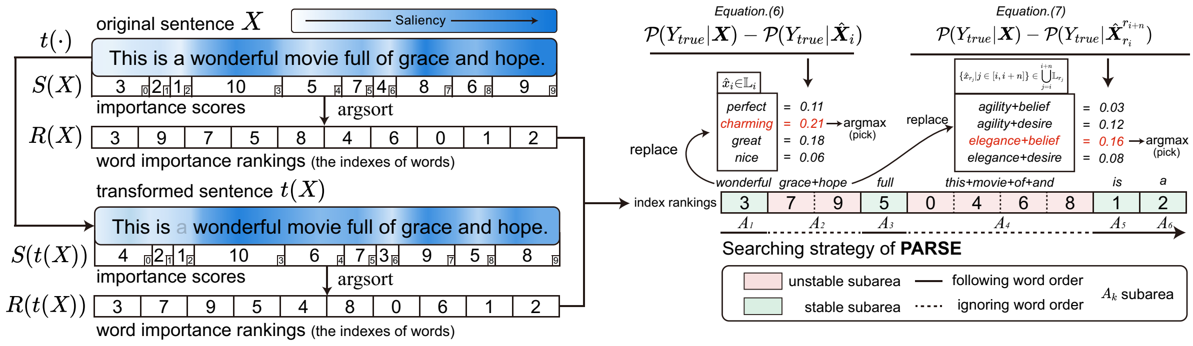


Figure 2: The general searching strategy of PARSE. We separate the entire sentence into multiple subarea according to the word stability. PARSE performs search in the descending order of subarea importance and takes each subarea as integration at each search step.

the subarea importance score, which is the average importance rankings of the words in a subarea:

$$score(\boldsymbol{A}) = \frac{1}{||\boldsymbol{A}||} \sum_{x_i \in \boldsymbol{A}} r(x_i) \tag{5}$$

where $\boldsymbol{A}$ denotes a subarea, $||\boldsymbol{A}||$ denotes the number of words in a subarea. PARSE behaves differently when encountering stable subarea and unstable subarea. To better understand the idea of PARSE, we first explain the detailed search process when beam width $w = 1$. When encountering stable subarea $\boldsymbol{A}_k$, which contains the only word $x_i \in \boldsymbol{A}_k$, we tend to find the substitution $\hat{x}_i$ that mostly reduces the model confidence from the search space:

$$\hat{x}_i = \underset{\hat{x}_i \in \mathbb{L}_i}{\arg\max} \{\mathcal{P}(Y_{true}|\boldsymbol{X}) - \mathcal{P}(Y_{true}|\hat{\boldsymbol{X}}_i)\} \tag{6}$$

where $\mathbb{L}_i$ is the potential search space for x_i under the fixed restrictions including δ and ε, and $\hat{\boldsymbol{X}}_i$ is the sentence with word x_i transformed into $\hat{x}_i$. When encountering unstable subarea $\boldsymbol{A}_k$ that contains multiple unstable words $(x_{r_i}, \ldots, x_{r_{i+n}}) \in \boldsymbol{A}_k$, we tend to find the group of substitutions $\{\hat{x}_{r_j} | j \in [i, i+n]\}$ that mostly reduces the model confidence, which equals to:

$$\underset{\{\hat{x}_{r_j}|j \in [i,i+n]\} \in \bigcup_{j=i}^{i+n} \mathbb{L}_{r_j}}{\arg\max} \{\mathcal{P}(Y_{true}|\boldsymbol{X}) - \mathcal{P}(Y_{true}|\hat{\boldsymbol{X}}_{r_i}^{r_{i+n}})\} \tag{7}$$

where $\bigcup_{j=i}^{i+n} \mathbb{L}_{r_j}$ is the potential search space for the multiple words $(x_{r_i}, \ldots, x_{r_{i+n}}) \in \boldsymbol{A}_k$, i.e., the combination of the search space of every single word, $\hat{\boldsymbol{X}}_{r_i}^{r_{i+n}}$ is the sentence with words $\{x_{r_j} | j \in [i, i+n]\}$ transformed into $\{\hat{x}_{r_j} | j \in [i, i+n]\}$. Intuitively, different from the search space of a single word $\mathbb{L}_i$, the same substitution for a single word in $\{\hat{x}_{r_j} | j \in [i, i+n]\}$ may appear many times in all possible substitution combinations in the search space of multiple words $\bigcup_{j=i}^{i+n} \mathbb{L}_{r_j}$. Thus the word order in an unstable subarea is ignored, and the words in an unstable subarea will be substituted at the same time at each search step, which helps reduce the impact of the inaccurate importance rankings of unstable words. The search is performed on every subarea in order until the generated example meets (2). When the beam width $w \neq 1$, we keep the top-w $\hat{x}_i$ or $\{\hat{x}_{r_j} | j \in [i, i+n]\}$ that the results most close to equation (6) or (7), i.e., mostly reduce the model confidence, at each search step. The details of PARSE are shown in Algorithm 1.

Algorithm 1: PARSE

input : Original sentence $\boldsymbol{X} = x_1 x_2 \ldots x_N$, Separated search space $\mathcal{A} = (\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n)$, beam width w, true label Y_{true}
output : Adversarial example $\boldsymbol{X}^{adv}$

Initialize candidate set $\mathcal{X}_{best} \leftarrow \{\boldsymbol{X}\}$
Sort $\mathcal{A}$ by the subarea score (Eq.(5)) in descending
for *all* $\boldsymbol{A}_k \in \mathcal{A}$ **do**
  reset the union of candidate set $\mathcal{X}_{all} \leftarrow \{\}$
  for *all* $\boldsymbol{X}'_j \in \mathcal{X}_{best}$ **do**
    if $\boldsymbol{A}_k$ *is stable zone* **then**
      $\mathcal{X}_{cand} \leftarrow$ {top-w sentences transformed from $\boldsymbol{X}'_j$ that $\hat{x}_i$ most close to Eq.(6)}
    else
      $\mathcal{X}_{cand} \leftarrow$ {top-w sentences transformed from $\boldsymbol{X}'_j$ that $\{\hat{x}_{r_j} | j \in [i, i+n]\}$ most close to Eq.(7)}
    $\mathcal{X}_{all} \leftarrow \mathcal{X}_{all} \cup \mathcal{X}_{cand}$
  $\mathcal{X}_{best} \leftarrow$ {top-w sentences $\boldsymbol{X}' \in \mathcal{X}_{all}$ that mostly reduce model confidence $\mathcal{P}(Y_{true}|\boldsymbol{X}) - \mathcal{P}(Y_{true}|\boldsymbol{X}')$}
  if $\exists \underset{Y \in \mathcal{Y}, \boldsymbol{X}' \in \mathcal{X}_{best}}{\arg\max} \mathcal{P}(Y|\boldsymbol{X}') \neq Y_{true}$ **then**
    return *the* $\boldsymbol{X}'$ *that mostly reduces model confidence as* $\boldsymbol{X}^{adv}$; /* Success */
return $\boldsymbol{X}$; /* Fail */

4 Experiment

4.1 Experiment Setup

Dataset. The experiments are conducted on Movie Review (MR) (Pang and Lee, 2005) and Yelp Review Polarity (Yelp) (Zhang et al., 2015). Both of them are sentiment classification tasks. For MR, the average text length is 18.49. For Yelp, the average text length is 135.66.

Model. We use TextCNN (Kim, 2014), LSTM, and DistilBERT (Sanh et al., 2019) in our experiments. More details of the model are in Appendix.

Search Space. We utilize the 300-Dimensional GloVe word vectors (Pennington et al., 2014) and HowNet (Dong and Dong, 2003) as the search space for substitutions. GloVe contains vector representations for words learned by an unsupervised learning algorithm. HowNet is a knowledge base of sememes with over 100,000 words.

Restrictions on Search Space. The possible substitutions for each word in the search space are filtered by the restrictions imposed on the search space. The substitutions picked from the search space should have the same part of speech as the original word. The similarity between the generated and original sentences measured by BERT should be larger than 90%.

Baselines. We compare PARSE with nine search methods: Random, Traverse in word order (TIWO), WIR-Delete, WIR-A, WIR-UNK, Greedy Search, Beam Search, Genetic Algorithm (Genetic) (Alzantot et al., 2018), and Particle Swarm Optimization (PSO) (Zang et al., 2020). The Random method randomly picks a word as the target word at each search step. TIWO method performs the search following the word order.

Implementation Details. For PARSE, we use the LOO-UNK to obtain the word importance. For the Genetic and PSO algorithm, the population size and the number of iterations are set to 60 and 20, respectively. All reported results are the average of five individual runs. All comparisons in our experiments are conducted under the same search space with the same restrictions on the same machine with an A5000 GPU.

4.2 Main Results

Comparisons on Performance. We perform adversarial attacks on 500 randomly picked examples, and the results on performance are shown in Table 2. Even when $w = 1$, PARSE still outperforms WIR methods on attack success rates, and #queries and time are only slightly increased. This indicates that considering the stability of words and treating the words of different stability differently in each search step helps reduce the impact of the inaccurate word importance. Compared with the complex search methods like greedy search, PARSE ($w = 8$) can achieve comparable attack success rates with fewer #queries and time, especially on the long text from Yelp. On MR, on average, PARSE ($w = 8$) needs 213 queries and 2.9 seconds for each successful attack, while greedy search needs 390 queries and 7.1 seconds. On Yelp, on average, PARSE ($w = 8$) needs 1320 queries and 13 seconds for each successful attack, while greedy search needs 4582 queries and 115 seconds. Others complex search methods even need far more queries and time. It should be noted that PARSE is more suitable for attacking long sentences as it is less affected by the increased search space compared to other complex search methods, while it can still achieve competitive results when attacking short sentences.

Comparisons under Different Search Space. We replace the search methods while maintaining the search space in TextBugger, BAE, and DeepWordBug, then perform attacks on 500 randomly picked examples from MR on three models. Table 3 shows the comparisons of different search methods. Genetic and PSO are excluded for their low efficiency (especially on DistilBERT). PARSE ($w = 8$) always achieves comparable attack success rates to complex methods while generally needing fewer queries and time, indicating that PARSE can be effectively applied to different search space.

Quality of Crafted Example. We measure the quality of the adversarial examples crafted by different search methods by attacking LSTM on MR in HowNet. We use the LanguageTool[1] to detect the grammar correctness and use the Universal Sentence Encoder (USE) (Cer et al., 2018) to measure the semantic similarity of 500 randomly picked successfully attacked examples and the original examples. We also conduct human evaluations on Amazon Mechanical Turk[2] by asking the workers

[1] https://languagetool.org/
[2] https://www.mturk.com/

Model	Search Method	Yelp GloVe A.S%	Yelp GloVe #Queries	Yelp GloVe Time (s)	Yelp HowNet A.S%	Yelp HowNet #Queries	Yelp HowNet Time (s)	MR GloVe A.S%	MR GloVe #Queries	MR GloVe Time (s)	MR HowNet A.S%	MR HowNet #Queries	MR HowNet Time (s)
LSTM	Random	87.45	305	5196	88.81	209	2955	66.58	64	856	60.54	38	483
	TIWO	86.64	237	4118	86.13	159	2221	72.73	61	877	65.14	38	266
	WIR-Delete	94.38	283	1803	94.19	233	986	80.84	69	684	71.38	51	286
	WIR-A	94.31	254	1721	94.88	216	969	81.13	69	673	71.35	52	284
	WIR-UNK	94.82	268	1787	94.87	221	971	81.45	68	671	72.47	52	257
	PARSE($w=1$)	96.47	480	2955	95.79	390	1525	83.38	73	680	74.23	73	302
	PARSE($w=2$)	97.31	568	4351	97.94	496	2650	84.87	98	683	77.76	81	397
	PARSE($w=4$)	97.81	899	4614	98.12	782	3774	87.18	149	907	78.68	113	542
	PARSE($w=8$)	98.64	1839	7736	98.37	1189	4129	88.11	223	1341	80.38	159	688
	Greedy	98.92	5786	85 763	98.33	2953	28 029	88.85	489	5288	81.72	253	1427
	Beam($w=4$)	99.37	14 271	177 167	98.54	6516	62 391	90.18	735	7539	82.17	368	2647
	Beam($w=8$)	99.54	25 311	458 740	98.97	11 979	100 524	90.93	1088	9399	83.71	566	5473
	Genetic	99.28	10 197	142 847	98.73	7991	97 482	93.74	3144	23 660	88.41	2579	11 232
	PSO	$\approx$ 224 / 1 week			99.17	38 749	317 175	89.90	3101	14 780	85.27	2458	6272
TextCNN	Random	79.12	399	8061	83.02	252	3503	62.98	65	1250	57.04	46	349
	TIWO	79.23	303	5160	82.31	191	1802	56.28	68	1149	54.19	48	530
	WIR-Delete	93.89	286	1957	91.64	234	1289	79.48	75	936	68.54	55	400
	WIR-A	94.68	267	1871	91.82	225	1347	78.32	73	925	69.71	53	424
	WIR-UNK	94.05	269	1845	91.79	221	1325	79.93	79	941	69.25	53	401
	PARSE($w=1$)	95.73	482	3055	93.23	378	1568	81.14	103	991	70.78	81	574
	PARSE($w=2$)	96.56	595	4482	93.38	472	2031	85.12	118	1036	73.02	95	724
	PARSE($w=4$)	96.95	838	5285	94.14	620	4670	86.96	185	1321	75.29	137	1119
	PARSE($w=8$)	97.26	1629	8619	96.80	1065	5665	87.93	278	2579	76.31	193	1342
	Greedy	96.54	6264	89 001	96.38	3327	27 494	88.61	532	5591	77.67	287	1935
	Beam($w=4$)	97.03	18 165	258 647	96.99	10 159	85 087	91.22	903	11 022	80.59	452	2730
	Beam($w=8$)	$\approx$ 287 / 1 week			97.84	18 327	211 399	93.47	1581	14 857	81.34	807	5918
	Genetic	96.98	13 678	168 335	97.13	13 469	136 602	95.34	3387	33 867	88.37	2918	14 455
	PSO	$\approx$ 168 / 1 week			98.51	51 341	402 452	92.25	4397	37 755	83.85	3081	7452

Table 2: The comparisons on attack success rate (A.S%), average #queries to attack one example (#Queries), and total seconds to attack 500 examples (Time) of different search methods. $\approx$ *n / 1 week* means the attack fails to complete in 1 week, and n is the number of the completed attacks.

	LSTM A.S%	LSTM #Que.	LSTM Time	TextCNN A.S%	TextCNN #Que.	TextCNN Time	DistilBERT A.S%	DistilBERT #Que.	DistilBERT Time
TextBugger (WIR-Delete)	78.44	48	29	78.86	48	28	69.84	51	107
w/ PARSE($w=1$)	79.73	65	33	81.34	65	29	75.24	66	112
w/ PARSE($w=2$)	83.81	79	39	84.99	78	38	78.18	95	157
w/ PARSE($w=4$)	84.65	97	56	86.88	113	53	80.74	132	213
w/ PARSE($w=8$)	87.53	138	78	87.23	168	81	81.05	218	335
w/ Greedy	86.73	227	89	86.35	243	91	83.74	312	436
w/ Beam($w=4$)	89.69	509	167	91.42	554	180	85.51	616	901
w/ Beam($w=8$)	90.89	819	273	91.74	855	319	87.23	1185	1637
BAE (WIR-Delete)	72.13	56	627	67.57	58	799	63.80	58	830
w/ PARSE($w=1$)	73.15	72	821	71.89	73	903	65.95	77	1011
w/ PARSE($w=2$)	74.43	89	970	72.87	92	1157	68.56	95	1248
w/ PARSE($w=4$)	74.63	108	1739	74.16	107	1971	70.34	125	2072
w/ PARSE($w=8$)	76.06	162	2758	74.88	169	2992	71.21	198	3228
w/ Greedy	77.39	229	3221	75.06	224	3719	71.33	234	4041
w/ Beam($w=4$)	79.98	384	5976	76.64	389	6731	75.50	395	7072
w/ Beam($w=8$)	80.65	494	10 679	78.92	568	12 493	76.32	642	12 929
DeepWordBug (WIR-Delete)	83.21	30	18	86.91	33	11	77.32	36	86
w/ PARSE($w=1$)	84.31	50	20	88.38	51	12	78.71	51	98
w/ PARSE($w=2$)	84.57	62	23	88.98	57	16	80.18	63	115
w/ PARSE($w=4$)	87.21	80	26	91.06	81	21	81.88	94	162
w/ PARSE($w=8$)	89.30	96	31	93.56	98	26	83.76	115	187
w/ Greedy	89.12	115	32	93.87	124	31	85.93	142	203
w/ Beam($w=4$)	92.20	254	58	95.56	249	48	91.32	179	419
w/ Beam($w=8$)	93.22	431	87	96.07	416	79	92.31	518	771

Table 3: Performance of different search methods when searching on the search space of previous frameworks. The original attacks use WIR-Delete as search method.

to give scores from 1 (best) to 5 (worse) to indicate the *Plausibility* of 100 adversarial examples and 100 randomly picked normal examples. Table 4 shows the results on the quality of the generated adversarial examples. PARSE effectively reduces the increased grammar errors by increasing the search

Search method	#Increased grammar errors	USE similarity	Perturbed%	Plausibility
Normal	-	-	-	3.15
Random	0.094	0.881	11.69	3.68
TIWO	0.063	0.893	10.72	3.51
WIR-Delete	0.101	0.894	10.05	3.46
WIR-A	0.104	0.896	10.01	3.45
WIR-UNK	0.097	0.897	9.95	3.45
PARSE($w=1$)	0.093	0.897	9.52	3.43
PARSE($w=2$)	0.091	0.897	9.59	3.42
PARSE($w=4$)	0.083	0.895	9.69	3.42
PARSE($w=8$)	0.077	0.895	9.76	3.39
Greedy	0.064	0.905	8.79	3.35
Beam($w=4$)	0.079	0.907	8.68	3.36
Beam($w=8$)	0.082	0.909	8.53	3.36
Genetic	0.074	0.880	11.21	3.37
PSO	0.089	0.896	10.18	3.35

Table 4: Quality of the adversarial examples crafted by different search methods.

width w, while a larger w will increase the grammar errors of the example crafted by beam search. The effect of PARSE on maintaining the USE similarity of sentences is similar to WIR and PSO methods and is just relatively 1.32% worse than greedy search and beam search. PARSE perturbs fewer words than WIR methods and only needs an average of 1.02% more perturbed words than greedy

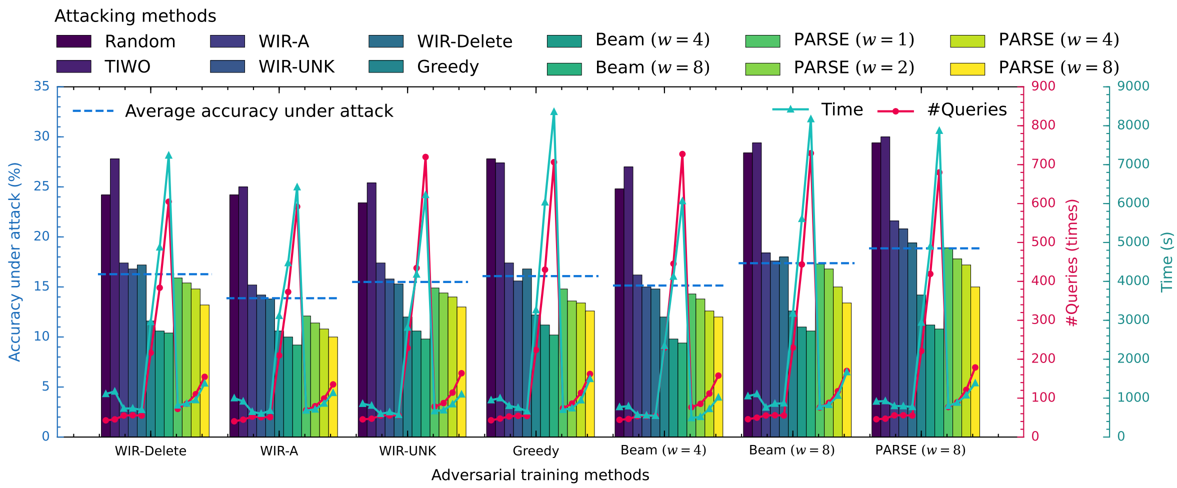


Figure 3: The comparisons of the #queries and time needed by different search methods to attack the adversarially trained models. Line plot correspond to the axis of the same color. Bar plot indicates the accuracy under attack.

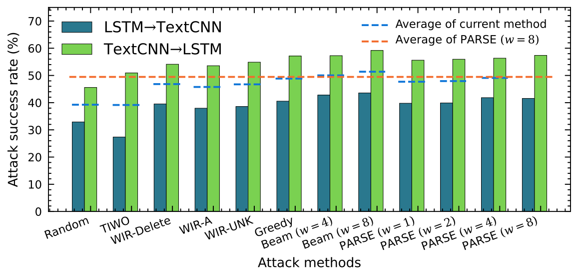


Figure 4: The comparison of the transferability of adversarial examples crafted by different search methods.

	A.S%	#Increased grammar errors	USE similarity	Perturbed%
PARSE ($w = 1$)	74.23	0.093	0.897	9.52
w/o Word Stability	72.47	0.097	0.897	9.95
PARSE ($w = 2$)	77.76	0.091	0.897	9.59
w/o Word Stability	74.45	0.095	0.894	10.05
PARSE ($w = 4$)	78.68	0.083	0.895	9.69
w/o Word Stability	76.74	0.096	0.893	10.12
PARSE ($w = 8$)	80.38	0.077	0.895	9.76
w/o Word Stability	77.23	0.101	0.892	10.45

Table 5: Influence of parameter w and word stability. *w/o word stability* means the search method does not make a distinction between the words of different stability, and *PARSE* $(w = 1)$ *w/o Word Stability* equals to *WIR-UNK* method.

search and beam search. The results on plausibility show that PARSE with larger w generates more human-understandable adversarial examples. The case study is shown in Table 6-12 in Appendix.

Adversarial Training and Model Robustness. We randomly generate 1000 adversarial examples by attacking LSTM on MR in HowNet with different search methods, and then adversarially retrains the LSTM with the generated adversarial examples. Figure 3 shows the results on model robustness and attack performance. The model trained with the adversarial examples crafted by PARSE $(w = 8)$ has the highest average accuracy, indicating that PARSE $(w = 8)$ outperforms other methods in improving model robustness. PARSE is effective and efficient even when attacking robust models.

Transferability. We randomly generate 1000 adversarial examples with different search methods on MR in HowNet. Figure 4 shows the result of transferability between LSTM and TextCNN. Increasing the beam width w in PARSE helps generate adversarial examples with higher transferability. When $w = 8$, the transferability of adversarial examples crafted by PARSE outperforms all baselines except beam search.

Ablation Study. Table 5 shows the influence of word stability when attacking LSTM on 500 examples from MR in HowNet. We find that changing the search space according to the word stability increases the attack success rate and helps generate adversarial examples with fewer grammar errors, higher USE similarity, and fewer perturbed words.

5 Conclusion

This paper proposes PARSE, an efficient search method for black-box adversarial text attacks, which performs search under dynamic search space following the subarea importance. PARSE can achieve comparable attack success rates to complex search methods while saving numerous queries and time. The adversarial examples crafted by PARSE are high quality and highly transferable. We hope the analysis in our paper will inspire future work.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. This research was supported by National Research and Development Program of China (No.2019YFB1005200).

References

Alice F. Healy Adam Drewnowski. 1978. Detection errors on the word the: Evidence for the acquisition of reading levels. *Memory & Cognition*, 5:403–409.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896. Association for Computational Linguistics.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Zhendong Dong and Qiang Dong. 2003. Hownet-a hybrid language and knowledge resource. In *International Conference on Natural Language Processing and Knowledge Engineering*, pages 820–824. IEEE.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36. Association for Computational Linguistics.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728. Association for Computational Linguistics.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Siddhant Garg and Goutham Ramakrishnan. 2020a. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181. Association for Computational Linguistics.

Siddhant Garg and Goutham Ramakrishnan. 2020b. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181. Association for Computational Linguistics.

Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. 2018. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031. Association for Computational Linguistics.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8018–8025. AAAI Press.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2021. Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 Conference of* *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5053–5069. Association for Computational Linguistics.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202. Association for Computational Linguistics.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4208–4215. ijcai.org.

Leo X McCusker, Philip B Gough, and Randolph G Bias. 1981. Word recognition inside out and outside in. *Journal of Experimental Psychology: Human Perception and Performance*, 7(3):538.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2574–2582. IEEE Computer Society.

John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020a. Reevaluating adversarial examples in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3829–3839. Association for Computational Linguistics.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020b. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124. Association for Computational Linguistics.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591. Association for Computational Linguistics.

Keith Rayner, Sarah J. White, Rebecca L. Johnson, and Simon P. Liversedge. 2006. Raeding wrods with jubmled lettres: There is a cost. *Psychological Science*, 17(3):192–193. PMID: 16507057.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

G. C Van Orden. 1987. A ROWS is a ROSE: Spelling, sound, and reading. *Memory & Cognition*, 15(3):181–198.

Jin Yong Yoo, John Morris, Eli Lifland, and Yanjun Qi. 2020. Searching for a search method: Benchmarking search algorithms for generating NLP adversarial examples. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 323–332. Association for Computational Linguistics.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

A Appendix

Additional Details on Model

The TextCNN has a 300-dimensional GloVe embedding layer (Pennington et al., 2014), a convolutional layer containing 150 filters with windows sizes $(3, 4, 5)$. The LSTM also has a 300-dimensional GloVe embedding layer and a bi-directional LSTM layer composed of 150 units. We use the *distilbert-base-uncased* as DistilBERT (Sanh et al., 2019), which is a fast Transformer model with 40% fewer parameters than BERT. The model used in Table 2 and Table 3 have the accuracy on clean dataset as follow: (LSTM, Yelp: 92.1%; LSTM, MR: 80.3%; TextCNN, Yelp: 91.4%; TextCNN, MR: 79.2%; DistilBERT, MR: 83.9%). Under our setting, attacking a base uncased version of BERT takes approximately 4 times as long as attacking a base uncased version of DistilBERT, and the beam search ($w = 4$), beam search ($w = 8$), Genetic algorithm, and PSO fails to attack 500 examples within one week when attacking the examples from Yelp on GloVe search space.

Additional Case Study

We give the case study of the adversarial examples crafted with PARSE in Table 6-12. The ~~green~~ word is the original word, and the following red word is the substitution.

Method	Perturbed Texts
PARSE($w = 1$)	There is a general ~~air~~ vent of ~~exuberance~~ ardour in all about the benjamins that's hard to resist.
PARSE($w = 2$)	There is a general ~~air~~ notification of ~~exuberance~~ fervor in all about the benjamins that's hard to resist.
PARSE($w = 4$)	There is a general ~~air~~ propaganda of ~~exuberance~~ eagerness in all about the benjamins that's hard to resist.
PARSE($w = 8$)	There is a general ~~air~~ propaganda of ~~exuberance~~ eagerness in all about the benjamins that's hard to resist.

Table 6: The case study of the adversarial examples crafted by attacking LSTM on the MR dataset in HowNet search space with PARSE.

Method	Perturbed Texts
PARSE($w = 1$)	Kids should have a stirring time at this ~~beautifully~~ prettily drawn ~~movie~~ cartoon. And adults will at least have a dream image of the west to ~~savor~~ taste whenever the film's lamer instincts are in the saddle.
PARSE($w = 2$)	Kids should have a stirring time at this ~~beautifully~~ prettily drawn ~~movie~~ cartoon. And adults will at least have a dream image of the west to ~~savor~~ taste whenever the film's lamer instincts are in the saddle.
PARSE($w = 4$)	Kids should have a stirring time at this ~~beautifully~~ pretty drawn ~~movie~~ cartoon. And adults will at least have a dream image of the west to ~~savor~~ taste whenever the film's lamer instincts are in the saddle.
PARSE($w = 8$)	Kids should have a stirring time at this ~~beautifully~~ wonderfully drawn ~~movie~~ cartoon. And adults will at least have a dream image of the west to ~~savor~~ taste whenever the film's lamer instincts are in the saddle.

Table 7: The case study of the adversarial examples crafted by attacking LSTM on the MR dataset in HowNet search space with PARSE.

Method	Perturbed Texts
PARSE($w = 1$)	It's dark but has ~~wonderfully~~ bizarrely ~~funny~~ recreational moments; you care about the characters; and the action and special effects are first-rate.
PARSE($w = 2$)	It's dark but has ~~wonderfully~~ bizarrely ~~funny~~ recreational moments; you care about the characters; and the action and special effects are first-rate.
PARSE($w = 4$)	It's dark but has ~~wonderfully~~ bizarrely ~~funny~~ recreational moments; you care about the characters; and the action and special effects are first-rate.
PARSE($w = 8$)	It's dark but has ~~wonderfully~~ suspiciously ~~funny~~ recreational moments; you care about the characters; and the action and special effects are first-rate.

Table 8: The case study of the adversarial examples crafted by attacking LSTM on the MR dataset in HowNet search space with PARSE.

Method	Perturbed Texts
PARSE($w = 1$)	In the director's cut, the film is not only a love song to the movies but it also is more fully an example of the kind of lush, all-enveloping movie ~~experience~~ aftertaste it ~~rhapsodizes~~ talks.
PARSE($w = 2$)	In the director's cut, the film is not only a love song to the movies but it also is more fully an example of the kind of lush, all-enveloping movie ~~experience~~ aftertaste it ~~rhapsodizes~~ talks.
PARSE($w = 4$)	In the director's cut, the film is not only a love song to the movies but it also is more fully an example of the kind of lush, all-enveloping movie ~~experience~~ aftertaste it ~~rhapsodizes~~ lectures.
PARSE($w = 8$)	In the director's cut, the film is not only a love song to the movies but it also is more fully an example of the kind of lush, all-enveloping movie ~~experience~~ aftertaste it ~~rhapsodizes~~ lectures.

Table 9: The case study of the adversarial examples crafted by attacking LSTM on the MR dataset in HowNet search space with PARSE.

Method	Perturbed Texts
PARSE($w = 1$)	A ~~smart~~ brainy and ~~funny~~ ridiculous, albeit sometimes superficial, cautionary ~~tale~~ narration of a ~~technology~~ tech in search of an artist.
PARSE($w = 2$)	A ~~smart~~ brainy and ~~funny~~ ridiculous, albeit sometimes superficial, cautionary ~~tale~~ narration of a ~~technology~~ tech in search of an artist.
PARSE($w = 4$)	A ~~smart~~ brainy and ~~funny~~ ridiculous, albeit sometimes superficial, cautionary ~~tale~~ narration of a ~~technology~~ tech in search of an artist.
PARSE($w = 8$)	A ~~smart~~ brainy and ~~funny~~ laughable, albeit sometimes superficial, cautionary ~~tale~~ story of a ~~technology~~ tech in search of an artist.

Table 10: The case study of the adversarial examples crafted by attacking LSTM on the MR dataset in HowNet search space with PARSE.

Method	Perturbed Texts
PARSE($w = 1$)	The ~~wonderfully~~ curiously ~~lush~~ drunk morvern callar is pure punk existentialism, and ms. ramsay and her co-writer, liana dognini, have dramatized the alan warner novel, which itself felt like an answer to irvine welsh's book trainspotting.
PARSE($w = 2$)	The ~~wonderfully~~ curiously ~~lush~~ drunk morvern callar is pure punk existentialism, and ms. ramsay and her co-writer, liana dognini, have dramatized the alan warner novel, which itself felt like an answer to irvine welsh's book trainspotting.
PARSE($w = 4$)	The ~~wonderfully~~ curiously ~~lush~~ drunk morvern callar is pure punk existentialism, and ms. ramsay and her co-writer, liana dognini, have dramatized the alan warner novel, which itself felt like an answer to irvine welsh's book trainspotting.
PARSE($w = 8$)	The ~~wonderfully~~ singularly ~~lush~~ drunk morvern callar is pure punk existentialism, and ms. ramsay and her co-writer, liana dognini, have dramatized the alan warner novel, which itself felt like an answer to irvine welsh's book trainspotting.

Table 11: The case study of the adversarial examples crafted by attacking LSTM on the MR dataset in HowNet search space with PARSE.

Method	Perturbed Texts
PARSE($w = 1$)	The film is hard to dismiss – moody, thoughtful, and ~~lit~~ fainted by ~~flashes~~ blazes of mordant ~~humor~~ animation.
PARSE($w = 2$)	The film is hard to dismiss – moody, thoughtful, and ~~lit~~ fainted by ~~flashes~~ blazes of mordant ~~humor~~ animation.
PARSE($w = 4$)	The film is hard to dismiss – ~~moody~~ listless, thoughtful, and lit by ~~flashes~~ winks of mordant ~~humor~~ animation.
PARSE($w = 8$)	The film is hard to dismiss – ~~moody~~ listless, thoughtful, and lit by ~~flashes~~ blazes of mordant ~~humor~~ vividness.

Table 12: The case study of the adversarial examples crafted by attacking LSTM on the MR dataset in HowNet search space with PARSE.