Generating Textual Adversaries with Minimal Perturbation

Xingyi Zhao
Utah State University
xingyi.zhao@usu.edu

Lu Zhang
University of Arkansas
lz006@uark.edu

Depeng Xu
University of North Carolina at Charlotte
depeng.xu@uncc.edu

Shuhan Yuan
Utah State University
shuhan.yuan@usu.edu

Abstract

Many word-level adversarial attack approaches for textual data have been proposed in recent studies. However, due to the massive search space consisting of combinations of candidate words, the existing approaches face the problem of preserving the semantics of texts when crafting adversarial counterparts. In this paper, we develop a novel attack strategy to find adversarial texts with high similarity to the original texts while introducing minimal perturbation. The rationale is that we expect the adversarial texts with small perturbation can better preserve the semantic meaning of original texts. Experiments show that, compared with state-of-the-art attack approaches, our approach achieves higher success rates and lower perturbation rates in four benchmark datasets[1].

1 Introduction

Recent studies have demonstrated that deep learning based natural language processing (NLP) models are vulnerable to adversarial attacks(Li et al., 2018; Gao et al., 2018; Yang et al., 2020). Various textual adversarial attack strategies have been proposed including character-level, word-level, and sentence-level attacks (Zhang et al., 2020; Morris et al., 2020). In this paper, we focus on word-level attacks on text classifiers, especially the word substitution-based attacks, which are shown to be both effective and efficient (Alzantot et al., 2018; Ren et al., 2019; Zang et al., 2019; Jin et al., 2020; Li et al., 2020b,a).

A successful adversarial attack should satisfy three requirements: 1) can fool the neural network models to make the wrong prediction; 2) the modification is slight and imperceptible to human judgments; and 3) can find the adversarial example in a reasonable run time. However, existing approaches suffer from various limitations. For example, the

Original	A good film with a solid pedigree both in front of and, more specifically, behind the camera.
PWWS (11.76%)	A **commodity** film with a solid pedigree both in front of and, more specifically, **bum** the camera.
TextFooler (29.4%)	A **better** film with a **stable** pedigree both in **newsweek** of and, more **concretely**, behind the **camcorder**.
Bert-Attack (29.4%)	A **fair positive** with a solid pedigree both in **after** of and, more **roughly**, behind the **canvas**.
TAMPERS (5.88%)	A **harmless** film with a solid pedigree both in front of and, more specifically, behind the camera.

Table 1: Adversarial examples generated by different attack approaches on MR. The bold red font indicates the perturbed words. The number in the bracket indicates the perturbation rate for each approach

greedy algorithm-based attack approaches (Ren et al., 2019; Jin et al., 2020) usually find a word in each step that can maximize the change of neural network outputs and do not explicitly consider the requirement of semantic preservation. On the other hand, the combinatorial optimization-based approaches, such as genetic algorithm (Alzantot et al., 2018; Zang et al., 2019), define fitness functions to control the semantic quality of adversarial texts, but due to the huge search space, they usually need a large amount of time to find a solution, especially for a long text.

In this work, we propose **T**extual **A**dversarial attack with **M**inimal **PE**rturbation in a **R**educed Search **S**pace (TAMPERS) to craft high quality adversarial texts.

The major motivation is that, by substituting a vulnerable word with its synonyms or sememes, we expect that the adversarial text with small perturbation can have large semantic preservation. We model the word-level attacks as a combinatorial optimization problem (Zang et al., 2019). We then develop a two-step heuristic for solving this problem. In the first step *search space reduction*, we first build candidate lists for all the content words by combining synonyms from WordNet (Fellbaum, 2010) and sememes from HowNet (Dong and Dong, 2006) similar to other word level attack strategies (Ren et al., 2019; Zang et al.,

[1]The code is available at https://github.com/xingyizhao/TAMPERS

2019). To avoid searching in a huge search space, instead of searching through all content words to generate adversarial texts, we reduce the search space by finding a small set of vulnerable words that are sufficient to fool the neural network using a greedy algorithm. Then, in the second step *iterative search*, we further minimize the perturbation by iteratively restoring the substituted words back to the original words while keeping the classifier making the wrong prediction. Each time we restore a word, we adopt the genetic algorithm (GA) to search adversarial texts over the remaining substituted words. In this way, we expect to find an adversarial example with the least perturbation on the original text.

We have conducted experiments on four benchmark datasets. The results show that our approach can successfully fool the fine-tuned BERT model with high success attack rates and low perturbation rates. Table 1 shows an example of adversarial texts generated by different attack approaches. We can notice that TAMPERS has a much lower perturbation rate compared with PWWS, TextFooler, and BERT-Attack.

2 Methodology

Given any text $X = \{w_1, \dots, w_n, \dots, w_N\}$ with N words and the true label Y, consider a pre-trained classifier that maps X to Y, i.e., $F : X \to Y$. We assume the soft-label black-box setting where adversaries can query the classifier for classification probabilities on the given sample, but have no access to the model structures, parameters as well as training data. Our goal is to craft an adversarial sample X^* such that $F(X^*) \neq Y$ with minimal word substitutions.

We propose a novel framework called **T**extual **A**dversarial attack with **M**inimal **PE**rturbation in a **R**educed search **S**pace (TAMPERS), which consists of two steps. The search space reduction step is to find vulnerable words with the goal of reducing the search space, while the iterative search step is to further minimize the perturbation so that the semantics of the original text can be well-preserved. The overview of TAMPERS is shown in Figure 1. Below we describe each step in detail.

2.1 Search Space Reduction

In order to formalize the search space to craft the adversarial texts, the first step is to find the vulnerable words in the text $X = \{w_1, \dots, w_n, \dots, w_N\}$.

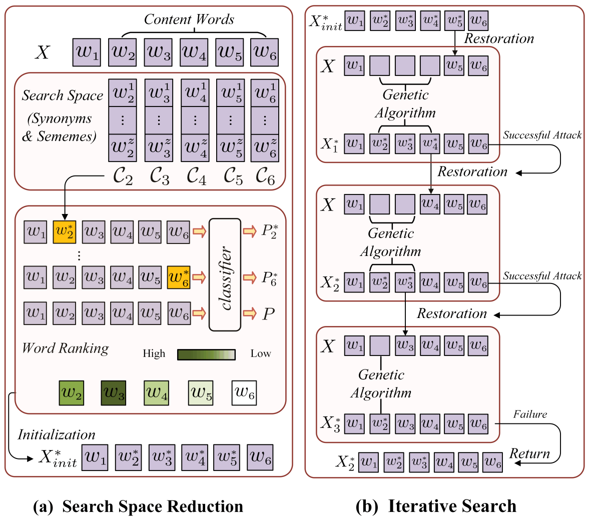

Figure 1: Overview of TAMPERS

Some related studies (Jin et al., 2020; Li et al., 2020b) define the importance score of a content word w_n based on the probability change of the classifier with and without the word w_n (word deletion). Differently, we derive the importance score of the word w_n by replacing the word with another word in a candidate list that leads to the maximum probability change of the classifier (word derivative). The motivation of our approach is to approximate the computation of the gradient of the classifier output on the search space.

Specifically, given a content word w_n, we first find its synonyms from WordNet and sememes from HowNet with the same part of speech. Then, the substitution candidate list $\mathcal{C}_n$ for the word w_n consists of the top-z synonyms and sememes that are similar to w_n based on the GloVe embeddings (Pennington et al., 2014). After getting the candidate list $\mathcal{C}_n$, we use each word $w'_n \in \mathcal{C}_n$ to substitute w_n to get a perturbed text X'_n. The set of perturbed texts formed by all words in the candidate list is denoted by $\mathcal{X}'_n$. Then, we compute the probability change of the classifier and define the importance score $S(w_n)$ as the maximum probability change across $\mathcal{X}'_n$:

$$S(w_n) = \max_{X'_n \in \mathcal{X}'_n} \{P(Y_{true}|X) - P(Y_{true}|X'_n)\}.$$

We denote the candidate word that achieves the score $S(w_n)$ as w_n^*.

Next, we rank all content words according to the importance score in descending order and adopt a greedy algorithm to generate the adversarial text, denoted as X^*_{init}. The greedy algorithm substitutes the content words one by one according to a descending order of their importance scores until the perturbed text can fool the classifier. In this step, we always substitute each word w_n with the candidate word w_n^* that is computed above. After this step, we get a small set of vulnerable words $\mathcal{L} = \{w_{(1)}^*, \ldots, w_{(k)}^*, \ldots, w_{(K)}^*\}$ as the initial substitution words that is sufficient to fool the classifier, where $w_{(k)}^*$ is the (k)-th vulnerable word by importance score. The pseudo-code of the search space reduction is shown in Algorithm 1 in the appendix.

2.2 Iterative Search

Step 1 could significantly reduce the search space and produce an initial solution X_{init}^* that already has a small perturbation. However, since the first step is in principle a one-step greedy strategy, we would like to further minimize the perturbation in order to better preserve the semantic integrity. To this end, we develop a novel iterative search approach, where we iteratively restore the substituted words back to the original words according to the importance scores while keeping the classifier getting fooled. Each time after a word is restored, we reduce the list $\mathcal{L}$ accordingly and adopt GA to search for new substitutions for the perturbed words in the reduced $\mathcal{L}$. Since the search space of the GA is limited to $\mathcal{L}$, TAMPERS is much more efficient than previous GA-based approaches (e.g., Alzantot et al., 2018).

Symbolically, we start from the least vulnerable word $w_{(K)}^*$ in $\mathcal{L}$ and restore it to the original word $w_{(K)}$. Then, we run the GA over $\mathcal{L} \setminus w_{(K)}^*$ in order to find an adversarial text that can still fool the classifier. If we successfully find a solution, we then continue to restore the second least vulnerable word $w_{(K-1)}^*$ to $w_{(K-1)}$, reduce the list $\mathcal{L}$, and run the GA to find the adversarial text. We repeat this process until the GA cannot find a solution in the reduced search space. Finally, the algorithm returns the last successful solution as the output, which is the adversarial text with the minimal perturbation among those that have ever been found by the algorithm. The pseudo-code of the algorithm is shown in Algorithm 2 in the appendix.

The details of the genetic algorithm including initialization, selection, crossover, and mutation are described as follows

Step 1. Initialization. This step is to generate the first generation of genetic algorithm $\mathcal{G}^0$ with M samples (population size). Given the current set of vulnerable words $\mathcal{L}$, we randomly choose a candidate from $\mathcal{C}$ for each vulnerable word in $\mathcal{L}$ and generate M texts, denoted as $\mathcal{G}^0 = \{X_1^0, \ldots, X_m^0, \ldots, X_M^0\}$.

Step 2. Selection. In the g-th generation $\mathcal{G}^g$, we first sort the texts based on the fitness score $h(X_m^g)$, defined as

$$h(X_m^g) = \begin{cases} P(Y_{true}|X) - P(Y_{true}|X_m^g), & \text{if } F(X_m^g) = Y_{true} \\ 1, & \text{if } F(X_m^g) \neq Y_{true} \end{cases}$$

Then, we select texts with the top 20% highest fitness score in the current generation as elitism and directly send to the next generation $\mathcal{G}^{g+1}$.

Step 3. Crossover. For the texts in the g-th generation $\mathcal{G}^g$ with the top 50% highest fitness score, we first randomly select two texts, denoted as X_p^g and X_q^g. Then, for the positions of vulnerable words in the text, we randomly sample a word from either X_p^g or X_q^g at each position to generate a new text. We repeat the above two steps to generate the rest 80% of texts in the next generation $\mathcal{G}^{g+1}$.

Step 4. Mutation. During the process of crossover, there is a small probability that we randomly select a new word from the candidate list instead of the word from X_p^g or X_q^g.

After initializing the first generation $\mathcal{G}_0$ in Step 1 and repeating Steps 2 to 4 multiple iterations, if we are able to find an adversarial text giving current vulnerable words $\mathcal{L}$, it means compared with the adversarial text X_{init}^* with a perturbation rate K/N, we generate new adversarial text with a perturbation rate $(K-1)/N$.

3 Experiments

3.1 Experimental Setup

Datasets and Victim Model. We evaluate TAMPERS on four text classification datasets, including **IMDB** (Maas et al., 2011), **Yelp** (Zhang et al., 2015), **MR** (Pang and Lee, 2005), and **SST2** (Socher et al., 2013). Following Jin et al., 2020 and Li et al., 2020b, we report the average outcome of 5 runs where in each run we randomly sample 1000 texts from each dataset. The victim model is the fine-tuned BERT$_{\text{base}}$.

Baselines. We choose three word-level adversarial attack models as baselines: 1) **TextFooler** adopts counter-fitting word embeddings and the greedy search to generate adversarial texts (Jin et al., 2020); 2) **PWWS** is a synonym-based attack method using the greedy algorithm (Ren et al., 2019); 3) **BERT-Attack** uses a mask language model to predict candidate words and applies the

Dataset	Model	Original Acc.	Attacked Acc.	Success Rate	Perturb Rate	Semantic Similarity
IMDB (225)	TextFooler	94.7	0.8	99.1	8.7	91.4
	PWWS		1.7	98.2	5.1	87.9
	Bert-Attack		3.3	96.5	6.6	91.7
	TAMPERS		**0.3**	**99.6**	**2.3**	**93.5**
Yelp (129)	TextFooler	97.3	5.8	94.0	10.6	88.5
	PWWS		5.5	94.3	7.5	87.9
	Bert-Attack		5.9	93.9	6.8	**89.7**
	TAMPERS		**4.1**	**95.9**	**4.3**	89.1
MR (19.16)	TextFooler	96.6	7.0	92.8	18.7	83.6
	PWWS		14.7	84.8	14.5	84.9
	Bert-Attack		11.3	88.3	14.8	84.1
	TAMPERS		**6.4**	**93.4**	**9.6**	**86.3**
SST-2 (16.78)	TextFooler	93.8	**5.2**	**94.5**	17.2	83.9
	PWWS		13.6	85.6	15.2	84.0
	Bert-Attack		7.0	92.5	14.0	**85.0**
	TAMPERS		6.3	93.5	**10.1**	84.9

Table 2: Results (shown in percentage) of attacking against fine-tuned BERT model on various datasets. We report the average outcome of 5 runs where in each run we randomly sample 1000 texts from each dataset. The number in the bracket indicates the average text length in each dataset.

Dataset	Model	Runtime (s/sample)
IMDB	Genetic Attack	2477
	PSO	1495
	TAMPERS	98.4

Table 3: Runtime Comparison

greedy algorithm to craft adversarial texts (Li et al., 2020b).

Evaluation Metrics. Following TextDefender (Li et al., 2021), we adopt various evaluation metrics to evaluate the quality of adversarial texts, including *original accuracy*, *attacked accuracy*, *attack success rate*, *average perturbation rate*, and *semantic similarity*. Following TextFooler and BERT-Attack, we adopt Universal Sentence Encoder (Cer et al., 2018) to evaluate the semantic similarity between the original text and its adversarial counterpart.

Implementation Details. We constrain the size of candidates set $\mathcal{C}_n$ as 50 for each content word. For the initialized adversarial texts X^*_{init} with more than one perturbed word, we apply iterative search to find better adversarial examples with fewer perturbation. In GA, we set the population size $M = 10$ and maximal generation $T = 100$.

3.2 Experimental Results

Attack Results. As shown in Table 2, TAMPERS successfully attacks the fine-tune BERT$_{\text{base}}$ model on four datasets. In general, TAMPERS has the smallest perturbation rate and highest success attack rate among all baselines. As expected, the performance of TAMPERS is better on long texts. For the IMDB and Yelp datasets that mainly contain long texts, TAMPERS significantly reduces the perturbation rates compared with baselines and still achieves the highest attack success rate. On the MR and SST-2 datasets that mainly contain short texts, the improvement of TAMPERS is less significant but it can still reduce the perturbation rates by substituting around 2 words while maintaining high attack success rates.

Runtime Comparison. We compare the runtime of TAMPERS with that of Genetic Attack (Alzantot et al., 2018) and Particle Swarm Optimization (PSO) (Zang et al., 2019), both of which are also combinatorial optimization-based approaches. The results on the IMDB dataset are shown in Table 3. We observe similar results on other datasets which are included in the appendix. As can be seen, TAMPERS is much faster and can craft an adversarial text in a reasonable time frame compared with Genetic Attack and PSO, thanks to the search space reduction step that significantly improves the efficiency of the algorithm.

4 Conclusions

In this work, we developed a novel word-level attack framework, called TAMPERS, that adopts a two-step approach with the goal of minimizing the perturbation. The first step reduces the search space and finds an initial adversarial text with the reduced perturbation. The second step then further minimizes the perturbation by iteratively restoring the substituted words back to the original words while keeping the classifier getting fooled. Experimental results show that TAMPERS generally achieves minimal perturbation in crafting adversarial examples while keeping a high success attack rate.

Limitations

One limitation of the proposed framework is that in the iterative search step, each time we restore a vulnerable word $w^*_{(k)}$ to the original word $w_{(k)}$, GA starts from the scratch to find an adversarial example, which requires a lot of queries to the victim models. In the future, we would like to study how to improve query efficiency while maintaining the high success rate and low perturbation rate.

Acknowledgement

This work was supported in part by NSF 2103829.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Zhendong Dong and Qiang Dong. 2006. *Hownet and the computation of meaning (with Cd-rom)*. World Scientific.

Christiane Fellbaum. 2010. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.

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.

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 *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020a. Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*.

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*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. 2021. Searching for an effective defender: Benchmarking defense against adversarial word substitution. *arXiv preprint arXiv:2108.12777*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*.

Jeffrey Pennington, Richard Socher, and Christopher D 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.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan. 2020. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *J. Mach. Learn. Res.*, 21(43):1–36.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2019. Word-level textual adversarial attacking as combinatorial optimization. *arXiv preprint arXiv:1910.12196*.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

A Appendix

A.1 Pseudo-code

The pseudo-code of Search Space Reduction and Iterative Search is given in Algorithms 1 and 2, respectively.

Algorithm 1: Search Space Reduction

```
Input  : Text $X$; Label $Y$; Classifier $F$
Output : Adversarial example $X^*_{init}$
1  for each content word $w_n$ in $X$ do
2      Get a substitution candidates set $\mathcal{C}_n$ for $w_n$;
3      Compute the importance score $S(w_n)$ and get
         the candidate word $w_n^*$;
4  end
5  Build a list $S$ of content words $w_n \in X$ in a
     descending order based on $S(w_n)$;
6  $\mathcal{L} = list()$;
7  $X^* \leftarrow \{w_1, ..., w_n, ...w_N\}$;
8  for each word $w_n$ in $S$ do
9      $\mathcal{L}.append(w_n)$;
10     $X^*.replace(w_n, w_n^*)$;
11     if $F(X^*) \neq Y$ then
12         $X^*_{init} \leftarrow X^*$;
13         return $X^*_{init}$ and $\mathcal{L}$;
14     end
15 end
16 return None;
```

Algorithm 2: Iterative Search

```
Input  : Text $X$; Label $Y$; Classifier $F$; Adversarial
         text $X^*_{init}$; Vulnerable word list $\mathcal{L}$ with size
         $K$; Substitution candidate sets $Q$ for $w \in \mathcal{L}$
Output : Adversarial example $X^*$
1  $X^* \leftarrow X^*_{init}$;
2  for $k = K$ to $1$ do
3      $\mathcal{L} = \mathcal{L} \setminus w_{(k)}$;
4      Initialize the first generation $\mathcal{G}^0$;
5      for $g = 1$ to $T$ do
6          Generate texts in $\mathcal{G}^g$ based on Selection,
             Crossover and Mutation;
7          $\mathcal{G}^g \leftarrow sort(\mathcal{G}^g)$ ;
8          if $F(\mathcal{G}^g[0]) \neq Y$ then
9              $X' \leftarrow \mathcal{G}^g[0]$;
10             break;
11         end
12     end
13     if $F(\mathcal{G}^g[0]) == Y$ then
14         break;
15     end
16     $X^* \leftarrow X'$;
17 end
18 return $X^*$;
```

A.2 Implementation Details

For TextFooler and PWWS, we follow the implementations in TextAttack (Morris et al., 2020), while for BERT-Attack, we follow the implementations in TextDefender (Li et al., 2021). We use the fine-tuned victim models for all the datasets shared by TextAttack [2]. All experiments are run on AMD Ryzen Threadripper 3960X 24-core Processor and NVIDIA GeForce RTX 3090.

A.3 Experimental Results

Due to the high time-consuming of Genetic Attack (Alzantot et al., 2018) and PSO (Zang et al., 2019) algorithms, especially on long texts, we compare Genetic Attack and PSO with TAMPERS on MR and SST-2 by randomly selecting 250 correctly predicted texts. Table 4 shows the experimental results. Overall, compared with Genetic Attack and PSO, TAMPERS can still achieve the lowest perturbation rate with a high success attack rate. Meanwhile, TAMPERS is much faster than Genetic Attack and PSO even in datasets with short texts, which shows the advantage of the search space reduction step proposed in our framework.

A.4 Case Study

We show the cases of adversarial examples generated by different approaches on IMDB and Yelp in Tables 5 and 6, respectively.

[2] https://huggingface.co/textattack

Dataset	Method	Original Acc.	Attacked Acc.	Success Rate	Perturb Rate	Semantic Similarity	Runtime (s/example)
MR (19.25)	Genetic Attack	97.2	30.8	68.3	14.5	**86.5**	133.9
	PSO		**6.0**	**93.8**	20.3	81.8	25.2
	TAMPERS		10.0	89.7	**8.3**	85.9	**10.6**
SST-2 (16.77)	Genetic Attack	92.4	35.6	61.5	15.6	**86.3**	43.2
	PSO		8.4	90.9	19.8	82.2	14.4
	TAMPERS		**6.8**	**92.7**	**10.4**	84.9	**9.8**

Table 4: Attacking results (shown in percentage) of Genetic Attack, PSO, and TAMPERS against fine-tuned BERT.

IMDB (Positive)	A wonderful little production. The filming technique is very unassuming very old time BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece. The actors are extremely well chosen Michael Sheen not only "has got all the polari" but he has all the voices down pat too! You can truly see the seamless editing guided by the references to Williams' diary entries, not only is it well worth the watching but it is a terrificly written and performed piece. A masterful production about one of the great master's of comedy and his life. The realism really comes home with the little things: the fantasy of the guard which, rather than use the traditional 'dream' techniques remains solid then disappears. It plays on our knowledge and our senses, particularly with the scenes concerning Orton and Halliwell and the sets (particularly of their flat with Halliwell's murals decorating every surface) are terribly well done.
PWWS (negative)	**amp** wonderful **petty** production. The filming technique is very unassuming very old time BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire **pick**. The actors are extremely well chosen Michael Sheen not only "has got all the polari" but he has all the voices down **slick** too! You can truly see the seamless editing **steer** by the references to Williams' diary entries, not only is it **substantially** worth the watching but it is a terrificly written and performed piece. A masterful production about **unitary** of the great master's of **clowning** and his **spirit**. The **pragmatism** really **amount** home with the little things: the **illusion** of the guard which, rather than use the traditional 'dream' techniques remains solid then disappears. It plays on our knowledge and our senses, particularly with the **conniption** concerning Orton and Halliwell and the sets (particularly of their flat with Halliwell's murals decorating every surface) are **atrociously fountainhead through**.
TextFooler (negative)	A **unbelievable** little production. The filming **tech** is **terribly** unassuming **incredibly** oldtimeBBC **attire** and **begs** a comforting, and **invariably** discomforting, **meanings** of **reality** to the **total lump**. The actors are **horribly** well taking Michael Sheen not only "has got all the polari" but he has all the voices down pat too! You can truly **suppose** the seamless editing **directorate** by the **referencing** to Williams' diary **inscriptions**, not only is it **sufficiently valuable** the watching but it is a terrificly **writing** and **achieve** piece. A masterful production about one of the **excellent** master's of **entertaining** and his **duration**. **Both** realism **surely arrives interiors** with the **scarcely stuff**: the **imagined** of the **guarding** which, **reasonably** than **employing** the **routine** 'dream' **technician** remains solid then disappears. **He gaming** on our **proficiency** and our **wits**, particularly with the **imagery implicated Smackdown** and Halliwell and the **provides** (**substantially** of their flat with Halliwell's **graphics** decorating every **cosmetic**) are **freakishly successfully effected**.
BertAttack (negative)	A wonderful little production. The filming **work** is very unassuming very oldtimeBBC **sort** and gives a comforting, and **perhaps** discomforting, sense of **horror** to the entire **exhibition**. The actors are **seriously holy appointed** Michael Sheen not only "has got all the polari" but he has all the voices down pat too! You can truly see the seamless editing guided by the references to Williams' **newspaper** entries, not only is it well **enjoyed** the watching but it is a terrificly written and performed piece. **good** masterful production about **being** of the great master's of **comedians** and his **personality**. The realism really **starts** home with the little things: the **projection** of the guard which, rather than use the traditional 'dream' techniques remains solid then **dismiss**. It plays on our knowledge and our senses, particularly with the **elements surrounding** Orton and Halliwell and the **production** (particularly of their flat with Halliwell's **pupils** decorating **totally** surface) are terribly **thoroughly awful**.
TAMPERS (negative)	A wonderful **less** production. The filming technique is very unassuming very oldtimeBBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece. The actors are extremely well chosen Michael Sheen not only "has got all the polari" but he has all the voices down pat too! You can truly see the seamless editing guided by the references to Williams' diary entries, not only is it well worth the watching but it is a terrificly written and performed scrap . A **tyrannical preparation** about one of the great master's of **drollery** and his life. The realism really comes home with the little things: the fantasy of the guard which, rather than use the traditional 'dream' techniques remains solid then disappears. It plays on our knowledge and our senses, particularly with the scenes concerning Orton and Halliwell and the sets (particularly of their flat with Halliwell's murals decorating every surface) are **atrociously so** done.

Table 5: Given an IMDB review, the adversarial examples generated by different approaches. The perturbed words are highlighted in red.

Yelp (Negative)	I don’t understand what all the hoopla is about. The food was lousy. The ribs tasted like the cow had mad cow disease. The sauce was weak. Overall, the food is overpriced. I’ve gone to this place twice and all I can say is I loved the beer!
PWWS (Positive)	**iodin** don’t **realise** what all the hoopla is about. The food was lousy. The ribs tasted like the cow had mad cow disease. The sauce was weak. Overall, the food is overpriced. I’ve **extend** to this place twice and all I can say is **iodin** loved the beer!
TextFooler (Positive)	I don’t understand what all the **tizzy** is about. The food was **sorrowful**. The ribs tasted like the cow had mad cow disease. The sauce was weak. **Comprehensive**, the food is overpriced. I’ve gone to this place twice and all I can say is I loved the **brews**!
BertAttack (Positive)	I don’t **forget** what all the hoopla is about. The **there** was lousy. The ribs tasted like the cow had mad cow disease. The sauce was weak. Overall, the food is overpriced. I’ve gone to this place twice and all I can say is **totally** loved the beer!
TAMPERS (Positive)	I don’t understand what all the hoopla is about. The **diet** was lousy. The ribs tasted like the cow had mad cow disease. The sauce was weak. Overall, the **snacks** is overpriced. I’ve **coming** to this place twice and all I can say is I loved the beer!

Table 6: Given a Yelp review, the adversarial examples generated by different approaches. The perturbed words are highlighted in red.