Modeling Adversarial Attack on Pre-trained Language Models as Sequential Decision Making

Xuanjie Fang[1]*, **Sijie Cheng**[1,2,3,4]*, **Yang Liu**[2,3,4,5], **Wei Wang**[1]†
[1]School of Computer Science, Fudan University, Shanghai, China
[2]Dept. of Comp. Sci. & Tech., Institute for AI, Tsinghua University, Beijing, China
[3]Institute for AI Industry Research (AIR), Tsinghua University, Beijing, China
[4]Beijing National Research Center for Information Science and Technology, Beijing, China
[5]Shanghai Artificial Intelligence Laboratory, Shanghai, China
{xjfang20,sjcheng20,weiwang1}@fudan.edu.cn, liuyang2011@tsinghua.edu.cn

Abstract

Pre-trained language models (PLMs) have been widely used to underpin various downstream tasks. However, the *adversarial attack* task has found that PLMs are vulnerable to small perturbations. Mainstream methods adopt a detached two-stage framework to attack without considering the subsequent influence of substitution at each step. In this paper, we formally model the adversarial attack task on PLMs as a *sequential decision-making* problem, where the whole attack process is sequential with two decision-making problems, i.e., word finder and word substitution. Considering the attack process can only receive the final state without any direct intermediate signals, we propose to use reinforcement learning to find an appropriate sequential attack path to generate adversaries, named SDM-ATTACK. Extensive experimental results show that SDM-ATTACK achieves the highest attack success rate with a comparable modification rate and semantic similarity to attack fine-tuned BERT. Furthermore, our analyses demonstrate the generalization and transferability of SDM-ATTACK. The code is available at https://github.com/fduxuan/SDM-Attack.

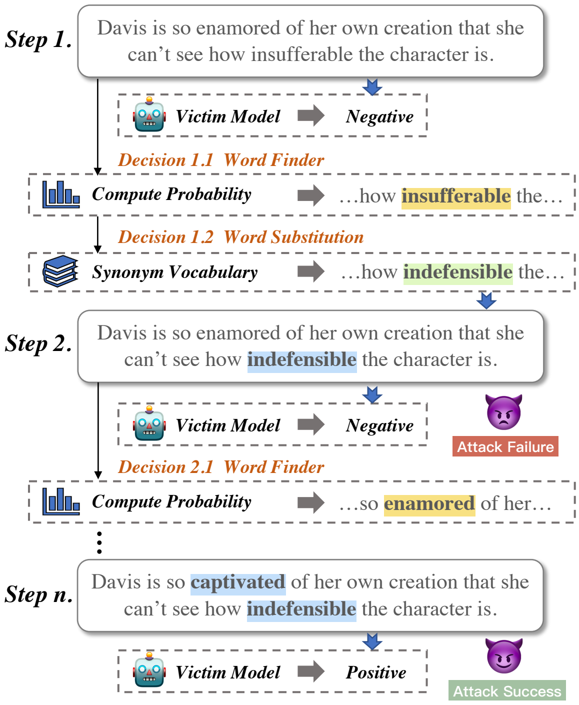
Step 1.
Davis is so enamored of her own creation that she can't see how insufferable the character is.
Victim Model
Negative
Decision 1.1 Word Finder
Compute Probability
...how insufferable the...
Decision 1.2 Word Substitution
Synonym Vocabulary
...how indefensible the...
Step 2.
Davis is so enamored of her own creation that she can't see how indefensible the character is.
Victim Model
Negative
Attack Failure
Decision 2.1 Word Finder
Compute Probability
...so enamored of her...
Step n.
Davis is so captivated of her own creation that she can't see how indefensible the character is.
Victim Model
Positive
Attack Success

Figure 1: Illustrative example of modeling the adversarial attack into sequential decision making. The entire attack process is a sequence with two decision-making problems, i.e., word finder and substitution, until the adversary against the victim model is successful.

1 Introduction

Nowadays, pre-trained language models (PLMs) have shown strong potential in various downstream tasks (Devlin et al., 2018; Brown et al., 2020). However, a series of studies about *adversarial attack* (Jin et al., 2020; Li et al., 2020a,b) have found that PLMs are vulnerable to some small perturbations based on the original inputs. The adversarial attack is essential to develop trustworthy and robust PLMs in Artificial Intelligence (AI) community (Thiebes et al., 2021; Marcus, 2020).

Despite the adversarial attack achieving success in both image and speech domains (Chakraborty et al., 2018; Kurakin et al., 2018; Carlini and Wagner, 2018), it is still far from perfect in the natural language processing (NLP) field due to the discrete nature of language (Studdert-Kennedy, 2005; Armstrong et al., 1995). The main problem is to find an appropriate search algorithm that can make perturbations to mislead the victim models (i.e., PLMs) successfully (Morris et al., 2020; Yoo and Qi, 2021). As mentioned in recent studies (Jin et al., 2020), the challenges are preserving the following properties: 1) *human prediction consistency*, misleading the PLMs while keeping human judges unchanged; 2) *semantic similarity*, keeping the semantics of the original inputs; 3) *language fluency*, ensuring the correctness of grammar.

*Equal Contribution
†Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2023, pages 7322–7336
July 9-14, 2023 ©2023 Association for Computational Linguistics

Mainstream solutions are typically a detached two-stage framework. Specifically, they first rank the importance scores of all tokens according to the original input and then orderly substitute these tokens via heuristic rules. Previous studies propose different strategies to rank the editing order of tokens, such as temporal-based algorithm (Gao et al., 2018), probability-weighted saliency (Ren et al., 2019; Li et al., 2020b,a; Jin et al., 2020), and gradient-based ranking (Yoo and Qi, 2021). However, these methods face two limitations. On the one hand, they use a threshold to filter the unsatisfactory substitutions at last, but neglect to integrally consider the properties during computing importance scores. On the other hand, their editing order only depends on the original input without considering the subsequent influence of substitution, as computing the importance score at each step is computationally burdensome in practice.

To solve the issues mentioned above, in this paper, we formally propose to transform the adversarial attack problem into a **sequential decision-making** task as shown in Figure 1. Rather than computing the importance scores all at once based on the original input, we regard the entire attack process as a sequence, where scores in the next step are influenced by the editing results in the current step. Furthermore, there are two types of decision-making problems during each step in the attack sequential process: 1) *word finder*, choosing the appropriate token to edit; 2) *word substitution*, replacing the token with a suitable substitution. Meanwhile, selecting edited tokens at each step should take the attack success rate and crucial properties, such as fluency, into account.

As a sequential decision-making task without a direct signal in each step, we naturally leverage reinforcement learning (RL) to find an appropriate sequential attack path to generate adversaries. In this paper, we propose a model-agnostic method based on policy-based RL for modeling the adversarial attack into Sequential Decision Making, entitled SDM-ATTACK. Given the victim model as the environment with designed reward functions and the original input text as the initial state, the reinforced agent needs to decide on tokens to edit and synonyms to replace sequentially, until it attacks successfully. The experimental results show that SDM-ATTACK achieves the highest attack success rate with a comparable modification rate and semantic similarity to attack fine-tuned BERT against state-of-the-art baselines. Furthermore, we also demonstrate the effectiveness, generalizability, and transferability of SDM-ATTACK in our analysis.

The main contributions of this work are summarized as the following:

- To the best of our knowledge, we are the first to model the adversarial attack on PLMs into a sequential decision-making problem, where the whole attack process is sequential with two decision-making problems, i.e., word finder and word substitution.

- Considering the sequential attack process can receive the final state without any direct intermediate signals, we propose SDM-ATTACK to use reinforcement learning to ask the agent to find an appropriate attack path based on our designed indirect reward signals yielded by the environment.

2 Preliminaries

As for NLP tasks, given a corpus of N input texts $\mathbb{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_N\}$ and an output space $Y = \{y_1, y_2, y_3, \cdots, y_K\}$ containing K labels, the language model F learns a mapping $f : \boldsymbol{x} \rightarrow y$, which learns to classify each input sample $\boldsymbol{x} \in \mathbb{X}$ to the ground-truth label $y_{\text{gold}} \in \mathcal{Y}$:

$$\mathrm{F}(\boldsymbol{x}) = \arg\max_{y_i \in Y} P(y_i|\boldsymbol{x}) \tag{1}$$

The adversary of text $\boldsymbol{x} \in \mathbb{X}$ can be formulated as $\boldsymbol{x}_{\text{adv}} = \boldsymbol{x} + \epsilon$, where ϵ is a slight perturbation to the input $\boldsymbol{x}$. The goal is to mislead the victim model F within a certain constraint $C(\boldsymbol{x}_{\text{adv}})$:

$$\begin{aligned} &\mathrm{F}(\boldsymbol{x}_{\text{adv}}) = \arg\max_{y_i \in Y} P(y_i|\boldsymbol{x}_{\text{adv}}) \neq \mathrm{F}(\boldsymbol{x}), \\ &\text{and } C(\boldsymbol{x}_{\text{adv}}, \boldsymbol{x}) \geq \lambda \end{aligned} \tag{2}$$

where λ is the coefficient, and $C(\boldsymbol{x}_{\text{adv}}, \boldsymbol{x})$ usually calculates the semantic or syntactic similarity (Cer et al., 2018; Oliva et al., 2011) between the input $\boldsymbol{x}$ and its corresponding adversary $\boldsymbol{x}_{\text{adv}}$.

Recently, the adversarial attack task has been framed as a combinatorial optimization problem. However, previous studies (Gao et al., 2018; Ren et al., 2019; Yoo and Qi, 2021) address this problem without considering the subsequent influence of substitution at each step, making attack far from the most effective. In this paper, we formally define the adversarial attack as a sequential decision-making task, where the decisions in the next step are influenced by the results in the current step.

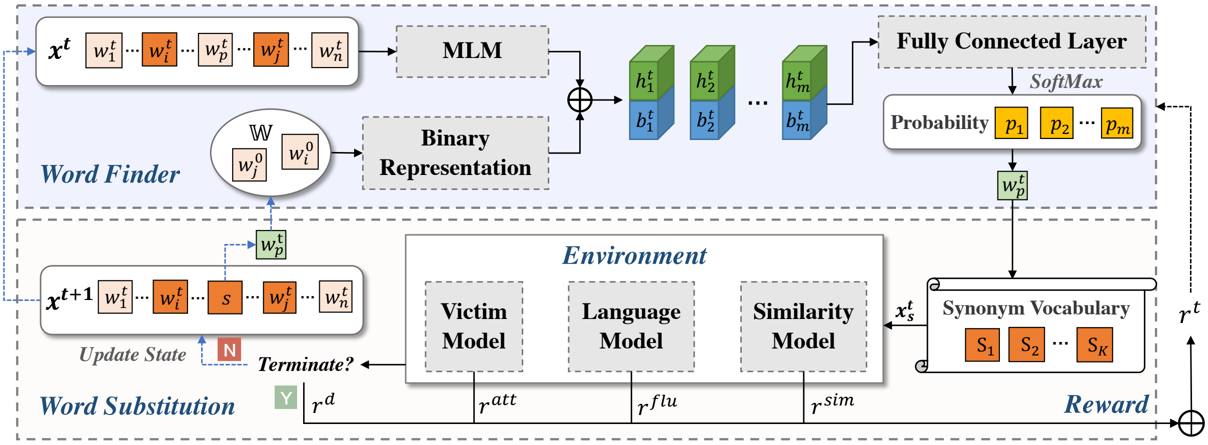

Figure 2: The framework of SDM-ATTACK.

3 Methodology

In this section, we model the adversarial attack on PLMs problem as a sequential decision-making task as shown in Figure 1, where the entire attack process is a sequence with two decision-making problems. Considering the lack of direct signal in each step during the attack process, we propose a model-agnostic method, named SDM-ATTACK, based on policy-based reinforcement learning. The illustration is shown in Figure 2. During each step in the attack process, the reinforced agent needs to take two actions: 1) *word finder*, choosing the appropriate token to edit; and 2) *word substitution*, replacing the token with a suitable substitution. Through an attack sequence toward the input, we obtain its adversary until the attack is successful.

3.1 Environment and Rewards

We regard the victim models (i.e., PLMs) as the whole environment. Intuitively, the agent needs to generate adversaries against the environment and achieve as high a reward as possible. The t-step environment state is our intermediate generation $\boldsymbol{x}^t = [\boldsymbol{w}_1^t, \boldsymbol{w}_2^t, ..., \boldsymbol{w}_n^t]$ containing n words, where the initial state $\boldsymbol{x}^0$ is the original input.

Considering the lack of direct signal in each step, our reward consists of a final discriminant signal r_d to present the state of termination and an instant reward r_t on every step. As for the final signal r_d, once the model prediction of t-step state is different from the initial state, the environment will terminate this episode and yield a *success* signal. However, if the model prediction does not change when all the tokens are replaced or the maximum number of steps is reached, a *failure* signal will be given. Overall, the final signal r_d is denoted as:

$$r_d = \begin{cases} 1, & \textit{success} \\ -1, & \textit{failure} \end{cases} \tag{3}$$

As for the instant reward r_i for each step, we hope that the t-step state $\boldsymbol{x}^t$ can not only mislead the victim model but also ensure semantics similarity and fluency. Firstly, we design one instant reward to evaluate attack success rates:

$$r_t^{att} = \begin{cases} r_d, & \textit{terminated} \\ P(y_{\text{gold}}|\boldsymbol{x}^{t-1}) - P(y_{\text{gold}}|\boldsymbol{x}^t), & \textit{survive} \end{cases} \tag{4}$$

where r_d is the final reward if the current episode is terminated. Secondly, we define a punishment by using an auto-regressive language model (LM) to measure fluency:

$$r_t^{flu} = \sum_i \frac{1}{|\boldsymbol{x}^t|}(\text{LM}(x_i|\boldsymbol{x}^t) - \text{LM}(x_i|\boldsymbol{x}^{t-1})) \tag{5}$$

where $\text{LM}(x_i|\boldsymbol{x}^t)$ is the cross-entropy loss of the token x_i in sentence $\boldsymbol{x}^t$. Thirdly, we also add semantic similarity constraints as another punishment:

$$r_t^{sim} = \text{Sim}(\boldsymbol{x}, \boldsymbol{x}^{t-1}) - \text{Sim}(\boldsymbol{x}, \boldsymbol{x}^t) \tag{6}$$

Finally, our overall instant reward r_t is defined as:

$$r_t = \beta_1 r_t^{att} - \beta_2 r_t^{flu} - \beta_3 r_t^{sim} \tag{7}$$

3.2 Decision Making

During each step in the whole attack process, there are two types of decision-making problems. The first is choosing the appropriate token to edit, while the second is replacing the token with a suitable substitution. In RL, the agent needs to determine the decisions according to the yielded rewards.

Word Finder To find the appropriate token to edit, we first employ the masked language models (MLM) as an encoder to represent the state $\boldsymbol{x}^t$. Due to the setup of the sub-word tokenizer in MLM, the encoder first converts $\boldsymbol{x}^t$ to a token sequence $\boldsymbol{x}^t_{token} = [\boldsymbol{o}^t_1, \boldsymbol{o}^t_2, ..., \boldsymbol{o}^t_m]$. We reverse the conversion mapping $\phi : \boldsymbol{x}^t_{token} \rightarrow \boldsymbol{x}^t$ to recover tokens into words in need. Then we obtain the hidden states $\boldsymbol{h}^t = [\boldsymbol{h}^t_1, \boldsymbol{h}^t_2, ..., \boldsymbol{h}^t_m]$, where $\boldsymbol{h}^t_i \in \mathbb{R}^d$ is the hidden state of token $\boldsymbol{o}^t_i$ with d dimensions.

Furthermore, we maintain a word set $\mathbb{W}$ to restore the words of $\boldsymbol{x}$ that have been already modified as well as stop words and punctuation. We then adopt a simple binary representation $\boldsymbol{b}^t$ according to the word set $\mathbb{W}$:

$$\boldsymbol{b}^t_i = \begin{cases} \mathbf{0} \in \mathbb{R}^d & \phi(\boldsymbol{o}^t_i) \in \mathbb{W} \\ \mathbf{1} \in \mathbb{R}^d & \phi(\boldsymbol{o}^t_i) \notin \mathbb{W} \end{cases} \tag{8}$$

Then, we fuse both the hidden states $\boldsymbol{h}^t_i$ and the binary representation $\boldsymbol{b}^t_i$ to obtain the final representation $\boldsymbol{e}^t_i$ of the environment states:

$$\boldsymbol{e}^t_i = [\boldsymbol{h}^t_i; \boldsymbol{b}^t_i] \tag{9}$$

where [;] denotes the concatenation operation.

During the process of training, we first adopt a simple linear layer to obtain the probability and further normalize it into a distribution. The probability distribution $p(\boldsymbol{o}^t_i|\boldsymbol{x}^t)$ of each token at t-step can be calculated as follows:

$$p(\boldsymbol{o}^t_i|\boldsymbol{x}^t) = \mathrm{softmax}(W \cdot \boldsymbol{e}^t_i + b) \tag{10}$$

where W, b are the weight matrix and the bias vector, respectively. Then the agent samples the word $\boldsymbol{w}^t$ to substitute according to the distribution and ensures the sampled word is not in the word set $\mathbb{W}$.

During the evaluation, the agent will directly select the token with the maximum probability at each step, which is formulated as follows:

$$\boldsymbol{w}^t = \arg\max p(\boldsymbol{o}^t_i|\boldsymbol{x}^t), \phi(\boldsymbol{o}^t_i) \notin \mathbb{W} \tag{11}$$

If the selected token $\boldsymbol{w}^t$ is a sub-word, we reverse the sub-word into a complete word via the conversion mapping ϕ as the newly selected word.

Word Substitution Following Jin et al. (2020), we adopt synonym substitution as our strategy after obtaining selected word $\boldsymbol{w}^t$ in t-step. Firstly, we gather a synonym set $\mathbb{S}_{\boldsymbol{w}^t}$ for $\boldsymbol{w}^t$ that contains top-k candidates from the external vocabulary, computing via cosine similarity (Mrkšić et al., 2016). Then, for each $\boldsymbol{s} \in \mathbb{S}_{\boldsymbol{w}^t}$, we replace $\boldsymbol{w}^t_p$ with $\boldsymbol{s}$ in the sentence $\boldsymbol{x}^t$ to get a substitution $\boldsymbol{x}^t_{\boldsymbol{s}} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_{p-1}, \boldsymbol{s}, \boldsymbol{w}_{p+1}, ...\boldsymbol{w}_n]$. Finally, according to the instant reward r_t in the Equation 4, we select the substitution with the highest reward as the final adversaries $\boldsymbol{x}^t_{\text{adv}}$. Meanwhile, the environment states further updates as follows:

$$\begin{cases} \boldsymbol{x}^{t+1} = \boldsymbol{x}^t_{\text{adv}} \\ \mathbb{W} = \mathbb{W} \cup \{\boldsymbol{w}^t_p\} \end{cases} \tag{12}$$

3.3 Agent Training

The training target is to maximize the total return $G(\tau)$, which is an accumulated reward based on the instant reward r_t , defined in Equation 7, with a discount factor $\gamma \in [0, 1)$:

$$G(\tau) = \sum_{t=1}^{T} \gamma^t r_t \tag{13}$$

The expected return of the decision trajectory, i.e., attack path, is defined as follows:

$$J(\theta) = \mathbb{E}[G(\tau)] \tag{14}$$

Furthermore, we regard the agent as π_θ with parameters θ and the attack path as $\tau = [(a^f_1, a^s_1), \cdots, (a^f_T, a^s_T)]$, where a^f_t and a^s_t represent actions of *word finder* and *substitution* in t-th step, respectively. The probability of this attack path is calculated as $\pi_\theta(\tau) = \prod_{t=1}^{T} \pi_\theta((a^f_t, a^s_t)|s_t)$, where $\pi_\theta((a^f_t, a^s_t)|s_t)$ is the probability of actions in step t based on current environment state s_t. Meanwhile, we consider a^s_t a prior knowledge so that this probability can be simplified. The gradient is calculated by REINFORCE algorithm (Kaelbling et al., 1996):

$$\nabla J(\theta) = \nabla \mathbb{E}[\log \pi_\theta(\tau) \cdot G(\tau)] \tag{15}$$

Detailed information of reinforce training is shown in appendix B.

4 Experiments

4.1 Experimental Setups

Tasks and Datasets Following Li et al. (2020b); Jin et al. (2020), we evaluate the effectiveness of SDM-ATTACK mainly on two standard NLP tasks, text classification and textual entailment. As for text classification, we use diverse datasets from different aspects, including news topic classification (AG's News; Zhang et al., 2015), sentence-level sentiment analysis (MR; Pang and Lee, 2005) and document-level sentiment analysis (IMDB[1] and

[1] https://datasets.imdbws.com/

Dataset	Method	A-rate$\uparrow$	Mod$\downarrow$	Sim$\uparrow$	Dataset	Method	A-rate $\uparrow$	Mod $\downarrow$	Sim$\uparrow$
Yelp	A2T	88.3	**8.1**	0.68	**IMDB**	A2T	89.9	4.4	0.79
	TextFooler	90.5	9.0	0.69		TextFooler	88.7	7.6	0.76
	BERT-Attack	89.8	12.4	0.66		BERT-Attack	88.2	5.3	0.78
	SDM-Attack	**95.8**	8.2	**0.71**		SDM-Attack	**91.4**	**4.1**	**0.82**
AG's News	A2T	53.7	**13.5**	**0.57**	**MR**	A2T	58.5	12.6	0.55
	TextFooler	66.2	18.4	0.52		TextFooler	80.5	15.8	0.50
	BERT-Attack	74.6	15.6	0.52		BERT-Attack	83.2	12.8	0.52
	SDM-Attack	**77.9**	15.3	0.53		SDM-Attack	**85.6**	**12.3**	**0.57**
SNLI	A2T	70.8	17.2	0.35	**MNLI**	A2T	66.0	14.4	0.45
	TextFooler	84.3	17.2	0.38		TextFooler	76.5	15.0	0.45
	BERT-Attack	81.9	16.5	0.38		BERT-Attack	78.1	14.0	0.46
	SDM-Attack	**85.5**	**15.9**	**0.43**		SDM-Attack	**78.7**	**13.8**	**0.49**

Table 1: Automatic evaluation results of attack success rate (A-rate), modification rate (Mod), and semantic similarity (Sim). ↑ represents the higher the better and ↓ means the opposite. The results of MNLI dataset are the average performance of MNLI-matched and MNLI-mismatched. The best results are **bolded**, and the second-best ones are underlined.

Yelp Polarity; Zhang et al., 2015). As for textual entailment, we use a dataset of sentence pairs (SNLI; Bowman et al., 2015) and a dataset with multi-genre (MultiNLI; Williams et al., 2017). The statistics of datasets and more details can be found in Appendix A. Following Jin et al. (2020); Alzantot et al. (2018), we attack 1k samples randomly selected from the test set of each task.

Baselines We compare SDM-Attack with recent state-of-the-art studies: 1) TextFooler (Jin et al., 2020): find important words via probability weighted word saliency and then apply substitution with counter-fitted word embeddings. 2) BERT-Attack (Li et al., 2020b): use mask-predict approach to generate adversaries. 3) A2T (Yoo and Qi, 2021): adopt faster search with gradient-based word importance ranking algorithm. We use open-source codes provided by the authors and TextAttack tools (Morris et al., 2020) to implement these baselines. Furthermore, to ensure fairness in comparing baselines and SDM-Attack, we apply constraints to all methods following Morris et al. (2020) in Appendix C.

Victim Models We conduct the main experiments on a standard pre-trained language model BERT following (Jin et al., 2020; Li et al., 2020b). To detect the generalization of SDM-Attack, we explore the effects on more typical models as discussed in Section 5.1. All victim models are pretrained from TextAttack (Morris et al., 2020).

Implementation Details We adopt BERT as the MLM model in word finder and GPT-2 (Radford et al., 2019) to measure fluency when computing rewards. To keep instant reward and punishment in a similar range, we set the hyper-parameters β_1 to be 1, β_2 to be 1 and β_3 to be 0.2. Moreover, the discount factor γ is set to be 0.9 to achieve a trade-off between instant reward and long-term return. We set the episode number as $M = 200$ and the learning rate as $\alpha = 3e^{-6}$ with Adam as the optimizer. In word substitution, the parameter K of the synonyms number is 50. Our experiments are conducted on a single NVIDIA 2080ti.

Automatic Evaluation Metrics Following previous studies (Jin et al., 2020; Morris et al., 2020), we use the following metrics as the evaluation criteria. 1) Attack success rate (A-rate): the degraded performance after attacking target model. 2) Modification rate (Mod): the percentage of modified words comparing to original text. 3) Semantic similarity (Sim): the cosine similarity between the original text and its adversary, computing via the universal sentence encoder (USE; Cer et al., 2018).

Manual Evaluation Metrics We further manually validate the quality of the adversaries from three challenging properties. 1) Human prediction consistency (Con): the rate of human judgement which is consistent with ground-truth label; 2) Language fluency (Flu): the fluency score of the sentence, measured on a Likert scale of 1 to 5 from ungrammatical to coherent (Gagnon-Marchand et al., 2019); 3) Semantic similarity (Sim_{hum}): the semantic consistency between each input-adversary pair, where 1 means *unanimous*, 0.5 means *ambiguous*, 0 means *inconsistent*.

Dataset		Con↑	Flu↑	Sim_{hum} ↑
IMDB	Original	0.95	4.5	0.95
	SDM-ATTACK	0.90	4.3	
MNLI	Original	0.88	4.0	0.83
	SDM-ATTACK	0.79	3.7	

Table 2: Manual evaluation results comparing the original input and generated adversary by SDM-ATTACK of human prediction consistency (Con), language fluency (Flu), and semantic similarity (Sim_{hum}).

4.2 Results

Automatic Evaluation As shown in Table 1, SDM-ATTACK consistently achieves the highest attack success rate to attack BERT in both text classification and textual entailment tasks, which indicates the effectiveness of SDM-ATTACK. Furthermore, SDM-ATTACK mostly obtains the best performance of modification and similarity metrics, except for AG's News, where SDM-ATTACK achieves the second-best. For instance, our framework only perturbs 4.1% of the words on the IMDB datasets, while the attack success rate is improved to 91.4% with a semantic similarity of 0.82. Although A2T performs better in modification and similarity metrics in Yelp and AG's News, their attack success rate is always much lower than SDM-ATTACK, even other baselines. Because the modification and similarity metrics only consider the successful adversaries, we conjecture that A2T can only solve the inputs which are simpler to attack. In general, our method can simultaneously satisfy the high attack success rate with a lower modification rate and higher similarity. Furthermore, We find that the attack success rate on document-level datasets, i.e., Yelp and IMDB, are higher than the other sentence-level datasets, which indicates that it is easier to mislead models when the input text is longer. The possible reason is the victim model tends to use surface clues rather than understand them to make predictions when the context is long.

Manual evaluation In manual evaluation, we first randomly select 100 samples from successful adversaries in IMDB and MNLI datasets and then ask three crowd-workers to evaluate the quality of the original inputs and our generated adversaries. The results are shown in Table 2. As for the human prediction consistency, we regard the original inputs as a baseline. Taking IMDB as an example, humans can correctly judge 95% of the original inputs while they can maintain 90% accuracy to our generated adversaries, which indicates SDM-ATTACK can mislead the PLMs while keeping human judges unchanged. The language fluency scores of adversaries are close to the original inputs, where the gap scores are within 0.3 on both datasets. Furthermore, the semantic similarity scores between the original inputs and our generated adversaries are 0.95 and 0.83 in IMDB and MNLI, respectively. In general, SDM-ATTACK can satisfy the challenging demand of preserving the three aforementioned properties. Detailed design of manual evaluation and more results are shown in appendix E.

Dataset	Model	A-rate↑	Mod↓	Sim↑
MR	RoBERTa	84.4	13.9	0.52
	WordCNN	72.1	10.3	0.48
	WordLSTM	80.7	8.9	0.56
IMDB	RoBERTa	88.3	8.3	0.70
	WordCNN	89.2	3.3	0.85
	WordLSTM	89.8	5.4	0.75
SNLI	InferSent	78.7	17.0	0.42
	ESIM	79.0	17.2	0.41

Table 3: Attack results against other models.

Dataset	Method	A-rate↑	Mod↓	Sim↑
AG's News	BERT-Attack	74.6	15.6	0.52
	SDM-ATTACK-mlm	76.2	15.0	0.51
MR	BERT-Attack	83.2	12.8	0.52
	SDM-ATTACK-mlm	84.3	11.5	0.53

Table 4: Attack results of different substitution strategies, where SDM-ATTACK-mlm is replaced with the same strategy of word finder as BERT-Attack.

5 Analyses

5.1 Generalization

We detect the generalization of SDM-ATTACK in two aspects, 1) attack more language models and 2) adapt to more substitution strategies. Firstly, we apply SDM-ATTACK to attack extensive victim models, such as traditional language models (e.g., WordCNN) and other state-of-the-art PLMs (e.g., RoBERTa; Liu et al., 2019). The results of text classification tasks in table 3 show that SDM-ATTACK not only has better attack effects against WordCNN and WordLSTM, but also misleads RoBERTa, which is a more robust model. For example, on the IMDB datasets, the attack success rate is up to 89.2% against WordCNN with a modification rate of only about 3.3% and a high semantic similarity of 0.85. As for the textual entailment task, SDM-ATTACK can also achieve remarkable attack

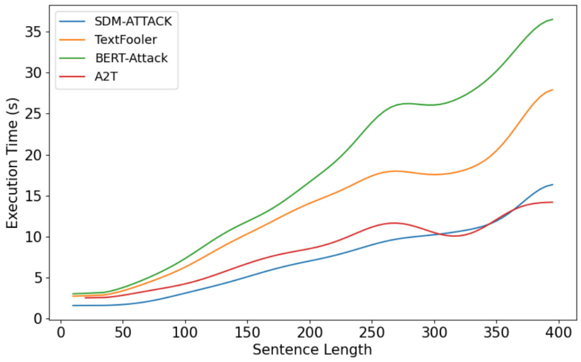

Figure 3: The time cost according to varying sentence lengths in the IMDB dataset, smoothed with Gaussian function where kernel size is 5.

success rates against InferSent and ESIM.

Secondly, although we directly adopt the word substitution strategy in Textfooler, this strategy can actually be replaced by other methods. To demonstrate this assumption, we further replace our word substitution strategy with the mask-fill way in BERT-attack, named SDM-ATTACK-mlm. As shown in Table 4, SDM-ATTACK-mlm completely beat BERT-Attack, indicating the part of word substitution of SDM-ATTACK has generalization ability to extend to different types of strategies and archives high performance. More results are displayed in appendix E.

5.2 Efficiency

In this section, we probe the efficiency according to varying sentence lengths in the IMDB dataset as shown in Figure 3. The time cost of SDM-ATTACK is surprisingly mostly better than A2T, which mainly targets obtaining cheaper computation costs with lower attack success rates in Table 1. Meanwhile, SDM-ATTACK can obviously beat BERT-attack and TextFooler, which need to conduct a model forward process for each token. Furthermore, with the increase of sentence lengths, SDM-ATTACK and A2T maintain a stable time cost, while the time cost of BERT-attack and TextFooler is exploding. These phenomena show the efficiency advantage of SDM-ATTACK, especially in dealing with long texts.

5.3 Transferability

We evaluate the transferability of SDM-ATTACK to detect whether the SDM-ATTACK trained on one dataset can perform well on other datasets. We conduct experiments on a series of text classification tasks and use the randomly initialized BERT as a baseline. As shown in Table 5, SDM-ATTACK has high transferability scores across different datasets, which are consistently higher than random. In detail, the performances among Yelp, IMDB and MR, which all belong to sentiment analysis, are higher than AG's News. Moreover, IMDB and MR are corpora about movies where SDM-ATTACK tends to learn a general attack strategy in this field and can transfer well to each other.

	Yelp	IMDB	MR	AG's News
Yelp	**87.6**	85.8	40.5	43.6
IMDB	82.9	**89.3**	51.4	43.4
MR	81.8	88.2	**66.5**	39.6
AG's News	62.4	59.2	29.9	**53.2**
Random	58.9	56.1	27.8	38.3

Table 5: Transferability evaluation of SDM-ATTACK generator on text classification task against BERT. Row i and column j is the attack success rate of SDM-ATTACK trained on dataset i evaluated on dataset j.

Dataset	Acc↑	A-rate↑	Mod↓	Sim↑
Yelp	97.4	95.8	8.2	0.71
+Adv Train	97.0	82.5	13.5	0.63
IMDB	91.6	91.4	4.1	0.82
+Adv Train	90.5	79.2	8.5	0.74
SNLI	89.1	85.5	15.9	0.43
+Adv Train	88.2	78.6	17.1	0.42

Table 6: The results of comparing the original training with adversarial training with our generated adversaries. More results can be found in Appendix D.

5.4 Adversarial Training

We further investigate to improve the robustness of victim models via adversarial training. Specifically, we fine-tune the victim model with both original training datasets and our generated adversaries, and evaluate it on the same test set. As shown in Table 6, compared to the results with the original training datasets, adversarial training with our generated adversaries can maintain close accuracy, while improving performance on attack success rates, modification rates, and semantic similarity. The victim models with adversarial training are more difficult to attack, which indicates that our generated adversaries have the potential to serve as supplementary corpora to enhance the robustness of victim models.

Method	**Text** (MR; Negative)	**Result**	**Mod↓**	**Sim↑**	**Flu↑**
Original	Davis is so enamored of her own creation that she can not see how insufferable the character is.	-	-	-	5
A2T	Davis is so enamored of her own institution that she can not behold how unforgivable the hallmark is.	*Failure*	22.2	0.16	3
TextFooler	Davis is well enamored of her own infancy that she could not admire how infernal the idiosyncrasies is.	*Success*	33.3	0.23	3
BERT-Attack	Davis is often enamoted of her own generation that she can not see how insuffoure the queen is.	*Failure*	27.8	0.09	2
SDM-ATTACK	Davis is so captivated of her own creation that she can't see how indefensible the character is.	*Success*	11.1	0.57	5

Table 7: Adversaries generated by SDM-ATTACK and baselines in MR dataset. The replaced words are highlighted in blue. *Failure* indicates the adversary fails to attack the victim model and *success* means the opposite.

5.5 Case Study

Table 7 shows adversaries produced by SDM-ATTACK and the baselines. Overall, the performance of SDM-ATTACK is significantly better than other methods. For this sample from the MR dataset, only TextFooler and SDM-ATTACK successfully mislead the victim model, i.e., changing the prediction from *negative* to *positive*. However, TextFooler modifies twice as many words as SDM-ATTACK, demonstrating our work has found a more suitable modification path. Adversaries generated by A2T and BERT-Attack are failed samples due to the low semantic similarity. BERT-Attack even generates an invalid word "*enamoted*" due to its subword combination algorithm. We also ask crowdworkers to give a fluency evaluation. Results show SDM-ATTACK obtains the highest score of 5 as the original sentence, while other adversaries are considered difficult to understand, indicating SDM-ATTACK can generate more natural sentences.

6 Related Work

Adversarial attack has been well-studied in image and speech domains (Szegedy et al., 2013; Chakraborty et al., 2018; Kurakin et al., 2018; Carlini and Wagner, 2018). However, due to the discrete nature of language, the adversarial attack against pre-trained language models is much more difficult. Earlier works mainly focus on designing heuristic rules to generate adversaries, including swapping words (Wei and Zou, 2019), transforming syntactic structure (Coulombe, 2018), and paraphrasing by back-translation (Ribeiro et al., 2018; Xie et al., 2020). However, these rule-based methods are label-intensive and difficult to scale.

Recently, adversarial attack in NLP is framed as a combinatorial optimization problem. Mainstream studies design a series of search algorithms with two detached stages In the first stage, they iteratively search for modification positions, including saliency-based ranking (Liang et al., 2017; Ren et al., 2019; Jin et al., 2020; Garg and Ramakrishnan, 2020), gradient-based descent algorithm (Sato et al., 2018; Yoo and Qi, 2021), and temporal-based searcher (Gao et al., 2018). In the second stage, a series of studies designs different substitution strategies, including dictionary method (Ren et al., 2019), word embeddings (Kuleshov et al., 2018; Jin et al., 2020) or language models (Li et al., 2020b; Garg and Ramakrishnan, 2020; Li et al., 2020a). In this paper, we formally propose to define the adversarial attack task as a sequential decision-making problem, further considering that scores in the next step are influenced by the editing results in the current step.

The other line of recent studies is sampling-based methods. Alzantot et al. (2018) and Wang et al. (2019) apply genetic-based algorithm, Zang et al. (2019) propose a particle swarm optimization-based method, and Guo et al. (2021) generate adversaries via distribution approximate sampling. However, their execution time is much more expensive due to the properties of sampling, so it is unlikely to generate large-scale adversarial samples. In addition, Zou et al. (2019) conducts reinforcement learning on attacking the neural machine translation task, but their search path is fixed from left to right. In this paper, SDM-ATTACK can determine any search order to find the appropriate attack path.

7 Conclusion

In this paper, we formally define the adversarial attack task as a sequential decision-making problem, considering the entire attack process as sequence with two types of decision-making problems, i.e., word finder and substitution. To solve this problem without any direct signals of intermediate steps, we propose to use policy-based RL to find an appropriate attack path, entitled SDM-ATTACK. Our experimental results show that SDM-ATTACK achieves the highest attack success rate. In this paper, we use our designed rewards as instant signals to solve these two decision-making problems approximately. We will further try to adopt hierarchical RL to optimize the solution.

8 Limitations

We define the adversarial attack task as a sequential decision-making problem and apply policy-based reinforcement learning to model it. This work must follow this assumption: the decision process conforms to Markov decision process (MDP) that the conditional probability distribution of the future state depends only on the current state. Meanwhile, reinforcement learning training requires additional time costs and the results may be unstable.

We only conduct the experiments on two NLP tasks with six selected datasets, which are all English corpus. Furthermore, our experimental results are mainly for BERT, with RoBERTa supplemented in the analysis. Thus, we lack the evaluation of other novel pre-trained language models, such as ELECTRA (Clark et al., 2020) and XLNET (Yang et al., 2019). Therefore, our work lacks multi-task, multi-model and multilingual verification in terms of generalization and transferability.

9 Ethics Statement

We declare that this article is in accordance with the ethical standards of *ACL Code of Ethics*. Any third party tools used in this work are licensed from their authors. All crowd-workers participating in the experiments are paid according to the local hourly wages.

10 Acknowledgment

We would like to thank anonymous reviewers for their insightful and constructive feedback. We appreciate Peng Li and Shuo Wang for their valuable discussions. We thank Qianlin Liu, Yanqi Jiang and Yiwen Xu for the crowdsourced work. This work is supported by the National Key R&D Program of China (2022ZD0160502) and the National Natural Science Foundation of China (No. 61925601, 62276152, 62236011).

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*.

David F Armstrong, William C Stokoe, and Sherman E Wilcox. 1995. *Gesture and the nature of language*. Cambridge University Press.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Nicholas Carlini and David Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE security and privacy workshops (SPW)*, pages 1–7. IEEE.

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*.

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Claude Coulombe. 2018. Text data augmentation made simple by leveraging nlp cloud apis. *arXiv preprint arXiv:1812.04718*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jules Gagnon-Marchand, Hamed Sadeghi, Md Haidar, Mehdi Rezagholizadeh, et al. 2019. Salsa-text: self attentive latent space based adversarial text generation. In *Canadian Conference on Artificial Intelligence*, pages 119–131. 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.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based adversarial attacks against text transformers. *arXiv preprint arXiv:2104.13733*.

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.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. 2018. Adversarial examples for natural language classification problems.

Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. 2018. Adversarial attacks and defences competition. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 195–231. Springer.

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*.

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*.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2017. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Gary Marcus. 2020. The next decade in ai: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.

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*.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Jesús Oliva, José Ignacio Serrano, María Dolores Del Castillo, and Ángel Iglesias. 2011. Symss: A syntax-based measure for short-text semantic similarity. *Data & Knowledge Engineering*, 70(4):390–405.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. 2018. Interpretable adversarial perturbation in input embedding space for text. *arXiv preprint arXiv:1805.02917*.

Michael Studdert-Kennedy. 2005. How did language go discrete. *Language origins: Perspectives on evolution, ed. M. Tallerman*, pages 48–67.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Scott Thiebes, Sebastian Lins, and Ali Sunyaev. 2021. Trustworthy artificial intelligence. *Electronic Markets*, 31(2):447–464.

X Wang, H Jin, and K He. 2019. Natural language adversarial attacks and defenses in word level. *arXiv preprint arXiv:1909.06723*.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

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

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33:6256–6268.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of nlp models. *arXiv preprint arXiv:2109.00544*.

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*.

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

Wei Zou, Shujian Huang, Jun Xie, Xinyu Dai, and Jiajun Chen. 2019. A reinforced generation of adversarial examples for neural machine translation. *arXiv preprint arXiv:1911.03677*.

A Datasets

We conduct experiments on the following datasets of two NLP tasks and detailed statistics are displayed in Table 8:

- **Text Classification**: (1) Yelp (Zhang et al., 2015): A dataset for binary sentiment classification on reviews, constructed by considering stars 1 and 2 negative, and 3 and 4 positive. (2) IMDB: A document-level movie review dataset for binary sentiment analysis. (3) MR (Pang and Lee, 2005): A sentence-level binary classification dataset collected from Rotten Tomatoes movie reviews. (4) AG's News (Zhang et al., 2015): A collection of news articles. There are four topics in this dataset: World, Sports, Business, and Science/Technology.
- **Textual Entailment**: (1) SNLI (Bowman et al., 2015): A dataset of human-written English sentence pairs and manually annotated labels of entailment, neutral and contradiction. (2) MNLI (Williams et al., 2017): Another crowd-sourced collection of sentence pairs labeled with textual entailment information. Compare to SNLI, it includes more complex sentences, e.g, enres of spoken and written text.

Dataset	Train	Test	Avg Len	Classes
Yelp	560k	38k	152	2
IMDB	25k	25k	215	2
AG's News	120k	7.6k	73	4
MR	9k	1k	20	2
SNLI	570k	3k	8	3
MNLI	433k	10k	11	3

Table 8: Overall statistics of datasets.

B Training Algorithm

The training process is shown in Algorithm 1. Since a_t^f is chosen through a probability distribution, the agent is encouraged to explore more possible paths. The instant reward r_t is obtained from environment after performing both two actions actions. Once the termination signal is raised, the environment will terminate this current episode and update the agent's parameters via a policy gradient approach. The expected return of decision trajectory is defined as follows:

$$J(\theta) = \mathbb{E}[G(\tau)] \tag{16}$$

Thus the gradient is calculated by REINFORCE algorithm (Kaelbling et al., 1996):

$$\nabla J(\theta) = \nabla \mathbb{E}[\log \pi_\theta(\tau) \cdot G(\tau)] \tag{17}$$

Then the expectation over the whole sequence is approximated by Monte Carlo simulations and can be expressed as follows:

$$\nabla J(\theta) = \frac{1}{M} \sum_{m=1}^{M} \nabla \log \pi_\theta(\tau^{(m)}) G(\tau^{(m)}) \tag{18}$$

where $[\tau^{(1)}, \tau^{(2)}, ..., \tau^{(M)}]$ are M samples of trajectories. The discount factor γ enables both long-term and immediate effects to be taken into account and trajectories with shorter lengths are encouraged.

Algorithm 1 Reinforce Training

```
Initialization: agent $\pi_\theta$ with parameters $\theta$, episode number $M$
for $i \leftarrow 1$ to M do
    initialize $t \leftarrow 1$
    while not receive termination signal do
        get environment state $s_t$
        compute $\pi_\theta((a_t^f, a_t^s)|s_t) \backsim \pi_\theta(a_t^f|s_t)$
        sample $a_t^f$ based on probability
        select $a_t^s$ from prior knowledge
        compute reward $r_t$
        update $t \leftarrow t + 1$
    end while
    initialize $G(\tau) \leftarrow 0$
    for $j \leftarrow T$ to 1 do
        $G(\tau) \leftarrow \gamma G(\tau) + r_j$
        accumulate $J_j(\theta)$
    end for
    update $\theta \leftarrow \theta + \alpha \nabla J(\theta)$
end for
```

We randomly select 2500 items from the training corpus for training the agent of each dataset. The average convergence time is approximately between 2-16 hours, related to the length of the input. When attacking large batches of samples, the impact of training cost is negligible compared to the cumulative attack time cost. During training, We adopt random strategies and short-sighted strategies in the initial stage for early exploration and to obtain better seeds.

C Implementation Constraint

In order to make the comparison fairer, we set the following constraints for SDM-ATTACK as well as all baselines: (1) **Max modification rate**: To better

Dataset	Acc↑	A-rate↑	Mod↓	Sim↑
Yelp	97.4	95.8	8.2	0.71
+Adv Train	97.0	82.5	13.5	0.63
IMDB	91.6	91.4	4.1	0.82
+Adv Train	90.5	79.2	8.5	0.74
AG's News	94.6	77.9	15.3	0.53
+Adv Train	91.8	50.6	23.3	0.50
MR	96.9	85.6	12.3	0.57
+Adv Train	92.4	72.0	16.7	0.57
SNLI	89.1	85.5	15.9	0.43
+Adv Train	88.2	78.6	17.1	0.42
MNLI	84.5	78.7	13.8	0.49
+Adv Train	76.8	58.6	15.2	0.49

Table 9: Adversarial training results.

maintain semantic consistency, we only keep adversarial samples with less than 40% of the words to be perturbed. (2) **Part-of-speech (POS)**: To generate grammatical and fluent sentences, we use NLTK tools[2] to filter candidates that have a different POS from the target word. This constraint is not employed on BERT-Attack. (3) **Stop words preservation**: the modification of stop words is disallowed and this constraint helps avoid grammatical errors. (4) **Word embedding distance**: For Textfooler, A2T and SDM-ATTACK, we only keep candidates with word embedding cosine similarity higher than 0.5 from synonyms dictionaries (Mrkšić et al., 2016). For *mask-fill* methods, following BERT-Attack, we filter out antonyms (Li et al., 2020b) via the same synonym dictionaries for sentiment classification tasks and textual entailment tasks.

D Tuning with Adversaries

Table 9 displays adversarial training results of all datasets. Overall, after fine-turned with both original training datasets and adversaries, victim model is more difficult to attack. Compared to original results, accuracy of all datasets is barely affected, while attack success rate meets an obvious decline. Meanwhile, attacking model with adversarial training leads to higher modification rate, further demonstrating adversarial training may help improve robustness of victim models.

E Supplementary Results

At the beginning of manual evaluation, we provided some data to allow crowdsourcing workers to unify the evaluation standards. We also remove the data with large differences when calculating the average value to ensure the reliability and accuracy of the evaluation results. More manual evaluation results are shown in Table 10.

Dataset		Con↑	Flu↑	Sim_{hum} ↑
IMDB	Original	0.95	4.5	
	TextFooler	0.84	4.0	0.88
	Bert-Attack	0.83	4.2	0.90
	SDM-ATTACK	0.90	4.3	0.95
MNLI	Original	0.88	4.0	
	TextFooler	0.77	3.5	0.80
	Bert-Attack	0.77	3.6	0.81
	SDM-ATTACK	0.79	3.7	0.83

Table 10: Manual evaluation results comparing the original input and generated adversary by attack method of human prediction consistency (Con), language fluency (Flu), and semantic similarity (Sim_{hum}).

Dataset	Method	A-rate↑	Mod↓	Sim↑
Yelp	BERT-Attack	89.8	12.4	0.66
	SDM-ATTACK-mlm	90.0	10.6	0.65
IMDB	BERT-Attack	88.2	5.3	0.78
	SDM-ATTACK-mlm	88.5	5.1	0.78
AG's News	BERT-Attack	74.6	15.6	0.52
	SDM-ATTACK-mlm	76.2	15.0	0.51
MR	BERT-Attack	83.2	12.8	0.52
	SDM-ATTACK-mlm	84.3	11.5	0.53

Table 11: Attack results of different substitution strategies, where SDM-ATTACK-mlm is replaced with the same strategy of word finder as BERT-Attack.

Table 11 displays the generalization ability of SDM-ATTACK with mask-fill strategy. However, the improvement effect is not particularly obvious. The mask-fill method makes the current candidate synonyms also affected by the sequence states. Compared to a fixed synonym dictionary, it has a larger prior knowledge and changing action space, which makes it harder to train the agent. Only increasing the size of the training corpus is not very effective. We will try adopting hierarchical RL to further solve this problem in the future.

[2] https://www.nltk.org/

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
8

☒ A2. Did you discuss any potential risks of your work?
There is no risks in our work.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
We write this paper ourselves.

B ☑ Did you use or create scientific artifacts?

4,9

☑ B1. Did you cite the creators of artifacts you used?
4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
9

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
4

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Our experiments are performed on public datasets.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
4, appendix

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
We provide statistics of datasets in appendix and detail information about baselines in section 4.

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?
4

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
4

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
9

☒ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
We have used publicly available datasets.

☒ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Our work does not involve collecting ethically relevant information.

☑ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
4