SHARP: Search-Based Adversarial Attack for Structured Prediction

Liwen Zhang[‡◇], **Zixia Jia**[‡◇], **Wenjuan Han**[‡†], **Zilong Zheng**[†], **Kewei Tu**[◇*]
[◇]School of Information Science and Technology, ShanghaiTech University
[◇]Shanghai Engineering Research Center of Intelligent Vision and Imaging
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences
[†]Beijing Institute for General Artificial Intelligence
{zhanglw1,jiazx,tukw}@shanghaitech.edu.cn
{hanwenjuan,zlzheng}@bigai.ai

Abstract

Adversarial attack of structured prediction models faces various challenges such as the difficulty of perturbing discrete words, the sentence quality issue, and the sensitivity of outputs to small perturbations. In this work, we introduce SHARP, a new attack method that formulates the black-box adversarial attack as a search-based optimization problem with a specially designed objective function considering sentence fluency, meaning preservation and attacking effectiveness. Additionally, three different searching strategies are analyzed and compared, *i.e.*, Beam Search, Metropolis-Hastings Sampling, and Hybrid Search. We demonstrate the effectiveness of our attacking strategies on two challenging structured prediction tasks: part-of-speech (POS) tagging and dependency parsing. Through automatic and human evaluations, we show that our method performs a more potent attack compared with pioneer arts. Moreover, the generated adversarial examples can be used to successfully boost the robustness and performance of the victim model via adversarial training.

1 Introduction

Adversarial attacking aims to mislead the victim model (*e.g.*, a trained dependency parser) to produce erroneous outputs when feeding adversarial examples. The process can be seen in Figure 1. Adversarial training improves the victim model in terms of performance and robustness by training on adversarial examples. Since structured prediction tasks such as sequence labeling and dependency parsing are critical building blocks of many natural language processing (NLP) systems, it is essential to study adversarial attacks and defense of structured prediction models (Jia and Liang, 2017; Wang et al., 2019).

	Structured Prediction	Classification
Ratio of GT Change	0.8	0.2
Meaning Preservation	0.94	0.92
Fluency	142	144

Table 1: Perturbation sensitivity comparisons between structured prediction and classification task.

However, multiple technical challenges are faced by attackers of structured prediction models in the NLP area. All adversarial attackers for NLP tasks face general challenges related to gradient computation of discrete inputs, grammatical correctness, and meaning preservation (Zhang et al., 2019a; Jia and Liang, 2017; Wang et al., 2019; Cheng et al., 2019b, 2020b). Another potential but important challenge lies in the sensitivity of structured prediction: small perturbations to input sentences may very likely change the target output structures. In contrast, small perturbations typically do not change sentence classification labels. Han et al. (2020) first qualitatively proposed this assumption. We quantitatively investigate this sensitivity in Table 1. Specifically, we attack two typical models (the sentiment classifier (Ren et al., 2019) for the classification task and the dependency parser (Dozat and Manning, 2017) for structured prediction task) using the widely-used word-substitution attackers PWWS (Ren et al., 2019) and FGSM (Goodfellow et al., 2015) respectively. For both tasks, we generate adversarial examples by substituting words with the same proportion. The adversarial examples have similar qualities: fluency with perplexity 142 vs. 144 and meaning-preservation degree with BLEU 0.94 vs. 0.92. However, when asking annotators to label the adversarial examples, we find that around 80% adversarial examples of

*Kewei Tu is the corresponding author. ‡ These authors contributed equally.

Findings of the Association for Computational Linguistics: NAACL 2022, pages 950 - 961

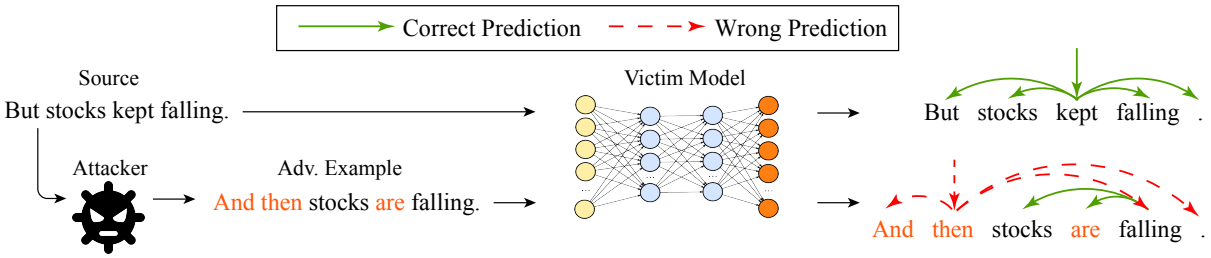


Figure 1: An illustration of attacking process.

the classification task keep the same target outputs as the original input sentences, while only around 20% adversarial examples of the structured prediction task have unchanged output target structures. The huge gap shows the sensitivity of structured prediction tasks, verifying the challenge of attacking structured prediction models.

Despite these challenging issues, recently a few researchers are working on attack of structured predictions. Zheng et al. (2020) tries to preserve the original target output structures by replacing words with the same part of speech tags. Wang et al. (2021) follows a similar method to generate adversarial examples. However, these approaches cannot handle the aforementioned sensitivity. Wang et al. (2021) reveals that the syntactic structures of 25% generated examples of the attacker from Zheng et al. (2020) and 15% from Wang et al. (2021) are changed, although they both carefully design specific rules based on linguistic prior knowledge to preserve the structures. On the other hand, Han et al. (2020) chooses to generate silver structures instead of assuming unchanged target outputs. However, the proposed method of training a sequence-to-sequence adversarial example generator needs time-consuming training and often leads to ungrammatical and unnatural-looking sentences.

To address the above challenges, in this paper, we propose a novel and efficient attack method: **S**earc**H**-based adversarial **A**ttack for st**R**uctured **P**rediction (SHARP)[1]. We formulate black-box adversarial attack as an optimization problem that seeks to maximize a specially designed objective function for better fluency, contextual consistency, and attacking effectiveness. In addition, we use a pretrained masked language model (PLM) to prune candidate sentences when exploring the search space. While our approach can be applied to any structured prediction tasks, in this paper we evaluate our approach on POS tagging models and dependency parsing models. Both automatic and human evaluations show that our method beats previous state-of-the-art (SOTA) approaches by a large margin. We also show that the generated adversarial examples can be used to boost the victim model in terms of accuracy and robustness with adversarial training.

[1]Our implementations are publicly available at https://github.com/JZXXX/SHARP.

2 Preliminaries

2.1 Structured Prediction

Structured prediction in NLP aims to predict a structured output such as a sequence in the POS tagging task or a tree in the dependency parsing task. Given an input sentence $\mathbf{x}$, a structured prediction model predicts the output $\mathbf{y}$ by maximizing the log conditional probability:

$$\arg\max_{\mathbf{y}\in\mathcal{T}} \log P(\mathbf{y}|\mathbf{x})$$

where $\mathcal{T}$ is the set of all possible outputs. The prediction model can be trained by maximizing the log probability of the target structure $\mathbf{y}^*$ given a training set which contains $(\mathbf{x}, \mathbf{y}^*)$ pairs.

Our purpose is to attack a well-trained structured prediction model through searching adversarial sentences. Besides, by leveraging adversarial sentences and the original training data to retrain the model, we can defend against attacks and enhance the model's robustness.

2.2 Adversarial Attack

Let $\mathbf{x}^{ori} = \{w_0^{ori}, w_1^{ori}, ..., w_{N-1}^{ori}\}$ denote an original sentence with N words. The victim model $\mathcal{M}_V(\mathbf{x}) : \mathbf{x} \rightarrow \mathbf{y}$ has been trained to produce a structured prediction output that is close to the golden structure $\mathbf{y}^*$. Then the task for adversarial attack is to fool the model $\mathcal{M}_V$ by feeding an imperceptible adversarial example $\mathbf{x}^{adv}$ such that $\mathcal{M}_V(\mathbf{x}^{adv}) \neq$ target output of $\mathbf{x}^{adv}$.

In this work, we focus on the black-box attack setting, where only the outputs of the victim model $\mathcal{M}_V$ are accessible, while the internal details are

invisible, including the model structure, hyper-parameters, training strategy, the training dataset, and gradients over each layer, *etc*.

3 Search-based Adversarial Attack

3.1 Attacking Objective

Typically, an ideal adversarial natural language example should be: (i) able to fool the victim parser to generate an erroneous output; (ii) fluent and grammatically correct; (iii) semantically consistent with the original sentence $\mathbf{x}^{ori}$. We consider the following objectives to address the above requirements, respectively.

Attacking Effectiveness. Ensuring the error of $\mathcal{M}_V(\mathbf{x}^{adv})$ is non-trivial due to the lack of new ground truth structured outputs. To estimate the new ground truth and further identify if the adversarial sentence can indeed fool the victim model $\mathcal{M}_V$, we follow Han et al. (2020) to make use of two external reference models $\mathcal{M}_A$ and $\mathcal{M}_B$. Because we want the victim model to predict wrong outputs of the adversarial examples, a good adversarial example $\mathbf{x}^{adv}$ should maximize the difference between the predicted structures $\mathcal{M}_V(\mathbf{x}^{adv})$ and reference outputs $(\mathcal{M}_A(\mathbf{x}^{adv}), \mathcal{M}_B(\mathbf{x}^{adv}))$, while minimizing the difference between $\mathcal{M}_A(\mathbf{x}^{adv})$ and $\mathcal{M}_B(\mathbf{x}^{adv})$. Formally, a scoring function can be formulated as

$$\begin{aligned} r(\mathbf{x}) =& sim(\mathcal{M}_A(\mathbf{x}), \mathcal{M}_B(\mathbf{x})) \\ &+ (1 - sim(\mathcal{M}_V(\mathbf{x}), \mathcal{M}_A(\mathbf{x}))) \\ &+ (1 - sim(\mathcal{M}_V(\mathbf{x}), \mathcal{M}_B(\mathbf{x}))) \end{aligned} \quad (1)$$

where $sim(\cdot, \cdot) \in [0, 1]$ is a similarity function, *e.g.*, Directed Dependency Accuracy (DDA) that evaluates the similarity between two parse trees.

Fluency. We use the perplexity of a PLM to evaluate the grammatical correctness and fluency of the generated sentences following Holtzman et al. (2018); Xu et al. (2018); Pang et al. (2020). For a single sentence $\mathbf{x}$, the Perplexity score (PPL) can be computed as

$$f(\mathbf{x}) = PPL(\mathbf{x}) = P(\mathbf{x})^{-\frac{1}{N}} \quad (2)$$

where N denotes the sequence length. A lower perplexity indicates that the sentence is more natural and grammatically correct.

Meaning Preservation. Note that the previous two scores neglect the original sentence $\mathbf{x}^{ori}$ when attacking, which will commonly result in a "zombie" output, *i.e.*, no matter what the input sentence is, the attack always produces exactly the same adversarial sentence. We maintain the diversity of generated sentences by using a score function to ensure the consistency of meanings between the generated sentences and the original sentences. We use BERTSCORE (Zhang et al., 2019b) to evaluate the similarity, which matches each token in $\mathbf{x}$ to a token in $\mathbf{x}^{ori}$ to compute recall, and each token in $\mathbf{x}^{ori}$ to a token in $\mathbf{x}$ to compute precision, finally combines precision and recall to compute an F1 measure.

$$s(\mathbf{x}, \mathbf{x}^{ori}) = \text{BERTSCORE}(\mathbf{x}, \mathbf{x}^{ori}) \quad (3)$$

Such metric correlates better with human judgment than traditional measures such as BLEU (Papineni et al., 2002).

Objective Function. Taking together, the objective of our adversarial attack can be defined as a non-negative function:

$$\mathcal{F}(\mathbf{x}) = \frac{r(\mathbf{x}) \cdot s(\mathbf{x}, \mathbf{x}^{ori})}{f(\mathbf{x})} \quad (4)$$

By maximizing $\mathcal{F}(\mathbf{x})$, we hope to produce $\mathbf{x}^{adv}$ that are natural-sounding, human-imperceptible, and effective in attacking the victim model.

3.2 Optimization-based Search

Our optimization problem can be considered as a T-step sequential decision-making process with its state changes along $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{T-1}\}$ and $\mathbf{x}_0 = \mathbf{x}^{ori}$. At step t, $\mathbf{x}_t$ moves to $\mathbf{x}_{t+1}$ with respect to

$$\begin{aligned} \mathbf{x}_{t+1} &= a_t^*(w_t^*, \mathbf{x}_t, w_c) \\ (a_t^*, w_t^*) &= \underset{a_t \in \mathcal{A}, w_t \in \mathbf{x}_t}{\arg\max}\ \mathcal{F}(a_t(w_t, \mathbf{x}_t, w_c)) \end{aligned} \quad (5)$$

where w_t is a selected word in $\mathbf{x}_t$, and a_t is a word-level manipulation such that $\mathbf{x}_{t+1} = a_t(w_t, \mathbf{x}_t, w_c)$. In this work, we consider three different manipulations $\mathcal{A} = \{Replace, Insert, Delete\}$: $Replace(w^*, \mathbf{x}, w_c)$ indicates replacing the word w^* in sentence $\mathbf{x}$ with another word $w_c \in \mathcal{W}$; $Insert(w^*, \mathbf{x}, w_c)$ inserts a word w_c after w^*; $Delete(w^*, \mathbf{x}, null)$ simply removes w^* from $\mathbf{x}$, $null$ means we do not need another word.

Pruning $\mathcal{W}$. Exploring the entire vocabulary set at each step for $Replace$ and $Insert$ will be time-consuming. Therefore, we prune the search space with a pretrained masked language model (PLM), *e.g.*, RoBERTa (Liu et al., 2019). Specifically, the expected position for w_c in sentence $\mathbf{x}_t$ is replaced with a mask token [*MASK*] and will be predicted using RoBERTa. Then w_c will be selected from the pruned word set $\mathcal{W}$ that includes the top $N_w = 50$ predictions in the masked position according to $P_{RoBERTa}(w[MASK]|\mathbf{x})$.

We consider the following three strategies of exploring the search space: Beam Search (BS), Metropolis-Hastings Sampling (MHS) and Hybrid Search (HS).

Beam Search Traditional beam search creates the beam by exhaustively searching all candidates created with one manipulation and one word from $\mathbf{x}_t$. Then the top k (beam size) candidates that maximize $\mathcal{F}(\mathbf{x})$ are selected and stored in the beam in a greedy manner, each of which will be considered as the input sentence for next step. However, for each sentence, there are a huge number of possible candidates, *i.e.*, different positions for manipulation and different words for replacement or insertion. To reduce the time complexity, at each step, we sample a single type of manipulation $a \in \mathcal{A}$ and a single word $w^* \in \mathbf{x}_t$ on which the manipulation is performed. Therefore, the time complexity for a single step is reduced to $\mathcal{O}(k|\mathcal{W}|)$. Due to the nature of local optimization of BS, we consider such strategy as *exploitation*.

Metropolis-Hastings Sampling To get out of the local optima that BS is commonly stuck in, we use a sampling-based approach – Metropolis-Hastings Sampling (Metropolis et al., 1953; Hastings, 1970; Chib and Greenberg, 1995) – to explore the space beyond the starting local optimum and increase the chance of finding a global optima. We consider this strategy as *exploration*.

Specifically, we can create a proposal $\mathbf{x}_{t+1}$ by sampling an action a_t, a selected word $w \in \mathbf{x}_t$ and a new word $w_c \in \mathcal{W}$. Given the stationary distribution defined as

$$\pi(\mathbf{x}) \propto \mathcal{F}(\mathbf{x}), \tag{6}$$

MHS accepts the proposal with the following rate:

$$\alpha(\mathbf{x}_{t+1}|\mathbf{x}_t) = \min\left\{1, \frac{\pi(\mathbf{x}_{t+1})\,T(\mathbf{x}_t|\mathbf{x}_{t+1})}{\pi(\mathbf{x}_t)\,T(\mathbf{x}_{t+1}|\mathbf{x}_t)}\right\}, \tag{7}$$

$$\begin{aligned} T(\mathbf{x}_{t+1}|\mathbf{x}_t) = {} & r_r\,T_r(\mathbf{x}_{t+1}|\mathbf{x}_t) + r_i\,T_i(\mathbf{x}_{t+1}|\mathbf{x}_t) \\ & + r_d\,T_d(\mathbf{x}_{t+1}|\mathbf{x}_t), \end{aligned} \tag{8}$$

where T defines the overall transition distribution, r_r, r_i and r_d are transition ratios, $T_r(\cdot)$, $T_i(\cdot)$ and $T_d(\cdot)$ are transition likelihoods of $Replace$, $Insert$ and $Delete$, respectively. The transition likelihoods can be calculated as:

$$T_r(\mathbf{x}'|\mathbf{x}) = \mathbb{1}(w_c \in \mathcal{W}) \cdot \frac{P(Replace(w^*, \mathbf{x}, w_c))}{\sum_{w \in \mathcal{W}} P(Replace(w^*, \mathbf{x}, w))}$$

$$T_i(\mathbf{x}'|\mathbf{x}) = \mathbb{1}(w_c \in \mathcal{W}) \cdot \frac{P(Insert(w^*, \mathbf{x}, w_c))}{\sum_{w \in \mathcal{W}} P(Insert(w^*, \mathbf{x}, w))}$$

$$T_d(\mathbf{x}'|\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x}' = Delete(w^*, \mathbf{x}) \\ 0, & \text{otherwise} \end{cases}$$

where $P(Replace(w^*, \mathbf{x}, w_c))$ and $P(Insert(w^*, \mathbf{x}, w_c))$ are the probabilities of the sentence $(w_0, w_1, ..., w_c, ..., w_{N-1})$ and the sentence $(w_0, ...w^*, w_c, ..., w_{N-1})$ given by a pre-trained language model *e.g.*, RoBERTa, respectively.

Hybrid Search The intuition behind HS stems from the balance between exploration and exploitation. Studies have shown that with a proper balance between exploration and exploitation, the optimization performance can be substantially improved (Črepinšek et al., 2013; Wilson et al., 2021). Specifically, at each step, we randomly select between BS and MHS.

$$HS = \begin{cases} BS & rand < i/n \\ MHS & rand \geqslant i/n \end{cases}, \tag{9}$$

where i is the current step number, n is the maximum number of search steps, and $rand$ is uniformly sampled from $[0, r]$, $r \in [0, 1]$ controls the exploitation-exploration trade-off. It can be seen that with the increase of i, HS gradually changes from exploration (MHS) to exploitation (BS). After $r * n$ step, only BS is used.

3.3 Adversarial Training

Following Goodfellow et al. (2015); Madry et al. (2017), we use adversarial training to resist attacks. More specifically, we defend against attacks and increase model robustness by retraining the model with a mix of adversarial examples and the original data. We choose those adversarial examples where the two reference models $\mathcal{M}_A$ and $\mathcal{M}_B$ have the same predictions and take the same predictions as the target outputs.

4 Experiments

4.1 Implementation Details

The PLM used to select candidate word set is RoBERTa (Liu et al., 2019). We search hyper-parameters on 300 sentences randomly sampled from the PTB development dataset. The criterion used to select all the hyper-parameters is the token level attacking success rate.

4.2 Evaluation

Automatic Evaluation Following Han et al. (2020), we evaluate the adversarial examples on two aspects: generation quality (including fluency and meaning preservation) and attacking efficiency. Specifically, we treat outputs from reference model $\mathcal{A}$, reference model $\mathcal{B}$, or the agreement part of models $\mathcal{A}\&\mathcal{B}$ as ground truths and evaluate the following two attacking success rates (ASRs):

- **Token-level ASR:** the percentage of words in the adversarial examples that are assigned the wrong head without considering the dependence type in dependency parsing or the wrong tag in POS tagging.
- **Sentence-level ASR:** the percentage of mispredicted examples in the adversarial examples.

Human Evaluation We conduct human evaluation of generation quality (fluency, meaning preservation) and attacking efficiency (token and sentence level). We hire three volunteers with linguistic background to label 50 data (sampled from the PTB test set). For generation quality, we ask the annotators to rate from 1 to 5, the higher, the better. For attacking efficiency, we ask them to manually annotate erroneous outputs in the same way as in automatic evaluation.

4.3 Attack on Dependency Parsing

We apply our approach to the dependency parsing task. We choose the Deep Biaffine parser (Dozat and Manning, 2017) as the victim parser $\mathcal{P}_V$ and two other SOTA dependency parsers as the reference parsers: StackPTR (Ma et al., 2018) as parser $\mathcal{P}_A$ and BiST (Kiperwasser and Goldberg, 2016) as parser $\mathcal{P}_B$.

The three parsers are all trained with Penn Treebank 3.0 (PTB, Marcus et al. (1994)) dataset following the same hyper-parameters reported in their papers.

	Token-level ASR↑		
	Parser A	*Parser B*	*Parser A & B*
MHS	18.7	26.4	10.0
BS	33.6	35.4	31.7
HS	**36.6**	**38.7**	**32.8**

Table 2: Experimental results of three modes: BS, MHS, and HS on PTB development set based on automatic evaluation in the dependency parsing task. ASR: Attacking Success Rate.

4.3.1 Main Results

Adversarial Attacks To assess our three modes, we random sample 300 samples from the PTB development set. Results are shown in Table 2. Observation shows that HS, taking advantage of both MHS and BS, performs better in attacking effectiveness than single MHS or BS. Therefore, we adopt HS in the following experiments.

Then we summarize automatic evaluation results in Table 3 and human evaluation results in Table 4. Human evaluation is consistent with automatic evaluation: our proposed method significantly outperforms the baseline model at almost all metrics. Particularly, our approach in the "Ours-HS" row demonstrates HS's advantage on the attacking success rate. The attacker from Zheng et al. (2020) uses the black-box setting to attack the same word-based Biaffine model and 15% words are allowed to be modified. Their method keeps better sentence quality, at the cost of a low ASR. Compared with it, our approach is more than twice better than theirs in ASR and maintains a comparable generation quality. In contrast of the attacker from Zheng et al. (2020) that can only use substitution with the same part of speech tag, our attacker allows more flexible manipulations. A case study is shown in Figure 2, our approach replace `But` and `kept` with `And` and `are` respectively, and add `then`. These manipulations lead to a successful attack. Compared with Han et al. (2020), our HS is much more effective in terms of ASR and GQ.

Defense Against Adversarial Attack Attacking the PTB training set, our HS approach can generate about 8000 adversarial examples satisfying $\mathcal{P}_V(\mathbf{x}) \neq \mathcal{P}_A(\mathbf{x}) = \mathcal{P}_B(\mathbf{x})$. The mixed dataset of adversarial examples and the original training data is used to retrain the victim parser. After adversarial training, the unlabeled attachment score (UAS) of the victim model increases from 95.37 to 95.53. To investigate the significance of the improvement, we perform significance tests on the UAS score.

	GQ		Token-level ASR↑			Sentence-level ASR↑		
	Fluency↓	*MP*↑	*Parser A*	*Parser B*	*Parsers A&B*	*Parser A*	*Parser B*	*Parsers A&B*
Original	**156.02**	**1**	3.9	9.1	1.7	36.7	59.3	21.0
Zheng et al. (2020)	281.99	0.94	7.5	14.5	2.8	52.2	71.4	30.2
Han et al. (2020)	174.16	0.80	14.5	19.9	5.6	78.4	87.7	53.8
Ours-HS	157.41	0.92	**35.3**	**37.8**	**32.4**	**98.7**	**98.8**	**100.0**

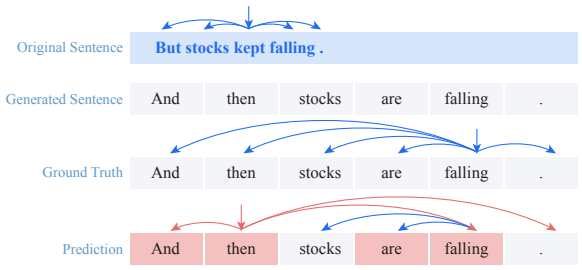
Table 3: Experimental results on PTB test set based on automatic evaluation in dependency parsing task. "Original" shows the results of original sentences in the PTB test set. We use the perplexity of GPT-2 (Radford et al., 2019) to evaluate the sentence fluency (*Fluency*), lower is better. We use the BERTSCORE (Zhang et al., 2019b) to evaluate the meaning preservation (*MP*). Higher is better. GQ: Generation Quality.

	GQ↑		ASR↑	
	Fluency	*MP*	*Tl.*	*Sl.*
Zheng et al. (2020)	**4.57**	**4.07**	15.3	42
Han et al. (2020)	4.07	3.16	23.8	70
Ours-HS	4.21	3.65	**38.4**	**93**

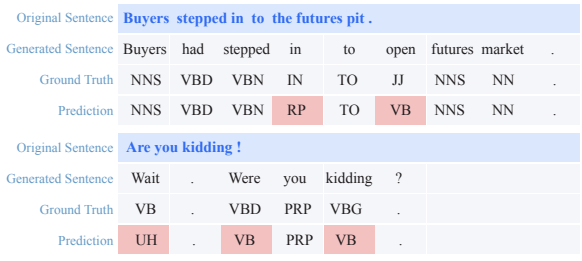
Table 4: Experimental results on PTB test set based on human evaluation. TL: Token-level. SL: Sentence-level.

	Token-level ASR↓		
	Parser A	*Parser B*	*Parser A&B*
Original	36.6	38.7	32.8
Retrain	**32.3**	**35.0**	**28.0**

Table 5: Token level attacking success rates after adversarial training.



(a) Dependency Parsing



(b) POS Tagging

Figure 2: Adversarial examples of dependency parsing (a) and POS tagging (b) generated by our approach. Red color indicates misprediction.

We calculate the p-value using the one-tailed sign test with the bootstrap re-sampling from the PTB test set following Chollampatt et al. (2019). We compare the retrained model with the original victim model. The p-values is $1.61e$-5 that shows the significance. To test the adversarial robustness, we use our HS approach to attack the retrained model on 300 randomly sampled data from the PTB development set. As shown in Table 5, adversarial training significantly reduces the token level attacking success rate on all three settings.

4.4 Attack on POS Tagging

We apply our approach to the POS tagging task. We use the tagger from Ma and Hovy (2016) as our victim tagger $\mathcal{T}_V$, and we choose two state-of-the-art taggers: Stanford POS tagger (Toutanova et al., 2003) and Senna tagger (Collobert et al., 2011) as our reference tagger $\mathcal{T}_A$ and tagger $\mathcal{T}_B$, respectively. We conduct experiments on the PTB dataset. All the hyper-parameters of the three taggers are the same as reported in their papers.

4.4.1 Main Results

Adversarial Attacks As in the experiments of the dependency parsing task, we first randomly sample 300 samples from the PTB development set to compare our three search methods. Results are shown in Table 7. We can find that the HS still performs the best in the attacking success rate. But it performs a relative more minor advantage in this task than its performance in the dependency parsing task. One possible reason is that, compared to dependency parsing, POS tagging is a simpler task, so BS is effective while MHS (*exploration* with more randomness) can not bring more benefit. We still adopt HS in the following experiments as the experiments on dependency parsing.

We show the automatic evaluation results on the test set in Table 6 and human evaluation results on the sampled test set in Table 8. Fluency, meaning preservation and attacking success rate of our approach are all above Han et al. (2020). Our approach shows its high efficiency even though in the relatively simple task. Particularly, our approach improves the token level attacking success rate by 8.4%, and on the sentence level, our ap-

	GQ		Token-level ASR↑			Sentence-level ASR↑		
	Fluency↓	*MP*↑	*Tagger A*	*Tagger B*	*Tagger A&B*	*Tagger A*	*Tagger B*	*Taggers A&B*
Original	156.02	**1**	2.7	3.4	0.7	38.8	45.8	12.9
Han et al. (2020)	142.59	0.88	**13.8**	9.5	2.3	76.8	74.6	23.2
Ours-HS	**136.16**	0.93	12.8	**14.2**	**10.7**	**96.3**	**97.0**	**93.4**

Table 6: Experimental results on PTB test set in the POS tagging task based on automatic evaluation.

	Token-level ASR↑		
	Tagger A	*Tagger B*	*Taggers A & B*
MHS	5.8	11.2	2.9
BS	14.8	15.9	12.8
HS	**15.1**	**16.3**	**13.1**

Table 7: Experimental results of three modes in the POS tagging task: BS, MHS, and HS on PTB dev set based on automatic evaluation.

	GQ↑		ASR↑	
	Fluency	*MP*	*Tl.*	*Sl.*
Han et al. (2020)	3.88	3.16	8.10	52.0
Ours-HS	**4.20**	**3.59**	**10.43**	**88.67**

Table 8: Experimental results on PTB test set in the POS tagging task based on human evaluation. TL: Token-level. SL: Sentence-level.

proach shows its powerful attack capability, achieving more than 70% improvement compared to previous baseline, even though the approach of Han et al. (2020) involves training a generating model and is much more computationally expensive than ours. Case studies are shown in Figure 2(b).

Defense Against Adversarial Attack We also conduct adversarial training in the POS tagging task. To compared with Han et al. (2020), we sample 1000 additional adversarial sentences generated by attacking the PTB training set using our HS approach. We mixed these sentences with the initial training set to retrain the victim tagger. As a result, the accuracy of the tagger improves 0.21 from 97.55 to 97.76 on the PTB test set, while Han et al. (2020) reports a 0.13 improvement in the same setting. That demonstrates the high quality and effectiveness of our generated adversarial sentences.

4.5 Analysis

We conduct our analytical experiments in the dependency parsing task.

Impact of Metric for Reference Models Due to the sensitivity to small perturbations as illustrated in Table 1, the structured-prediction attackers require an automated yet unbiased evaluation scheme that is suitable for assessing the prediction of the adversarial examples. For this purpose, we adopt the agreement of two pre-trained parsers. The criterion

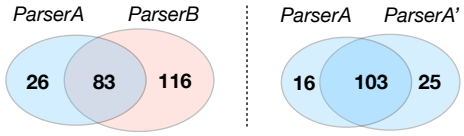


Figure 3: Illustration of *OIoU*. *Parser A* and *Parser A'* denote the same architectures trained with different random seeds. The numbers in the set denote the mispredicted token numbers.

for selecting these reference parsers is that they should be diverse besides having a high parsing accuracy (Han et al., 2020). Therefore, we propose the metric *Opposite Intersection over Union (OIoU)* to evaluate the diversity degree of reference parsers. In particular, we propose *OIoU* as one minus the number of common erroneous predictions from two reference parsers divided by the total number of unique erroneous predictions made by two reference parsers. A quantitative example is demonstrated in Figure 3. The diverse degrees of the parsers on the left and right are 0.631 $(1 - 83/(26 + 83 + 116))$ and 0.285 respectively. Our experiments find that the parsers on the left with a higher *OIoU* indeed result in better attacking efficiency than those on the right (12.9% vs. 5.2% token-level ASR). Note that, for fair comparison of SHARP with prior work, we reuse the same reference parsers for all the experiments.

Generalizability We exchange the victim parser and reference parser to show the generalization of our black-box adversarial attack methods. Specifically, we take the Deep Biaffine parser (Dozat and Manning, 2017) as the reference parser $\mathcal{P}_A$ and the StackPTR (Ma et al., 2018) as the victim parser $\mathcal{P}_V$. We repeat the experiments of dependency parsing keeping the same setup of Table 3 except for the parser choice. Experimental results based on automatic evaluation on the test set are shown in Table 9. We can find that our approach still can keep high attacking success rate both on token level and sentence level. Moreover, the fluency of the generated adversarial sentences of Han et al. (2020) becomes worse after changing victim parser, but our approach well maintains the quality of the sentences.

	GQ		Token-level ASR↑			Sentence-level ASR↑		
	Fluency↓	*MP*↑	*Parser A*	*Parser B*	*Parsers A&B*	*Parser A*	*Parser B*	*Parsers A&B*
Original	**156.02**	**1.00**	3.7	8.5	1.5	36.7	57.2	9.2
Han et al. (2020)	244.69	0.92	19.6	23.3	13.4	70.8	77.2	24.3
Ours-HS	142.78	0.94	**33.6**	**36.8**	**30.8**	**98.1**	**98.2**	**92.4**

Table 9: Experimental results on PTB test set in the dependency parsing task based on automatic evaluation with StackPTR as the victim model while the Deep Biaffine parser and BiST as reference models.

	GQ		Token-level ASR↑			Sentence-level ASR↑		
	Fluency↓	*MP*↑	*Parser A*	*Parser B*	*Parsers A&B*	*Parser A*	*Parser B*	*Parsers A&B*
Ours-HS	188.01	0.92	35.8	39.0	32.8	98.6	98.6	95.6
-$s(\mathbf{x}, \mathbf{x}^{ori})$	220.35	0.91	36.2	38.9	33.0	99.3	99.7	97.7
-$f(\mathbf{x})$	672.54	0.92	44.9	46.9	41.2	99.7	100	100
-$r(\mathbf{x})$	67.00	0.93	6.72	13.75	3.0	56.8	77.1	24.4
random	919.85	0.88	16.3	25.2	7.9	90.4	93.8	45.5

Table 10: Ablation study on 300 samples of PTB development set in the dependency parsing task based on automatic evaluation.

Ablation Study We show the impact of the three different scores in our objective function. Table 10 shows the automatic evaluation results of generation quality and attacking success rate on 300 samples randomly sampled from the development set. It can be seen that without considering the fluency of generated sentences (Row $-f(\mathbf{x})$ in Table 10), it is natural that the attacking success rate can be further increased, but the generation quality becomes worse. Without considering meaning reservation between original sentences and generated sentences (Row $-s(\mathbf{x}, \mathbf{x}^{ori})$), we can find that MP drops marginally. To verify the importance of meaning preservation (Row $s(\mathbf{x}, \mathbf{x}^{ori})$), we experiment in the same setting as Section 4.3.1 except for using the adversarial examples generated without considering meaning preservation. We find that the unlabeled attachment score (UAS) of the victim model is 94.57 (vs 95.53), which shows that the quality of generated sentences is important to improve the victim model in terms of performance and robustness after retraining on adversarial examples. The importance is also demonstrated by prior work of Wang et al. (2021). Without optimizing $r(\mathbf{x})$, we can see that the attack success rate is even lower than random sampling because it needs to promise the quality of sampled sentences. A big gap on both generated quality and attacking success rate between random sampling and our HS approach demonstrates that the strength of our methods.

Impact of Candidate Size and Manipulation Count Figure 4 and 5 show the result on different candidate sizes and manipulation counts, respectively. We can find that, with the increase of the candidate size, the attacking success rate also increases, but the growth rate gradually slows down. The manipulation count shows a similar trend.

In our setting, the computing cost has a linear relationship with these two hyper-parameters. Thus we have a trade-off between time and performance: in our experiments, we set the candidate size and manipulation count to 5 and 50, respectively.

Impact of Beam Size We investigate the impact of the beam size under the same computing resources. For example, when the beam size is 1, the candidate number is 10; and when the beam size is 2, the candidate number is 5. The manipulation count keeps the same. The results are shown on Figure 6. It can be seen that 2 is the best.

Causal Analysis of Adversarial Attacks Since SHARP searches the whole sentence space for adversarial attacks without pre-defined templates, the generated adversarial examples have the potential for unseen discoveries. Therefore, we analyze the adversarial examples and conclude some new templates that cause mispredictions of the victim model. We list two observations here. First, uncommon words (e.g., replacing the place name with an adverb or adding a surname) often cause a misprediction. For example, the adversarial example "What is Santa actually worth?" (the original sentence is "What is Santa Fe worth?") makes the victim model mispredict the head of "Santa" being "actually". A probable reason is that interrogative sentences with uncommon words are rare in the training set and the victom model is confused with these out-vocabulary words. The other observation is that symbols such as "%" and "." of-

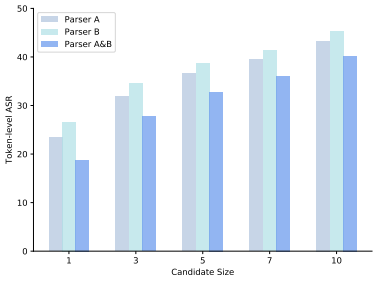


Figure 4: Token-level attacking success rate on different candidate sizes.

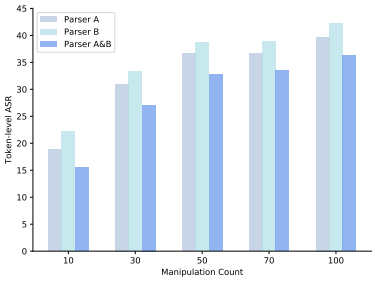


Figure 5: Token-level attacking success rate on different manipulation counts.

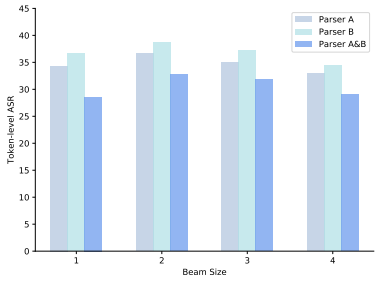


Figure 6: Token-level attacking success rate on different beam size.

ten mislead the model predictions. Surprisingly, the victim model can predict the correct structure for the sentence "It is widely expected that they will." but fails when the period in this sentence is dropped. We give a simple quantitative analysis of these adversarial types. For the first type (adding a surname to an interrogative sentence), we randomly choose 100 names and write the sentence "What does $Name\ Surname$ like?". We get 100% sentence-level attack success. For the second type (adding an adverb after "%"), we randomly select 100 sentences which contain "%" from the original dataset, and then add adverb "globally" after the "%" to generate new sentences as adversarial samples. We get about 50% sentence-level attack success in this case.

5 Related Work

There is limited literature available for adversarial attacking on structured prediction tasks. Previous adversarial training has been conducted on NLP tasks such as text classification (Liang et al., 2018; Alzantot et al., 2018), machine translation (Zhao et al., 2018; Ebrahimi et al., 2018; Cheng et al., 2020a) and dialogue systems (Cheng et al., 2019a). Recently, adversarial training has also been explored on structured prediction tasks, such as dependency parsing (Zheng et al., 2020; Han et al., 2020). Zheng et al. (2020) replaced some words with adversarially chosen counterparts with the same part of speech tags. They target specific syntactic adversarial sentence examples to attack dependency parser. Han et al. (2020) investigated generation-based attackers for structured prediction tasks.

6 Conclusion

In this paper, we quantitatively investigate the sensitivity of structured prediction tasks and formulate the black-box adversarial attack as a search problem that seeks to maximize a specially designed objective function. Both automatic and human evaluations show that our approach beats the previous approaches by a large margin in attacking victim models and simultaneously guarantees better fluency and meaning preservation. Our defense experiments show that the adversarial samples generated by our approach can be used to improve the original model's robustness and performance.

Acknowledgements

Liwen Zhang, Zixia Jia and Kewei Tu were supported by the National Natural Science Foundation of China (61976139). Wenjuan Han and Zilong Zheng were supported by the National Key R&D Program of China (2021ZD0150200).

References

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, Brussels, Belgium. Association for Computational Linguistics.

Minhao Cheng, Wei Wei, and Cho-Jui Hsieh. 2019a. Evaluating and enhancing the robustness of dialogue systems: A case study on a negotiation agent. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3325–3335.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. 2020a. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3601–3608.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019b. Robust neural machine translation with doubly adversarial inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4324–4333, Florence, Italy. Association for Computational Linguistics.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020b. AdvAug: Robust adversarial augmentation for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5961–5970, Online. Association for Computational Linguistics.

Siddhartha Chib and Edward Greenberg. 1995. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335.

Shamil Chollampatt, Weiqi Wang, and Hwee Tou Ng. 2019. Cross-sentence grammatical error correction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 435–445, Florence, Italy. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR)*, 45(3):1–33.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. *5th International Conference on Learning Representations (ICLR)*.

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, Melbourne, Australia. Association for Computational Linguistics.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *International Conference on Learning Representations (ICLR)*.

Wenjuan Han, Liwen Zhang, Yong Jiang, and Kewei Tu. 2020. Adversarial attack and defense of structured prediction models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2327–2338, Online. Association for Computational Linguistics.

W Keith Hastings. 1970. Monte carlo sampling methods using markov chains and their applications.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649.

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.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 4208–4215. AAAI Press.

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*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1403–1414.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Bo Pang, Erik Nijkamp, Wenjuan Han, Linqi Zhou, Yixian Liu, and Kewei Tu. 2020. Towards holistic and automatic evaluation of open-domain dialogue

generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3619–3629.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

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.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Dilin Wang, Chengyue Gong, and Qiang Liu. 2019. Improving neural language modeling via adversarial training. In *International Conference on Machine Learning*, pages 6555–6565.

Yuxuan Wang, Wanxiang Che, Ivan Titov, Shay B Cohen, Zhilin Lei, and Ting Liu. 2021. A closer look into the robustness of neural dependency parsers using better adversarial examples. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2344–2354.

Robert C Wilson, Elizabeth Bonawitz, Vincent D Costa, and R Becket Ebitz. 2021. Balancing exploration and exploitation with information and randomization. *Current Opinion in Behavioral Sciences*, 38:49–56.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Dp-gan: diversity-promoting generative adversarial network for generating informative and diversified text. *arXiv preprint arXiv:1802.01345*.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019a. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019b. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *International Conference on Learning Representations*.

Xiaoqing Zheng, Jiehang Zeng, Yi Zhou, Cho-Jui Hsieh, Minhao Cheng, and Xuanjing Huang. 2020. Evaluating and enhancing the robustness of neural network-based dependency parsing models with adversarial examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6600–6610, Online. Association for Computational Linguistics.

A Technical Details

A.1 Hyper-Params

The shared hyper-parameters of Beam Search (BS), Metropolis-Hastings Sampling (MHS), and Hybrid Search (HS) are listed in Table 11.

Beam Size	2
Size of Candidate Set	5
Manipulation Count	50
p_r	0.5
p_i	0.25
p_d	0.25

Table 11: Hyper-parameter setting.

A.2 Evaluation

In Table 2 and Table 5 of the main body, we use different evaluation metrics to measure results of *Parsers A&B* compared with Han et al. (2020). Take **Token-level ASR** as an example, they calculate *Parser A&B* using $1 - s/n$, where s is the number of tokens of which the three parsers (A, B and C) have the same prediction (*i.e.*, $\mathcal{M}_A(x) = \mathcal{M}_B(x) = \mathcal{M}_C(x)$), and n is the total number of tokens. We use a/b, where a is the number of tokens that $\mathcal{M}_A(x) \neq (\mathcal{M}_B(x) = \mathcal{M}_C(x))$, and b is the number of tokens that $\mathcal{M}_B(x) = \mathcal{M}_C(x)$. Since the assumption behind identifying the ground truth of the structured outputs is that two external reference parsers have the same prediction, it is more reasonable to use a/b when calculating the attack success rate to identify if the adversarial sentence can indeed fool the victim model. We re-calculated all the *Parsers A&B* results of Han et al. (2020) in the main body of this work. For reference, we give results of our model calculated according to their formula as Table 12. It can be seen that our approach outperforms the approach proposed by Han et al. (2020) even with the original metrics.

	Token-level ASR↑		
	Parser A	*Parser B*	*Parser A & B*
Han	13.9	19.2	24.1
our-HS	**36.6**	**38.7**	**37.6**

Table 12: Experimental results on PTB test dataset in dependency parsing task.

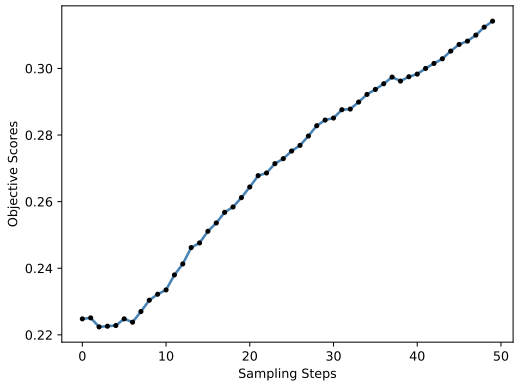


Figure 7: Objective score curve on increasing sampling steps.

B Optimizing Process

We show our optimizing curve in the Figure 7. We can see that the objective score is oscillating in the early stages of sampling, because MHS sampling is chosen with high probability in the early stages, which shows the *exploration* process. After the early stage, we see a clear upward trend of the objective score, where our approach chooses BS with high probability, and BS directly searches the local optimal.