A Reinforced Generation of Adversarial Examples for Neural Machine Translation

Wei Zou[1] **Shujian Huang**[1] **Jun Xie**[2] **Xinyu Dai**[1] **Jiajun Chen**[1]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
[2]Tencent Technology Co, China
zouw@smail.nju.edu.cn, {huangsj,daixinyu,chenjj}@nju.edu.cn
stiffxie@tencent.com

Abstract

Neural machine translation systems tend to fail on less decent inputs despite its significant efficacy, which may significantly harm the credibility of these systems—fathoming how and when neural-based systems fail in such cases is critical for industrial maintenance. Instead of collecting and analyzing bad cases using limited handcrafted error features, here we investigate this issue by generating adversarial examples via a new paradigm based on reinforcement learning. Our paradigm could expose pitfalls for a given performance metric, e.g., BLEU, and could target any given neural machine translation architecture. We conduct experiments of adversarial attacks on two mainstream neural machine translation architectures, RNN-search, and Transformer. The results show that our method efficiently produces stable attacks with meaning-preserving adversarial examples. We also present a qualitative and quantitative analysis for the preference pattern of the attack, demonstrating its capability of pitfall exposure.

1 Introduction

Neural machine translation (NMT) based on the encoder-decoder framework, such as RNN-Search (Bahdanau et al., 2014; Luong et al., 2015, RNNSearch) or Transformer (Vaswani et al., 2017, Transformer), has achieved remarkable progress and become a de-facto in various machine translation applications. However, there are still pitfalls for a well-trained neural translation system, especially when applied to less decent real-world inputs compared to training data (Belinkov and Bisk, 2017). For example, typos may severely deteriorate system outputs (Table 1). Moreover, recent studies show that a neural machine translation system can also be broken by noisy synthetic inputs (Belinkov and Bisk, 2017; Lee et al., 2018). Due to the black-box nature of a neural system, it has been a challenge to fathom when and how the system tends to fail.

in	耶路撒冷发生自杀**爆炸**事件
out	suicide bombing in jerusalem
in	耶路撒冷发生自杀**爆**事件
out	*eastern* jerusalem *explores a case of eastern europe*

Table 1: Fragility of neural machine translation. A typo leaving out a Chinese character "炸" leads to significant errors (noted by italics) in English translation. Both "爆" and "爆炸" mean "bombing" in English.

Intuitively, researchers seek to apprehend such failures by the analysis of handcrafted error indicating features (Zhao et al., 2018; Karpukhin et al., 2019). This strategy is costly because it requires expert knowledge for both linguistics and the target neural architecture. Such features are also less applicable because some common errors in deep learning systems are hard to formulate, or very specific to certain architectures.

Instead of designing error features, recent researchers adopt ideas from adversarial learning (Goodfellow et al., 2014) to generate adversarial examples for mining pitfalls of NLP systems (Cheng et al., 2018a; Ebrahimi et al., 2018; Zhao et al., 2017). Adversarial examples are minor perturbed inputs that keep the semantic meaning, yet yield degraded outputs. The generation of valid adversarial examples provides tools for error analysis that is interpretable for ordinary users, which can contribute to system maintenance. Though it has achieved success concerning continuous input, e.g., images, there are following major issues for NLP tasks.

First, it is non-trivial to generate valid discrete tokens for natural language, e.g., words or characters. Cheng et al. (2018a) follow Goodfellow et al. (2014) to learn noised representation then sample tokens accordingly. However, there is no guaranteed correspondence between arbitrary representation and valid tokens. Therefore, it may gen-

in	Two man are playing on the street corner.
perturbed in	Two man are playing ***frisbee in the park***.
out	Zwei Männer spielen an einer Straßenecke.
perturbed out	Zwei Männer spielen frisbee im park.

Table 2: Example of undesirable perturbation in adversarial examples for machine translation in (Zhao et al., 2017), though it yields very different output compare to the origin, it does not indicate system malfunction.

erate tokens departing from learned representation, which undermines the generation. Ebrahimi et al. (2018) turns to a search paradigm by a brute-force search for direct perturbations on the token level. To lead the search, a gradient-based *surrogate* loss must be designed upon every token modification by given target annotations. However, this paradigm is inefficient due to the formidable computation for gradients. Furthermore, surrogate losses defined upon each token by targets requires high-quality targets, and risks being invalidated by any perturbation that changes tokenization.

Another issue is to keep the semantics of original inputs. Different from the fact that minor noises on images do not change the semantics, sampling discrete tokens from arbitrary perturbed representation (Cheng et al., 2018a) may generate tokens with different semantics and lead to ill-perturbed samples (Table 2). Searching for the perturbed input also requires a semantic constraint of the search space, for which handcrafted constraints are employed (Ebrahimi et al., 2018). Though constraints can also be introduced by multitask modeling with additional annotations (Zhao et al., 2017), this is still not sufficient for tasks requiring strict semantic equivalence, such as machine translation.

In this paper, we adopt a novel paradigm that generates more reasonable tokens and secures semantic constraints as much as possible. We summarize our contributions as the following:

- We introduce a reinforcement learning (Sutton and Barto, 2018, RL) paradigm with a discriminator as the terminal signal in its environment to further constrain semantics. This paradigm learns to apply discrete perturbations on the token level, aiming for direct translation metric degradation. Experiments show that our approach not only achieves semantically constrained adversarial examples but also leads to effective attacks for machine translation.

- Our paradigm can achieve the adversarial example generation with outclassed efficiency by only given source data. Since our method is model-agnostic and free of handcrafted error feature targeting architectures, it is also viable among different machine translation models.

- We also present some analysis upon the state-of-the-art Transformer based on its attack, showing our method's competence in system pitfall exposure.

2 Preliminaries

2.1 Neural Machine Translation

The most popular architectures for neural machine translation are RNN-search (Bahdanau et al., 2014) and Transformer (Vaswani et al., 2017). They share the paradigm to learn the conditional probability $P(Y|X)$ of a target translation $Y = [y_1, y_2, ..., y_m]$ given a source input $X = [x_1, x_2, ..., x_n]$. A typical NMT architecture consists of an encoder, a decoder and attention networks. The encoder encodes the source embedding $X_{emb} = [emb_1, emb_2, ...emb_n]$ into hidden representation $H = [h_1, h_2, ..., h_n]$. Then a decoder f_{dec} with attention network attentively accesses H for an auto-regressive generation of each y_i until the end of sequence symbol (EOS) is generated:

$$P(y_i|y_{<i}, X) = \text{softmax}(f_{dec}(y_{i-1}, s_t, c_t; \theta_{dec})) \quad (1)$$

where c_t is the attentive result for current decoder state s_t given H.

2.2 Actor-Critic for Reinforcement Learning

Reinforcement learning (Sutton and Barto, 2018, RL) is a widely used machine learning technique following the paradigm of *explore and exploit*, which is apt for unsupervised policy learning in many challenging tasks (e.g., games (Mnih et al., 2015)). It is also used for direct optimization for non-differentiable learning objectives (Wu et al., 2018; Bahdanau et al., 2016) in NLP.

Actor-critic (Konda and Tsitsiklis, 2000) is one of the most popular RL architectures where the agent consists of a separate policy and value networks called actor and critic. They both take in environment state s_t at each time step as input, while

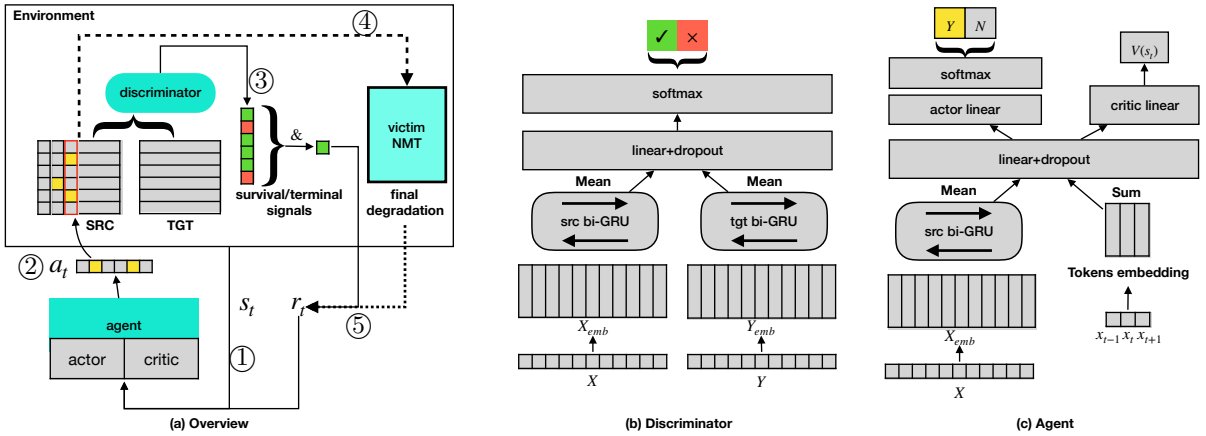

Figure 1: [a] Overview of our RL architecture. ① Environment states are processed as inputs for agent; ② agent yield modification upon SRC in environment; ③ determine survival and step reward of environment; ④ determine degradation with victim NMT as episodic reward; ⑤ update agent with total rewards. During a training episode, we loop ① to ③ and accumulate step rewards until environment terminates. Dash line indicates execution at the end of an episode. [b] Architecture of discriminator. [c] Architecture of agent.

actor determines an action a_t among possible action set $\mathcal{A}$ and critic yields value estimation $V_t(s_t)$. In general, the agent is trained to maximize discounted rewards $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ for each state, where $\gamma \in (0, 1]$ is the discount factor. Such goal can be further derived as individual losses applied to actor and critic. Thus the actor policy loss L^π on step t is:

$$L_t^\pi(\theta_\pi) = \log P(a_t|s_t) A_t(s_t, a_t); a_t \in \mathcal{A} \quad (2)$$

where θ_π denotes actor parameters, $A_t(s_t, a_t)$ denotes general advantage function (Schulman et al., 2015) on state s_t for action a_t given by $\sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}) - V(s_t)$, which can be further derived as:

$$\begin{aligned} A_t(s_t, a_t) = \gamma A_{t+1}(s_{t+1}, a_{t+1}) + r_t \\ + \gamma V_{t+1}(s_{t+1}) - V_t(s_t) \end{aligned} \quad (3)$$

Meanwhile, critic learns to estimate R_t via minimizing a temporal difference loss L^v on each step t:

$$L_t^v(\theta_v) = \frac{1}{2}(r_t + \gamma R_{t+1} - V_t(s_t))^2 \quad (4)$$

where θ_v denotes critic parameter.

Usually, the training is regularized by maximizing policy entropy H^π to avoid exploration failure before exploiting optimum policy (Ziebart, 2010). Thus the total loss becomes:

$$L(\theta) = \sum_t (\alpha L_t^v - L_t^\pi - \beta H^\pi(\cdot|s_t)) \quad (5)$$

where α and β are hyperparameters for value loss and entropy coefficients.

2.3 adversarial examples in NLP

A general adversarial example generation can be described as the learning process to find a perturbation δ on input X that maximize system degradation L_{adv} within a certain constraint $C(\delta)$:

$$\underset{\delta}{\text{argmax}} L_{adv}(X + \delta) - \lambda C(\delta) \quad (6)$$

where λ denotes the constraint coefficient, L_{adv} is determined by the goal of the attack. However, currently effective adversarial generation for NLP is to search by maximizing a surrogate gradient-based loss:

$$\underset{1 \le i \le n, x' \in vocab}{\text{argmax}} \quad L_{adv}(x_0, x_1, ...x_i'...x_n) \quad (7)$$

where L_{adv} is a differentiable function indicating the adversarial object. Due to its formidable search space, this paradigm simply perturbs on a small ratio of token positions and greedy search by brute force among candidates. Note that adversarial example generation is fundamentally different from noised hidden representation in adversarial training (Cheng et al., 2019; Sano et al., 2019), which is not to be concerned in this work.

3 Approach

In this section, we will describe our reinforced learning and generation of adversarial examples (Figure 1) in detail. Overall, the victim model is a part of the **environment** (denoted as Env), which yields rewards indicating overall degradation based on modified inputs. A reinforced **agent**

learns to modify every source position from left to right sequentially. Meanwhile, a **discriminator** in Env provides every-step survival signals by determining whether SRC is ill-perturbed.

3.1 Environment

We encapsulate the victim translation model with a discriminative reward process as an Env for a reinforced agent to interact.

3.1.1 Environment State

The state of the Env is described as $s_t = (SRC, t)$, where $SRC = [src_0, src1, ..., src_N]$ are N sequences processed by victim model's vocabulary and tokenization. Each sequence $src_i = [x_1, x_2, ..., x_n]$ is concatenated with BOS, EOS, which indicate the begin and end of the sequence, then padded to same length. Time step $t \in [1, n]$ also indicates the token position to be perturbed by the agent. Env will consecutively loop for all token positions and update s_t based on the agent's modification. Env also yields reward signals until the end or intermediately terminated. That is, *all* sequences in SRC are determined by D as ill-perturbed during the reward process. Once the Env terminates, it finishes the current episode and reset its state with a new batch of sequences as SRC.

3.1.2 Reward Process with Discriminator

The reward process is *only* used during training. It consists of a survival reward r_s on every step and a final degradation r_d concerning an overall metric if the agent survives till the end. Overall, we have:

$$r_t = \begin{cases} -1, \text{ terminated} \\ \frac{1}{N}\sum_N a \cdot r_\text{s}, \text{ survive } \& \ t \in [1, n) \\ \frac{1}{N}\sum_N (a \cdot r_\text{s} + b \cdot r_\text{d}), \text{ survive } \& \ t = n \end{cases} \tag{8}$$

where a, b are hyper parameters that keeps the overall r_s and r_d within similar magnitude.

Instead of direct optimization of the constrained adversarial loss in Eq.6, we model discriminator D's output as survival rewards similar to that in gaming (Mnih et al., 2015). That is, the agent must survive for its goal by also fooling D, which attempts to terminate ill-perturbed modifications. We define an ill-perturbed source by determining whether it still matches the original target tgt.

Discriminator As it is shown in Figure 1(b), discriminator D consists of bi-directional GRU encoders for both source and target sequence. Their corresponding representation is averaged and concatenated before passed to a feedforward layer with dropout. Finally, the output distribution is calculated by a softmax layer. Once D determines the pair as positive, its corresponding possibility is regarded as the reward, otherwise 0:

$$r_\text{s} = \begin{cases} P(\text{positive}|(src', tgt); \theta_d), & \text{positive} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

As long as the environment survives, it yields averaged reward among samples from SRC (Eq.8) to mitigate rewards' fluctuation that destabilize training.

Discriminator Training Similar to GAN training, the environment's D must update as the agent updates. During its training, the agent's parameter is freezed to provide training samples. For every D's training epoch, we randomly choose half of the batch and perturb its source using the current agent as negative samples. During D's updates, we randomly generate a new batch of pairs from parallel data likewise to test its accuracy. D is updated at most $step_D$ epochs, or until its test accuracy reaches acc_bound.

Env only[1] yields -1 as overall terminal rewards when all sequences in SRC are *intermediately* terminated. For samples classified as negative during survival, their follow-up rewards and actions are masked as 0. If the agent survives until the end, Env yields additional averaged r_d as final rewards for an episode. We follow Michel et al. (2019) to adopt relative degradation:

$$r_\text{d} = \frac{\text{score}(y, refs) - \text{score}(y', refs)}{\text{score}(y, refs)} \tag{10}$$

where y and y' denote original and perturbed output, $refs$ are references, and $score$ is a translation metric. If $score(y, refs)$ is zero, we return zero as r_d. To calculate score we *retokenize* perturbed SRC by victim models vocabulary and tokenizer before translation.

[1]It is commonly accepted that frequent negative rewards result in agents' tendency to regard zero-reward as optimum and fail exploration, which further leads to training failure.

3.2 Agent

As it is shown in Figure 1 (c), the agent's actor and critic share the same input layers and encoder, but later processed by individual feedforward layers and output layers. Actor takes in SRC and current token with its surrounding (x_{t-1}, x_t, x_{t+1}), then yields a binary distribution to determine whether to attack a token on step t, while critic emits a value $V(s_t)$ for every state. Once the actor decides to perturb a specific token, this token will be replaced by another token in its candidate set.

Candidate Set We collect at most K candidates for each token in the victim's vocabulary within a distance of ϵ. ϵ is the averaged Euclidean distance of K-nearest embedding for all tokens in victim vocabulary. We note that there shall always be candidates for a token in test scenarios that are beyond victim's vocabulary, for those without a nearby candidate, we assign UNK as its candidate. Once the agent chooses to replace a token with UNK, we follow Michel et al. (2019) to present a valid token that is also UNK to the victim's vocabulary.

Agent Training The agent is trained by algorithm in appendix A. Since the agent is required to explore with stochastic policy during training, it will first sample based on its actor's output distribution on whether to perturb the current position, then randomly choose among its candidates. The agent and discriminator take turns to update. We assume the training is converged when test accuracy for D does not reach over a certain value within certain continuous learning rounds of agent and discriminator.

Agent Inference To generate adversarial examples, the agent will take in source sequences and perturb on each position based on the actor's output from left to right, then choose the nearest candidate. As the agent's critic learns to estimate expected future rewards for a step, only when it yields positive value will agent perturb, otherwise it indicates an undesirable perturbation; thus, the agent is muted.

4 Experiments

4.1 Data Sets

We test our adversarial example generations on Zh→En, En→Fr, and En→De translation tasks, which provide relatively strong baselines for victim models and mass test samples.

We train our agent using only parallel data that is used for victims' training. we train on LDC Zh→En[2](1.3M pairs), WMT14 En→De[3] (4.5M pairs) and WMT15 En→Fr[4](2.2M pairs) for victim models respectively. For subword level translation, we apply byte pair encoding (Sennrich et al., 2015, BPE) for both source and target languages with the vocabulary size of 37k. We also use join-BPE for En-De and En-Fr experiments with 34k and 33k vocabulary size, respectively. For word-level translation, we use NLPIR-ICTCLAS and Moses tokenizer for Chinese and English tokenization, respectively. We adopt 30k as vocabulary size for both source and target language. We adopt NIST test sets [5] for Zh→En and WMT test sets for En→De and En→Fr, then generate adversarial examples for these sources for analysis.

4.2 Victim Models

We choose the state-of-the-art RNN-search and Transformer as victim translation models. For RNN-search, we train subword level models and strictly follow the architecture in Bahdanau et al. (2014). As for Transformer, we train both word-level and subword-level model for Zh→En and only subword-level models for En→De and En→Fr with the architecture and the base parameter settings by Vaswani et al. (2017). For the above models, we apply the same batch scheme and Adam optimizer following Vaswani et al. (2017). We choose MT03, newsdiscuss2015 and newstest2013 for Zh→En, En→Fr, En→De as validation set respectively.

4.3 Metrics

We first report attack results both in terms of char-level BLEU (*chrBLEU*) of perturbed source by the origin to indicate modification rate, and relative decrease in target BLEU (*RD*):

$$RD = \frac{\text{BLEU}(y, refs) - \text{BLEU}(y', refs)}{(1 - \text{chrBLEU}(x', x)) \times \text{BLEU}(y, refs)} \tag{11}$$

We adopt sacreBLEU (Post, 2018) to test case-insensitive BLEU on detokenized targets.

[2]ldc2002E18, ldc2003E14, ldc2004T08, ldc2005T06
[3]https://nlp.stanford.edu/projects/nmt/
[4]Europarl-v7, news-commentary-v10
[5]MT02,03,04,05,06

	Zh-En MT02-06			
	BLEU	chrBLEU	RD↑	HE↑
Transformer-word	41.16	-	-	-
RSNI (0.2)*	29.68	0.892	2.580*	1.39*
RSNI (0.3)*	19.94	0.781	2.350*	1.10*
GS (0.2)	33.46	0.749	0.746	3.23
GS (0.3)	29.86	0.676	0.847	2.49
Ours	33.72	0.804	**0.952**	**3.73**
Transformer-BPE	44.06	-	-	-
RSNI (0.2)*	34.44	0.892	2.019*	1.45*
RSNI (0.4)*	25.78	0.781	1.891*	1.08*
GS (0.2)	35.52	0.823	**1.094**	**3.88**
GS (0.4)	28.18	0.675	1.004	2.90
Ours	35.48	0.807	1.009	3.79
RNN-search-BPE	40.90	-	-	-
RSNI (0.2)*	32.54	0.892	1.891*	1.44*
RSNI (0.4)*	25.54	0.781	1.712*	1.36*
GS (0.2)	32.94	0.823	**1.102**	3.79
GS (0.4)	27.02	0.678	1.053	2.88
Ours	31.58	0.785	1.059	**3.81**

Table 3: Experiment results for Zh→En MT attack. We list *BLEU* for perturbed test sets generated by each adversarial example generation method, which is expect to deteriorate. An ideal adversarial example should achieve high *RD* with respect to high *HE*.

	En-De newstest13-16			
	BLEU	chrBLEU	RD↑	HE↑
RNN-search-BPE	25.35	-	-	-
RSNI (0.2)*	16.70	0.949	6.691*	2.32*
RSNI (0.4)*	10.05	0.897	5.860*	1.58*
GS (0.2)	19.42	0.881	1.966	3.81
GS (0.4)	9.27	0.680	1.982	3.01
Ours	21.27	0.921	**2.037**	**3.95**
Transformer-BPE	29.05	-	-	-
RSNI (0.2)*	18.775	0.949	6.935*	2.39*
RSNI (0.4)*	11.125	0.897	5.991*	1.58*
GS (0.2)	18.29	0.861	2.665	3.69
GS (0.4)	10.03	0.751	2.629	3.33
Ours	19.29	0.875	**2.688**	**3.79**
	En-Fr newstest13-14 + newsdiscuss15			
RNN-search-BPE	32.6	-	-	-
RSNI (0.2)*	21.93	0.947	6.175*	2.23*
RSNI (0.4)*	14.3	0.894	5.271*	1.56*
GS (0.2)	22.7	0.833	1.818	3.80
GS (0.4)	15.2	0.708	1.828	3.25
Ours	22.3	0.843	**2.009**	**3.87**
Transformer-BPE	34.7	-	-	-
RSNI (0.2)*	24.0	0.947	5.774*	2.34*
RSNI (0.4)*	15.8	0.894	5.114*	1.67*
GS (0.2)	23.01	0.830	1.982	3.74
GS (0.4)	19.6	0.788	**2.053**	3.68
Ours	21.33	0.798	1.907	**3.78**

Table 4: Experiment results for En→De and En→Fr MT attack.

As Michel et al. (2019) suggest, there is a trade-off between achieving high *RD* and maintaining semantic. One can achieve rather high *RD* by testing with mismatched references, making degradation less meaningful. Therefore, we also test source semantic similarity with human evaluation (*HE*) ranging from 0 to 5 used by Michel et al. (2019) by randomly sampling 10% of total sequences mixed with baselines for a double-blind test.

4.4 Results

We implement state-of-the-art adversarial example generation by gradient search (Michel et al., 2019) (GS) as a baseline, which can be currently applied to various translation models. We also implemented random synthetic noise injection (Karpukhin et al., 2019) (RSNI) as an unconstrained contrast. Both baselines are required to provide a ratio for the amount of tokens to perturb during an attack, where we present the best results. Unlike our paradigm can generate on monolingual data, GS also requires target annotations, where we use one of the references to provide a strong baseline. Note that RSNI can significantly break semantics with distinctly lower *HE* to achieve rather high *RD*, which we do not consider as legit adversarial example generation and noted with "*" for exclusion.

As it is shown in Table 3 and 4, our model stably generate adversarial examples without significant change in semantics *by the same training setting* among different models and language pairs, achieving stably high *HE* (>3.7) without any handcrafted semantic constraints, while search methods (GS) must tune for proper ratio of modification, which can hardly strike a balance between semantic constraints and degradation. Unlike search paradigm relying on reference and victim gradients, our paradigm is model-agnostic yet still achieving comparable RD with relatively high HE.

4.5 Case Study

As it is shown in Table 5, our method is less likely to perturb some easily-modified *semantics* (e.g. numbers are edited to other "forms", but not different numbers), while search tends to generate semantically different tokens to achieve degradation. Thus our agent can lead to more insightful and plausible analyses for neural machine translation than search by gradient.

5 Analysis

5.1 Efficiency

As it is shown in Figure 2, given the same amount of memory cost, our method is significantly more

	a
origin in **origin out**	全国 4000 万选民将在16名候选人中选举法兰西第五共和国第七任总统。 40 million voters throughout the country will elect the seventh president of the fifth republic of france among the 16 candidates
references	40 million voters in the nation will elect the 7th president for the french fifth republic from 16 candidates. there are 40 million voters and they have to pick the fifth republic france's seventh president amongst the sixteen candidates. forty million voters across the country are expected to choose the 7th president of the 5th republic of france from among 16 candidates. 40 million voters around france are to elect the 7th president of the 5 republic of france from 16 candidates .
GS (0.4) in **GS (0.4) out**	全国 **性** 4000 万 **市**民将在 ***6*** 名候选人中选举法兰西第五**国家** 第七任 **外交部长**。 of the 6 candidates, 40 million people will elect the seventh foreign minister of the five countries.
ours in **ours out**	全国**性**4000万选民将在16**位**候选人中选举法兰西第***5***共和国第***7***任总统 among the 16 candidates , 40 million voters will elect five presidents of France and seven presidents of the republic of France.
	b
origin in **origin out**	千案者目前被也门当局扣留。 the persons involved in the case are currently detained by the yemeni authorities.
references	the perpetrator is currently in the custody of the yemeni authorities. yemeni authority apprehended the suspect. the suspect is now in custody of yemeni authorities . the ones involed in this case were also detained by the authority.
GS (0.4) in **GS (0.4) out**	千案者目前**为**也门**现**局留。 the person involved in the case is now detained by the authorities!
ours in **ours out**	千案**方** 目前被也门当局扣留。 the victim is currently detained by the yemeni authorities.

Table 5: (a) an example of perturbed number and quantifier severely damaging outputs in Zh→En translation, where we highlight the changes. "五" is the character for 5 and "七" for 7, "名" and "位" are both commonly used quantifiers for people. However, search-based attack achieves degradation by some significant changes of semantics, where number "16" is changed to "6", and "外交部长" means "foreign minister". (b) an example of changed suffix which breaks the result. "方" and "者" are common suffixes (K) sharing same meaning used for people. Our model spots that victim model's fragility upon such perturb, while search does not.

efficient compared to the search paradigm. Gradient computation concerning every modified source sequence can cost considerably in time or space for a state-of-the-art system, which could be even worse for systems with recurrent units. When it comes to mass production of adversarial examples for a victim translation system, our method can also generate by given only monolingual inputs. In contrast, search methods must be provided the same amount of well-informed targets.

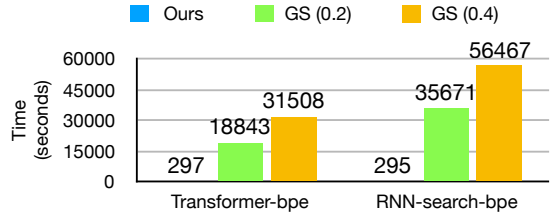

Figure 2: Time consumption of different methods: we limit memory usage to 2.5G on single Nvidia 1080, and generate adversarial examples for the same 800 inputs in Zh→En MT with different methods, our method significantly outclasses the state-of-the-art search paradigm (GS).

5.2 Attack Patterns

NMT systems may have different robustness over different parts of the inputs, thus some researchers implement input preprocessing targeting certain empirically weak parts, e.g., named entities(Li et al., 2018). Since the agent's policy is to attack without handcrafted error features, we can further investigate vulnerability by its attack preferences of different parts of speech. We choose Chinese, for example, and adopt LTP POS tagger[6] to label NIST test sets, then check the modification rate for each POS. To ensure the reliability of our analysis, we run three rounds of experiments on both baselines and our agent with *similar modification rate* targeting state-of-the-art Transformer with BPE, and collect overall results. We also present random synthetic noise injection (Karpukhin et al., 2019) (RSNI), which is not intended for any preference as an additional baseline.

As it is shown in Figure 3, our reinforced paradigm shows distinct preference upon certain POS tags, indicating pitfalls of a victim translation system. At the same time, RSNI distributed almost evenly upon different POS tags. Though the search paradigm (GS) does expose some types of pitfall, our method can further expose those omitted by the search. Note that unlike existing work relying on feature engineering to indicate errors, we have no such features implemented for an agent. However, our agent can still spot error patterns by favoring some of the POS, such as

[6]https://github.com/HIT-SCIR/ltp

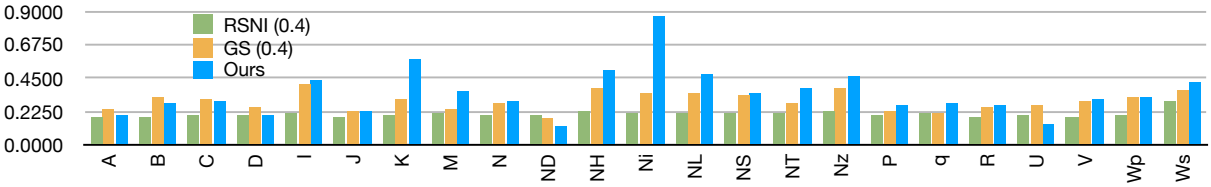

Figure 3: Attack preferences of different paradigms targeting Zh→En Transformer-BPE model. All share a similar modification rate. Our agent shows a significant preference for some POS (e.g., Ni, Nh, Nz, I), which are commonly regarded as hard-to-translate phrases among industrial implementations, while some (e.g., K) are less noticed. Preference among different choices.

	Attack by	BLEU(Δ)
Zh-En MT02-06		
RNN-search-BPE	-	40.90
	agent-RNN	31.58(-9.32)
	agent-TF	32.14(-8.76)
Transformer-BPE	-	44.06
	agent-TF	35.48(-8.58)
	agent-RNN	33.14(-10.92)
En-De Newstest13-16		
RNN-search-BPE	-	25.35
	agent-RNN	21.27(-4.08)
	agent-TF	17.18(-8.18)
Transformer-BPE	-	29.05
	agent-TF	19.29(-9.76)
	agent-RNN	24.2(-4.85)
En-Fr Newstest13-14+newsdiscuss15		
RNN-search-BPE	-	32.60
	agent-RNN	22.3(-10.30)
	agent-TF	19.83(-14.87)
Transformer-BPE	-	34.70
	agent-TF	21.33(-13.37)
	agent-RNN	22.35(-10.25)

Table 6: Attacks targeting different architecture from the trained one. We note agent with the architecture that is trained with(e.g., agent-RNN stands for agent trained by targeting RNN-search).

Ni (organization name), Nh (person name), Nl (location name), M (numbers), which are commonly accepted as hard-to-translate parts. Moreover, the agent also tends to favor K (suffix) more, which is less noticed.

5.3 Attack Generalization

We additionally test agents by attacking different model architecture from the one that it's trained. As it is shown in Table 6, we perturb the inputs by agents trained to attack a different architecture, then test for degradation. The results show that our agent trained by targeting Transformer architecture can still achieve degradation on RNN-search, and vice-versa.

	Clean test	Noisy test	IWSLT11-17
Transformer-BPE	44.06	35.48	11.27
Tuned	43.60(-0.46)	40.31(+4.83)	11.73(+0.46)

Table 7: Tuning Zh→En Transformer-BPE model with adversarial examples. We generate adversarial examples for every training sources for tuning, achieving overall improvements for noisy tests.

5.4 Tuning with Adversarial Examples

Since the agent generates meaning-preserving adversarial examples efficiently, we can directly tune the original model with those samples. We choose Zh→En Transformer-BPE, for example, and generate the same amount of adversarial examples given original training sources(1.3M pairs), then paired with initial targets. We mix the augmented pairs with original pairs for a *direct* tuning. We test the tuned model on original test data and noisy test data generated by the attacking agent. We additionally test on IWSLT11-17 Zh→En test data, which is not used for training or tuning, to verified robustness improvement. As Table 7 shows, our agent can achieve substantial improvement(+4.83) on noisy tests with only minor loss on clean tests(-0.46). The improvement on the IWSLT test also indicates the adversarial tuning contributes to not only defending the agent's attack, but also overall robustness.

5.5 Reinforced Examples for Machine translation

We additionally switched the episodic rewards in the environment, then ignored all modifications that induce UNK tokens to train an agent, hoping to generate minor perturbed samples that can improve the translation metric. Though we failed to achieve overall improvements, we do succeed for quite a portion of samples, as shown in Table 8. Similar to adversarial examples, we call them *reinforced examples*. Such improvement is different from adversarial training that tunes model

in	钱其琛同突尼斯外长会谈。
perturbed in	钱其琛同突尼斯外长会谈-
out	Chinese, Tunisian minsters hold talks.
perturbed out	qian qichen holds talks with Tunisian foreign minister.
in	中巴及城巴车辆在南区通宵停泊
perturbed in	中巴及城巴车辆在南区通宵停**车**
out	overnight parking of cmb and city bus
perturbed out	overnight parking of cmb and city bus in southern district

Table 8: Example of minor perturbed samples that improves machine translation for Zh→En Transformer-BPE model. The "。" in first sample is modified to "-", then model yields the omitted "钱其琛 (qian qi chen)". The "停泊" in second sample is modified to "停车", where they both mean "parking", then comes the omitted "in southern district" for "在南区".

for defense or strict text correction before the test phase. Reinforced examples are still noisy and can be directly applied for a test without any model updates to achieve improvements, which to our best knowledge is less investigated by researchers. Since we discovered that not all perturbed inputs are harmful, such an issue can be a good hint and alternative for better adversarial defense in NLP and should be further considered.

6 Related Work

Cheng et al. (2018a) and Cheng et al. (2018b) applied continuous perturbation learning on token's embedding and then manage a lexical representation out of a perturbed embedding. Zhao et al. (2017) learned such perturbation on the encoded representation of a sequence, and then decode it back as an adversarial example. These methods are applicable for simple NLP classification tasks, while failing machine translation which requires higher semantic constraints. Zhao et al. (2017) further attempted to constrain semantic in such paradigm by introducing multi-task modeling with accessory annotation, which further limits applicability.

On the other hand, Ebrahimi et al. (2018), Chaturvedi et al. (2019) and Cheng et al. (2019) regarded it as a search problem by maximizing surrogate gradient losses. Due to the formidable gradient computation, such methods are less viable to more complex neural architectures. Cheng et al. (2019) introduced a learned language model to constrain generation. However, a learned language model is not apt for common typos or UNK. Another pitfall of this paradigm is that surrogate losses defined by a fixed tokenization for non-character level systems, risks being invalidated once the attack changes tokenization. Therefore, Ebrahimi et al. (2018) simply focused on char-level systems, while Michel et al. (2019) specially noted to exclude scenarios where attack changes tokenization in their paradigm.

Other works turn to more sophisticated generation paradigms, e.g., Vidnerová and Neruda (2016) adopts a genetic algorithm for an evolutionary generation targeting simple machine learning models. Zang et al. (2019) consider adversarial generation as a word substitution-based combinatorial optimization problem tackled by particle swarm algorithm. Our paradigm shares some common ideology with Miao et al. (2019) and Xiao et al. (2018), which iteratively edit inputs constrained by generative adversarial learning.

7 Conclusion

We propose a new paradigm to generate adversarial examples for neural machine translation, which is capable of exposing translation pitfalls without handcrafted error features. Experiments show that our method achieves stable degradation with meaning preserving adversarial examples over different victim models.

It is noticeable that our method can generate adversarial examples efficiently from monolingual data. As a result, the mass production of adversarial examples for the victim model's analysis and further improvement of robustness become convenient. Furthermore, we notice some exceptional cases which we call as "reinforced samples", which we leave as the future work.

Acknowledgement

We would like to thank the anonymous reviewers for their insightful comments. Shujian Huang is the corresponding author. This work is supported by National Science Foundation of China (No. 61672277, 61772261), the Jiangsu Provincial Research Foundation for Basic Research (No. BK20170074).

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Akshay Chaturvedi, KP Abijith, and Utpal Garain. 2019. Exploring the robustness of nmt systems to nonsensical inputs. *arXiv: Learning*.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018a. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. *arXiv preprint arXiv:1906.02443*.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018b. Towards robust neural machine translation. *arXiv preprint arXiv:1805.06130*.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On adversarial examples for character-level neural machine translation. *arXiv preprint arXiv:1806.09030*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Vladimir Karpukhin, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. 2019. Training on synthetic noise improves robustness to natural noise in machine translation. *arXiv preprint arXiv:1902.01509*.

Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.

Katherine Lee, Orhan Firat, Ashish Agarwal, Clara Fannjiang, and David Sussillo. 2018. Hallucinations in neural machine translation. *NIPS 2018 Workshop IRASL*.

Zhongwei Li, Xuancong Wang, Ai Ti Aw, Eng Siong Chng, and Haizhou Li. 2018. Named-entity tagging and domain adaptation for better customized translation. In *Proceedings of the Seventh Named Entities Workshop*, pages 41–46, Melbourne, Australia. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. *arXiv preprint arXiv:1903.06620*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Motoki Sano, Jun Suzuki, and Shun Kiyono. 2019. Effective adversarial regularization for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 204–210.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Petra Vidnerová and Roman Neruda. 2016. Evolutionary generation of adversarial examples for deep and shallow machine learning models. In *Multidisciplinary International Social Networks Conference*.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1808.08866*.

Chaowei Xiao, Bo Li, Jun yan Zhu, Warren He, Mingyan Liu, and Dawn Song. 2018. Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3905–3911. International Joint Conferences on Artificial Intelligence Organization.

Yuan Zang, Chenghao Yang, Fanchao Qi, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2019. Textual adversarial attack as combinatorial optimization. *arXiv:1910.12196v2*.

Yang Zhao, Jiajun Zhang, Zhongjun He, Chengqing Zong, and Hua Wu. 2018. Addressing troublesome words in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 391–400.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*.

Brian D Ziebart. 2010. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Ph.D. thesis, figshare.

A Training Details for Agent

We adopt commonly accepted translation metric BLEU as $score$ in Eq.9. We use 50 sequence pairs per batch both in environment initialization and training of discriminator and agent. It is essential to train on batches of sequences to stabilize reinforced training. Furthermore, note that D can be too powerful during the early training stage compared to the agent's actor that it can quickly terminate an exploration. Therefore, we must train on batches and determine an overall terminal signal as aforementioned to ensure early exploration. The $step_D$ and $step_A$ are set as 80 [7] and 120. acc_bound for discriminator training is set to 0.85. The a and b in Eq.8 are set to 0.5 and 10. The dimension of feedforward layers in the agent's actor-critic and discriminator are all 256. We initialize the embedding of both agent and discriminator by the victim's embedding.

For reinforcement learning, we adopt asynchronous learning with an additional global agent with an additional set of parameter θ^{Ω}, we set discount factor γ to 0.99, α and β in Eq.5 to 0.5 and 0.05 respectively. As for the stop criterion, we set $patience\_round$ to 15 with convergence boundary for acc_D to 0.52. We adopt Adafactor(Shazeer and Stern, 2018) for training, which is a memory-efficient Adam. The learning rate for agent's optimizer is initiated as 0.001 and scheduled by rsqrt with 100 steps of warmup. The K for the candidate set is 12.

Our agent takes around 30 hours to converge on a single Nvidia 1080ti. Note that higher acc_bound and lower convergence boundary for D indicates higher semantic constraints, which will increase training time.

B Search-based Attack

Search-based adversarial generation is currently widely applied in various robustness machine translation system. We generally follow the strategy of Ebrahimi et al. (2018); Michel et al. (2019) which is applicable for both RNN-search and Transformer. More specifically, the L_{adv} in Eq.7 is derived as:

$$\underset{1\leq i\leq n, emb'_i\in vocab}{\operatorname{argmax}} |emb' - emb_i| \nabla_{emb_i} L_{adv}, \quad (12)$$

$$L_{adv}(X', Y) = \sum_{t=1}^{|y|} log(1 - P(y_t|X', y_1 ... y_{t-1}))$$

[7] Three times the average convergence episodes to train a discriminator with initial agent by the given batch size.

Algorithm 1: Reinforced training for agent

Result: A learned global agent $\pi_{\theta^{\Omega}}$

Assume global agent as $\pi_{\theta^{\Omega}}$ with parameter θ^{Ω}
Assume agent as π_{θ} with parameter set θ
initialize: Env with D, θ^{Ω}, θ ;
while *not Stop Criterion* **do**
  for $step_D$ **do**
    train D with current agent π_{θ} ;
    if $acc_D > acc\_bound$ break;
  end
  test current D's accuracy acc_D for stop criterion;
  for $step_A$ **do**
    initialize Env state s_0;
    synchronize π_{θ} with $\pi_{\theta^{\Omega}}$;
    $t = t_{start}$;
    while s_t *survive and* $t - t_{start} < t_{max}$ **do**
      get $out_t^{\text{actor}}, V_t = \pi_{\theta}(s_t)$;
      compute entropy $H(out_t^{\text{actor}})$;
      sample a_t based on out_t^{actor} ;
      perform a_t and receive r_t and s_{t+1} ;
      $t \leftarrow t + 1$;
    end
    $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t) & \text{for non-terminal } s_t \end{cases}$
    for $i \in \{t-1, ..., t_{start}\}$ **do**
      $R \leftarrow \gamma R + r_i$;
      accumulate $L_t^v(\theta)$;
      accumulate $L_t^{\pi}(\theta)$;
    end
    compute overall loss $L(\theta)$;
    perform asynchronous updates on θ^{Ω} with gradient $\frac{\partial L(\theta)}{\partial \theta}$;
  end
end

where each $P(y_t|X)$ is calculated by Eq.1 given a corresponding target. For every source sequence, a small ratio of positions is sampled for search. Then we greedy search[8] by the corresponding loss upon those positions with given candidates. For better comparison, we adopt the candidate set used in our model instead of naive KNN candidates. Both baseline and our model share the same UNK generation for presentation. We use homophone replacement for Chinese, and strategy by Michel et al. (2019) for English.

[8] Ebrahimi et al. (2018) suggest that greedy search is a good enough **approximation**.