Improving Grammatical Error Correction Models with Purpose-Built Adversarial Examples

Lihao Wang, Xiaoqing Zheng
School of Computer Science, Fudan University, Shanghai, China
Shanghai Key Laboratory of Intelligent Information Processing
{wanglh19, zhengxq}@fudan.edu.cn

Abstract

A sequence-to-sequence (seq2seq) learning with neural networks empirically shows to be an effective framework for grammatical error correction (GEC), which takes a sentence with errors as input and outputs the corrected one. However, the performance of GEC models with the seq2seq framework heavily relies on the size and quality of the corpus on hand. We propose a method inspired by adversarial training to generate more meaningful and valuable training examples by continually identifying the weak spots of a model, and to enhance the model by gradually adding the generated adversarial examples to the training set. Extensive experimental results show that such adversarial training can improve both the generalization and robustness of GEC models.

1 Introduction

The goal of Grammatical Error Correction (GEC) is to identify and correct different kinds of errors in the text, such as spelling, punctuation, grammatical, and word choice errors, which has been widely used in speech-based dialogue, web information extraction, and text editing software.

A popular solution tackles the grammatical error correction as a monolingual machine translation task where ungrammatical sentences are regarded as the source language and corrected sentences as the target language (Ji et al., 2017; Chollampatt and Ng, 2018a). Therefore, the GEC can be modeled using some relatively mature machine translation models, such as the sequence-to-sequence (seq2seq) paradigm (Sutskever et al., 2014).

They are many types of grammatical errors and all of them can occur in a sentence, which makes it impossible to construct a corpus that covers all kinds of errors and their combinations. Deep learning so far is data-hungry and it is hard to train a seq2seq model with good performance without sufficient data. Therefore, recent studies have turned the focus to the methods of generating high-quality training samples (Xie et al., 2018; Lichtarge et al., 2019; Zhao et al., 2019). Generating pseudo training data with unlabeled corpora can be roughly divided into direct noise and back translation (Kiyono et al., 2019). The former applies text editing operations such as substitution, deletion, insertion and shuffle, to introduce noises into original sentences, and the latter trains a clean-to-noise model for error generation. However, the noise-corrupted sentences are often poorly readable, which are quite different from those made by humans. The sentences generated by the back translation also usually cover a few limited types of errors only\(^1\), and it is difficult for the back translation to generate the errors not occurred in the training set. Although these methods can produce many ungrammatical

---

\(^1\)Most of the errors are generated by replacing prepositions and inserting or deleting articles, which occur more frequently in the GEC training set.

| Method           | Sentences                                                                 |
|------------------|---------------------------------------------------------------------------|
| Clean            | His reaction should give you an idea as to whether this matter is or is not your business. |
| Direct Noise     | His reaction should give you an idea as to whether this matter is or so is not your business. |
| Back Translation | His reaction should give you the idea as to whether this matter is or is not your business. |
| Adversarial Example | His reaction should gave you an idea as to whether this matter is or is n’t their business. |

Table 1: Example ungrammatical sentences generated by different methods. The direct noise method seems to add some meaningless noises to the original sentence. Most of the grammatical errors generated by the back translation are those produced by replacing prepositions, and inserting or deleting articles. The example generated by our adversarial attack algorithm looks more meaningful and valuable.
examples, most of them have little contribution to improving the performance.

We also found that the resulting models are still quite vulnerable to adversarial examples, although they are trained with the data augmented by their methods. Taking a state-of-the-art system of Zhao et al. (2019) on CoNLL-2014 (Ng et al., 2014) as an example, we generate adversarial samples by intentionally introducing few grammatical errors into the original clean sentences under a white-box setting2. The model’s performance of $F_{0.5}$ drops from 0.592 to 0.434 if just one grammatical error is added into each sentence, to 0.317 if three errors are added. To our knowledge, we first show in this study that adversarial examples also exist in grammatical error correction models.

Inspired by adversarial attack and defense in NLP (Jia and Liang, 2017; Zhao et al., 2017; Cheng et al., 2018), we explore the feasibility of generating more valuable pseudo data via adversarial attack, targeting the weak spots of the models, which can improve both the quality of pseudo data for training the GEC models and their robustness against adversarial attacks. We propose a simple but efficient method for adversarial example generation: we first identify the most vulnerable tokens with the lowest generation probabilities estimated by a pre-trained GEC model based on the seq2seq framework, and then we replace these tokens with the grammatical errors people may make to construct the adversarial examples.

Once the adversarial examples are obtained, they either can be merged with the original clean data to train a GEC model or used to pre-train a model thanks to their coming with great numbers. The examples generated by our method based on the adversarial attack are more meaningful and valuable than those produced by recent representative methods, such as the direct noise and the back translation (see Table 1). Through extensive experimentation, we show that such adversarial examples can improve both generalization and robustness of GEC models. If a model pre-trained with large-scale adversarial examples is further fine-tuned by adversarial training, its robustness can be improved about 9.5% while without suffering too much loss (less than 2.4%) on the clean data.

2 Related Work

2.1 Grammatical Error Correction

The rise of machine learning methods in natural language processing (NLP) has led to a rapid increase in data-driven GEC research. The predominant paradigm for the data-driven GEC is arguably sequence-to-sequence learning with neural networks (Yuan and Briscoe, 2016; Xie et al., 2016; Sakaguchi et al., 2017; Schmaltz et al., 2017; Ji et al., 2017), which is also a popular solution for machine translation (MT).

Some task-specific techniques have been proposed to tailor the seq2seq for the GEC task. Ji et al. (2017) proposed a hybrid neural model using word and character-level attentions to correct both global and local errors. Zhao et al. (2019) explicitly applied the copy mechanism to the GEC model, reflecting the fact that most words in sentences are grammatically correct and should not be changed. Diverse ensembles (Chollampatt and Ng, 2018a), rescoring (Chollampatt and Ng, 2018b), and iterative decoding (Ge et al., 2018; Lichtarge et al., 2018) strategies also have been tried to tackle the problem of incomplete correction.

Although the advancement of the GEC has made an impressive improvement, the lack of training data is still the main bottleneck. Very recently, data augmentation techniques began to embark on the stage (Xie et al., 2018; Zhao et al., 2019). The GEC models that achieved the competitive performance (Kiyono et al., 2019; Zhao et al., 2019; Lichtarge et al., 2019; Grundkiewicz et al., 2019) were usually pre-trained on large unlabeled corpora and then fine-tuned on the original training set.

2.2 Textual Adversarial Attack

Fooling a model by perturbing its inputs, which is also called an adversarial attack, has become an essential means of exploring the model vulnerabilities. To go a further step, incorporating adversarial samples in the training stage, also known as adversarial training, could effectively improve the models’ robustness. Depending on the degree of access the target model, adversarial examples can be constructed in two different settings: white-box and black-box settings. An adversary can access the model’s architecture, parameters, and input feature representations in the white-box setting while not in the black-box one. The white-box attacks typically yield a higher success rate because the knowledge of target models can guide the genera-

---

2 In contrast to a black-box setting, an attacker can access to the model’s architecture, parameters, and training data set under the white-box setting.
We formally define the GEC task and then briefly introduce the seq2seq baseline. As we mentioned above, the GEC can be modeled as an MT task by viewing an ungrammatical sentence \( x \) as the source sentence and a corrected one \( y \) as the target sentence. Let \( D = \{ (x, y) \} \) be a GEC training dataset. The seq2seq model first encode a source sentence having \( N \) tokens into a sequence of context-aware hidden representations \( h^s_{1:N} \), and then decodes the target hidden representations \( h^t_i \) from the representations of \( h^s_{1:N} \). Finally, the target hidden representations can be used to produce the generation probability \( p(y_i | y_{1:i-1}, x) \), also called *position score* \( g(y_i) \), and to generate the output sequence \( y_{1:i-1} \) through the projection matrix \( WH \) and softmax layer as follows:

\[
h^s_i = \text{encoder}(x_i) \tag{1}
\]

\[
p(y_i | y_{1:i-1}, x) = \text{softmax}(h^t_i W^H) \tag{2}
\]

\[
g(y_i) = \log(p(y_i | y_{1:i-1}, x)) \tag{3}
\]

The negative log-likelihood of generation probabilities is used as the objective function, where \( \theta \) are all the model parameters to be trained.

\[
\mathcal{L}(\theta) = -\frac{1}{|D|} \sum_{(x, y) \in D} \log(p(y | x)) \tag{4}
\]

## 3 Method

### 3.1 Baseline Model

We formally define the GEC task and then briefly introduce the seq2seq baseline. As we mentioned above, the GEC can be modeled as an MT task by viewing an ungrammatical sentence \( x \) as the source sentence and a corrected one \( y \) as the target sentence. Let \( D = \{ (x, y) \} \) be a GEC training dataset. The seq2seq model first encode a source sentence having \( N \) tokens into a sequence of context-aware hidden representations \( h^s_{1:N} \), and then decodes the target hidden representations \( h^t_i \) from the representations of \( h^s_{1:N} \). Finally, the target hidden representations can be used to produce the generation probability \( p(y_i | y_{1:i-1}, x) \), also called *position score* \( g(y_i) \), and to generate the output sequence \( y_{1:i-1} \) through the projection matrix \( WH \) and softmax layer as follows:

\[
h^s_i = \text{encoder}(x_i) \tag{1}
\]

\[
p(y_i | y_{1:i-1}, x) = \text{softmax}(h^t_i W^H) \tag{2}
\]

\[
g(y_i) = \log(p(y_i | y_{1:i-1}, x)) \tag{3}
\]

The negative log-likelihood of generation probabilities is used as the objective function, where \( \theta \) are all the model parameters to be trained.

\[
\mathcal{L}(\theta) = -\frac{1}{|D|} \sum_{(x, y) \in D} \log(p(y | x)) \tag{4}
\]

### 3.2 Adversarial Example Generation

We found that adversarial examples also exist in the GEC models and up to 100% of input examples admit adversarial perturbations. Adversarial examples yield broader insights into the targeted models by exposing them to such maliciously crafted examples. We try to identify the weak spots of GEC models by a novel adversarial example generation algorithm that replaces the tokens in a sentence with the grammatical mistakes people may make. Our adversarial example generation algorithm also uses the two-step recipe: first determining the important tokens to change and then replacing them with the grammatical mistakes that most likely occur in the contexts.

#### 3.2.1 Identifying Weak Spots

We use the positional scores to find the vulnerable tokens (or positions) that most likely can successfully cause the models to make mistakes once they are modified. The lower the positional score of a token is, the lower confidence the model gives its prediction, and the more likely this prediction will fully cause the models to make mistakes once they are modified. The lower the positional score of a token is, the lower confidence the model gives its prediction, and the more likely this prediction will fully cause the models to make mistakes once they are modified. The lower the positional score of a token is, the lower confidence the model gives its prediction, and the more likely this prediction will fully cause the models to make mistakes once they are modified.

We found that the words having relatively lower positional scores are lexical or open class words such as nouns, verbs, adjectives and adverbs. Besides, rare and out-of-vocabulary words are also given low positional scores. By adding the pseudo examples generated by making small perturbations to those tokens, we can force a GEC model to better explore the cases that may not be encountered before. If the function words are used correctly, the model usually gives higher positional scores to them; otherwise, the model will lower its confi-
ence in the prediction by decreasing such scores, which is precisely what we expect.

We here formally describe how to use the positional scores to locate the weak spots of a sentence. Like (Bahdanau et al., 2014; Ghader and Monz, 2017), we first use the attention weights $\alpha_{i,j}$ of a seq2seq-based model to obtain the soft word alignment between the target token $y_i$ and the source one $x_j$ by Equation (6) and (7) below:

$$q_i, k_j = h^Q_i W^Q, h^K_j W^K$$

$$\alpha_{i,j} = \text{softmax}(\frac{q_i \cdot k_j}{\sqrt{d_k}})$$

where $W^Q$ and $W^K$ denote the projection matrices required to produce the representations of a query $q_i$ and a key $k_j$ from which an attention score can be computed, and $d_k$ is the dimension of $h^S_j$. We then can obtain a word alignment matrix $A$ from the attention weights $\alpha_{i,j}$ as follows:

$$A_{i,j}(\alpha) = \begin{cases} 1 & j = \arg\max_j \alpha_{i,j'} \\ 0 & o/w \end{cases}$$

(8)

When $A_{i,j} = 1$, we known that $y_i$ is aligned to $x_j$. If $y_i$ is identified as a vulnerable token, we try to make perturbation to $x_j$ to form an attack. The positional scores $g(y_i)$ are obtained by the GEC model trained on the original training set. If the token’s positional score $g(y_i)$ is less than a given threshold $\epsilon$, we take $x_j$ as a candidate to be modified to fool the target GEC model.

$$A_{i,j} = 1, g(y_i) < \epsilon$$

(9)

3.2.2 Word Substitution-based Perturbations

Although adversarial examples have been studied recently for NLP tasks, previous work almost exclusively focused on semantic tasks, where the attacks aim to alter the semantic prediction of models (e.g., sentiment prediction or question answering) without changing the meaning of original texts. Once a vulnerable position is determined, the token at that position is usually replaced with one of its synonyms. However, generating adversarial examples through such synonym-based replacement is no longer applicable to the GEC task.

Motivated by this, we propose two methods to replace vulnerable tokens. One is to create a correction-to-error mapping from a GEC training set and get a suitable substitute using this mapping. If there are multiple choices, we sample a substitute from those choices according to the similarity of their context. Another is to generate a substitute based on a set of rules that imitates human errors. We give a higher priority to the former than the latter when generating the adversarial examples for the GEC.

**Context-Aware Error Generation** From a parallel training set of GEC, we can build a correction-to-error mapping, by which given a token, we can obtain its candidate substitutes (with grammatical errors) and their corresponding sentences. Assuming that a token is selected to be replaced, and its candidate substitutes are retrieved by the mapping, we want the selected substitute can fit in well with the token’s context and maintain both the semantic and syntactic coherence. Therefore, we define a function $s$ based on the edit distance (Marzal and Vidal, 1993) to estimate the similarity scores between two sentences. This function allows us to estimate how well a substitute’s context sentence $c_i'$ is similar to an original sentence $c_i$ to be intentionally modified. To encourage the diversity of generated examples, we choose to sample a substitute from the candidates according to the weights $w_i$ derived from their sentences’ similarity scores to the original one as follows.

$$w_i = \begin{cases} \frac{s(c_i, c_i')}{\lambda} & s(c_i, c_i') > \lambda \\ \frac{1}{w_i} & \lambda \leq s(c_i, c_i') \leq \lambda \end{cases}$$

(10)

Equation (11) describes a weighted random sampling (Efraimidis and Spirakis, 2006) process in which the weights are calculated by the function $s(c_i, c_i')$ and truncated by a threshold $\lambda$. Note that polysemous words should be carefully handled. For example, the word “change” has two semantic terms with different part-of-speech of noun and verb, which produce different errors. Therefore, we remove the candidates that do not have the same part-of-speech as the original token.

**Rule-based Error Generation** If we cannot find any candidate substitute, we use a set of predefined rules to generate the substitute. Table 2 lists diverse word transformations for error generation according to the tokens’ part-of-speech. To maintain the sentence’s semantics to the greatest extent, we just transform the nouns to their singular and plural counterparts instead of searching the synonyms from dictionaries. For verbs, we randomly choose their present, progressive, past, perfect, or third-person-singular forms. Adjectives and corre-
Table 2: Rule-based word perturbations according to the part-of-speech.

| Type of tokens | Perturbation                                      |
|----------------|---------------------------------------------------|
| Noun           | To its singular or plural forms                   |
| Verb           | Change its tense or number (third-person-singular form) |
| Adjective      | To its adverb form                                |
| Adverb         | To its adjective form                             |
| Numerals       | Remaining unchanged                               |
| Proper Nouns   | Remaining unchanged                               |
| Punctuation    | Deletion                                          |
| Others         | <unk> or Deletion                                  |

Table 2: Rule-based word perturbations according to the part-of-speech.

Type of tokens | Perturbation
--- | ---
Noun | To its singular or plural forms
Verb | Change its tense or number (third-person-singular form)
Adjective | To its adverb form
Adverb | To its adjective form
Numerals | Remaining unchanged
Proper Nouns | Remaining unchanged
Punctuation | Deletion
Others | <unk> or Deletion

Adversarial examples generation

**Input:**
- \(x\): A grammatical sentence with \(n\) words.
- \(f\): A target GEC model.
- \(D_t\): The training corpus of GEC.
- \(\epsilon\): A threshold to select vulnerable tokens.

**Output:**
- \(S\): Adversarial examples set towards \(x\).

1. Set \(S = \emptyset\) and extract the correction-to-error mappings \(m(x) \ x \in M\), from the corpus \(D_t\).
2. for each \(j \in [1, n]\) do
   3. Calculate the position score \(g(y_i) = f(x_j)\)
   4. if \(g(y_i) < \epsilon\) then
      5. if \(x_j \in M\) then
         6. sample the error \(e\) from \(m(x_j)\)
      7. else
         8. Obtain \(e\) by rule-base method \(r(x_j)\)
      9. end if
   10. \(x' = [x_1:j−1, e, x_j+1:n]\)
   11. if \(f(x') \neq x\) then
      12. \(S = S \cup \{x'\}\)
   13. end if
   14. end if
15. end for
16. return \(S\);

Leveraging Large Unlabeled Corpora

The standard BEA-19 training set (Bryant et al., 2019) has only about 640,000 sentence pairs, which is very insufficient for the GEC task. Thus, the clean sentences in large unlabeled corpora, such as Wikipedia or Gigaword (Napoles et al., 2012), and One billion word benchmark (Chelba et al., 2013), are usually used as seeds to generate ungrammatical sentences for data augmentation. Some studies found that the more unlabeled corpora used, the more improvement the GEC model will achieve (Kiyono et al., 2019). Unlabeled corpora also contribute to correct out-of-training-set errors. If a sentence in the test set contains these unseen errors, the GEC model training without external corpora is hard to correct. Therefore, We also leverage the unlabeled corpora and obtain large-scale adversarial examples for the later training.

3.3 Adversarial Training

We also show that GEC models’ robustness can be improved by crafting high-quality adversaries and including them in the training stage while suffering little to no performance drop on the clean input data. In this section, we conduct adversarial training with sentence pairs generated by large unlabeled corpora and adopt the pre-training and fine-tuning training strategy.

**Training strategy**

Adversarial training by means of adding the adversarial examples into the training set can effectively improve the models’ robustness. However, Some studies show that the models tend to overfit the noises, and the accuracy of the clean data will drop if the number of adversarial examples dominates the training set. Zhao et al. (2019) and Kiyono et al. (2019) adopt the pre-training and fine-tuning strategy to alleviate the noise-overfit problem. Similarly, our model is pre-trained with
adversarial examples then fine-tuned on the original training set instead of adding the large-scale data to the training set directly. The training strategy can be formally divided into four steps:

1. Train a base model \( f \) on training set \( D_t \).
2. Generate the adversarial examples set \( D_e \) on unlabeled sentences by attacking \( f \).
3. Pre-train the model \( f \) on the \( D_e \).
4. Fine-tune it on the training set \( D_t \).

We also can use adversarial training to improve the model’s robustness:

1. Generate adversarial examples set \( D_{adv} \) from the training set \( D_t \) [Optional].
2. Add examples into the training set and train the model on \( D_t = D_t \cup D_{adv} \) [Optional].

We can alternately run the step (5) and (6) to further improve models’ robustness.

4 Experiments

4.1 Datasets and Evaluations

Like the previous studies of GEC models, we use the BEA-2019 workshop official dataset\(^5\) (Bryant et al., 2019) as our training and validation data. We remove the sentence pairs with identical source and target sentences from the training set and sample about 1.2M sentences without numerals and proper nouns from the Gigaword dataset\(^4\) as our unlabeled data for pre-training. Table 3 shows the statistics of the datasets.

Our reported results are measured by the Max-Match (\( M^2 \)) scores\(^5\) (Dahlmeier and Ng, 2012) on CoNLL-2014 (Ng et al., 2014) and use the GLEU metric for JFLEG\(^6\) (Napoles et al., 2017). Following Ge et al. (2018); Junczys-Dowmunt et al. (2018); Chollampatt and Ng (2018a); Zhao et al. (2019), we apply spell correction with a 50,000-word dictionary extracted in Lang-8 corpus (Tajiri et al., 2012) before evaluation.

4.2 Models and Hyper-parameter Settings

We adopt Transformer (Vaswani et al., 2017) as our baseline seq2seq model implemented in the fairseq toolkit (Ott et al., 2019), and apply byte-pair-encoding (BPE) (Sennrich et al., 2015)

\(^5\)https://www.cl.cam.ac.uk/research/ml/bea2019st/
\(^4\)English Gigaword Fourth Edition (LDC2009T13)
\(^5\)https://github.com/muslinslp/m2scorer
\(^6\)available at https://github.com/keisks/jfleg

| Dataset      | #sent pairs | #split |
|--------------|-------------|--------|
| BEA-train    | 635582      | train  |
| BEA-valid    | 4384        | valid  |
| CoNLL-2014   | 1312        | test   |
| JFLEG        | 747         | test   |
| Gigaword*    | 1.2M        | pre-train |

Table 3: The statistics of data sets used in our experiments. A subset of Gigaword, denoted by Gigaword*, was randomly sampled from the entire Gigaword.\(^4\) The sentences containing numerals or proper nouns were not be sampled.

to source and target sentences and the number of merge operation is set to 8,000.

The base model is iterated 20 epochs on the training set. For adversarial examples generation, we set the threshold \( \epsilon \) to \(-0.2\), and use the edit distance to calculate the context similarity with the minimum weight \( \lambda = 0.1 \). To avoid the sentences being changed beyond recognition, we choose at most 6 tokens with the lowest positional score to attack. After generation, we use these data to pre-train the base model for 10 epochs and then fine-tune on the training set for 15 epochs. Our model is also further fine-tuned by adversarial training several epochs. Here, “one epoch” means that we generate an adversarial example against the current model for each sentence in the training set, and the model is continuously trained on those generated examples for three normal epochs. We also implement other data augmentation methods as comparison models on the same unlabeled data and training settings. For direct noise, the operation choice is made based on the categorical distribution \((\mu_{add}, \mu_{del}, \mu_{replace}, \mu_{keep}) = (0.1, 0.1, 0.1, 0.7)\), then shuffle the tokens by adding a normal distribution bias \(N(0, 0.5^2)\) to the positions and re-sort the tokens. For back translation, we trained a reverse model with original sentence pairs and generated the error sentences with diverse beam strength \( \beta = 6 \) following Kiyono et al. (2019).\(^7\)

To measure the model’s robustness, we conduct adversarial attacks on the model by our adversarial examples generation method to add one or three errors to the test text, respectively (to ensure that errors are always generated during the attack, we set \( \epsilon = 0 \)). We measure the models’ robustness by the drop of \( F_{0.5} \) scores and the correction rates of newly added errors. Due to the randomness of the

\(^7\)The detailed hyper-parameters are listed in Appendix A.
We also compare our model with the current top models. Our model outperforms Junczys-Dowmunt et al. (2018) and reach the same level of as Lichtarge et al. (2019), Grundkiewicz and Junczys-Dowmunt (2018). Notably, Lichtarge et al. (2019) pre-trained their model with the data generated by back translation on a subset of Wikipedia, which contains 170M sentences, and we only use 1.2M unlabeled data, which also indicates the high value of our adversarial examples. Similarly, Zhao et al. (2019) leveraged 30M unlabeled data (direct noise) and Kiyono et al. (2019) 70M (back translation) for pre-training, taking the performance of the model to a new level.

We also conducted ablation experiments to evaluate the different components of our adversarial error generation model. If a random strategy is used to select the weak spots, the $F_{0.5}$ score drops.
We conduct the attack experiments on the base, pre-trained and adversarially trained models, including the model provided by Zhao et al. (2019)\(^8\). Table 5 shows the results of adversarial attack experiments. Under the attack, our models dropped the less in performance. We also analyzed the trade-off between generalization and robustness. (i) The results in Table 5 are worth exploring, from which we can draw several other conclusions: (i) Current seq2seq-based GEC models, including some state-of-the-art models, are vulnerable to adversarial examples. (ii) The models using the direct noise method for data augmentation, Zhao et al. (2019) for example, are less robust to adversarial attacks even than their vanilla versions. It is likely to be associated with the editing operations of direct noise injecting a lot of task-irrelevant noise, which might be detrimental to the robustness of the model. (iii) Combined with adversarial training, the robustness of the model can be improved continually at the cost of acceptable loss in performance. We also analyzed the trade-off between generalization and robustness at the adversarial training stage. The results are visualized in Figure 1.

4.4 Analysis of Adversarial Attack

We conduct the attack experiments on the base, pre-trained and adversarially trained models, including the model provided by Zhao et al. (2019)\(^8\). Table 5 shows the results of adversarial attack experiments. Under the attack, our models dropped the less in performance. We also analyzed the trade-off between generalization and robustness. (i) The results in Table 5 are worth exploring, from which we can draw several other conclusions: (i) Current seq2seq-based GEC models, including some state-of-the-art models, are vulnerable to adversarial examples. (ii) The models using the direct noise method for data augmentation, Zhao et al. (2019) for example, are less robust to adversarial attacks even than their vanilla versions. It is likely to be associated with the editing operations of direct noise injecting a lot of task-irrelevant noise, which might be detrimental to the robustness of the model. (iii) Combined with adversarial training, the robustness of the model can be improved continually at the cost of acceptable loss in performance. We also analyzed the trade-off between generalization and robustness at the adversarial training stage. The results are visualized in Figure 1.

Figure 1: Trade-off between generalization and robustness. The blue and orange lines respectively denote the scores of \(F_{0.5}\) (generalization) and the correction rates (robustness) with four different training epochs. The model’s robustness will continue to improve with a slight drop in performance.

We would also like to know which type of words to modify is most likely to form a successful attack. Therefore, we calculate the correction rates of the newly added errors with different types of part-of-speech. Table 6 shows that: (i) The robustness of the model after adversarial training is significantly improved against the attack that tries to replace the lexical words such as nouns and verbs. It shows that the generated examples by our adversarial attack algorithm cover a variety of grammatical errors involving various POS types. (ii) The errors involving adjectives and adverbs are less likely to be corrected without adversarial examples. Whether an adjective or adverb is properly used heavily relies on the context, making it difficult for the GEC systems to correct them all. (iii) Not surprisingly, most of the prepositions errors are inherently hard to correct, even for native speakers.

\(^8\)We run the code and the pre-trained model offered by the authors at https://github.com/zhawe01/fairseq-gec.
5 Conclusion

In this paper, we proposed a data augmentation method for training a GEC model by continually adding to the training set the adversarial examples, particularly generated to compensate the weakness of the current model. To generate such adversarial examples, we first determine an important position to change and then modify it by introducing specific grammatical issues that maximize the GEC model’s prediction error. The samples generated by our adversarial attack algorithm are more meaningful and valuable than those produced by recently proposed methods, such as the direct noise and the back translation. Experimental results demonstrate that the GEC models trained with the data augmented by these adversarial examples can substantially improve both generalization and robustness.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by National Key R&D Program of China (No. 2018YFC0830900), National Science Foundation of China (No. 62076068), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and Zhangjiang Lab.

References

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Christopher Bryant, Mariano Felice, Østein E Andersen, and Ted Briscoe. 2019. The bea-2019 shared task on grammatical error correction. In Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 52–75.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. arXiv preprint arXiv:1312.3005.

Minhao Cheng, Wei Wei, and Cho-Jui Hsieh. 2019. 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, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2stick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. arXiv preprint arXiv:1803.01128.

Shamil Chollampatt and Hwee Tou Ng. 2018a. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In Thirty-Second AAAI Conference on Artificial Intelligence.

Shamil Chollampatt and Hwee Tou Ng. 2018b. Neural quality estimation of grammatical error correction. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2528–2539.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 568–572. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. arXiv preprint arXiv:1712.06751.

Pavlos S Efraimidis and Paul G Spirakis. 2006. Weighted random sampling with a reservoir. Information Processing Letters, 97(5):181–185.

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

Tao Ge, Furu Wei, and Ming Zhou. 2018. Fluency boost learning and inference for neural grammatical error correction. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1055–1065.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? arXiv preprint arXiv:1710.03348.

Roman Grundkiewicz and Marcin Junczyz-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. arXiv preprint arXiv:1804.05945.

Roman Grundkiewicz, Marcin Junczyz-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 252–263.
Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. 2019. On the robustness of self-attentive models. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1520–1529.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. arXiv:1707.02026.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. arXiv preprint arXiv:1707.07328.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? natural language attack on text classification and entailment. arXiv preprint arXiv:1907.11932.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. arXiv preprint arXiv:1804.05940.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. arXiv preprint arXiv:1909.00502.

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

Qi Lei, Lingfei Wu, Pin-Yu Chen, Alexandros G Dimakis, Inderjit S Dhillon, and Michael Witbrock. 2018. Discrete adversarial attacks and submodular optimization with applications to text classification. arXiv preprint arXiv:1812.00151.

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

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. arXiv preprint arXiv:1904.05780.

Jared Lichtarge, Christopher Alberti, Shankar Kumar, Noam Shazeer, and Niki Parmar. 2018. Weakly supervised grammatical error correction using iterative decoding. arXiv preprint arXiv:1811.01710.

Andres Marzal and Enrique Vidal. 1993. Computation of normalized edit distance and applications. IEEE transactions on pattern analysis and machine intelligence, 15(9):926–932.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction, pages 95–100. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. arXiv preprint arXiv:1702.04066.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwimoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pages 1–14.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. arXiv preprint arXiv:1904.01038.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. arXiv preprint arXiv:1707.00299.

Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. arXiv preprint arXiv:1707.02812.

Allen Schmaltz, Yoon Kim, Alexander M Rush, and Stuart M Shieber. 2017. Adapting sequence models for sentence correction. arXiv preprint arXiv:1707.09067.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, pages 198–202. Association for Computational Linguistics.
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 Advances in neural information processing systems, pages 5998–6008.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for nlp. arXiv preprint arXiv:1908.07125.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. arXiv preprint arXiv:1603.09727.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 619–628.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 380–386.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. arXiv preprint arXiv:1903.00138.

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

Xiaoqing Zheng, Jiechang 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.
## A Detailed Hyper-parameter Settings

| Transformer (large) | Values |
|---------------------|--------|
| Number of Encoder Layers $N_e$ | 6 |
| Number of Decoder Layers $N_d$ | 6 |
| Dimension of Embeddings $d_{emb}$ | 1024 |
| Dimension of Hiddens $d_h$ | 1024 |
| Dimension of Feed-forward Layers $d_{ff}$ | 4096 |
| Number of Multi-heads $h$ | 16 |
| Dropout $P_{drop}$ | 0.3 |
| Number of parameters | $2.13 \times 10^8$ |

Table 7: The architecture of the Transformer(big) (Vaswani et al., 2017) is used as our base model.

| Configurations | Values |
|----------------|--------|
| **Base Model Training** | |
| Number of Epochs | 20 |
| Loss Function | Label smoothed cross entropy (smoothing value: $\epsilon_{ls} = 0.1$) (Szegedy et al., 2016) |
| Optimizer | Adam (Kingma and Ba, 2014) ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$) |
| Learning Rate Schedule | Inverse square root of the update number, initial value = $5 \times 10^{-4}$ |
| Gradient Clipping | 1.0 |
| **Pre-training** | |
| Number of Epochs | 10 |
| Loss Function | Label smoothed cross entropy |
| Optimizer | Adam |
| Learning Rate Schedule | Inverse square root of the update number, initial value = $5 \times 10^{-4}$ |
| Gradient Clipping | 1.0 |
| **Fine-tuning** | |
| Number of Epochs | 15 |
| Loss Function | Label smoothed cross entropy |
| Optimizer | Adafactor (Shazeer and Stern, 2018) |
| Learning Rate Schedule | Fixed learning rate $3 \times 10^{-5}$ |
| Stopping Criterion | Use the model with best validation perplexity on BEA-valid |
| Gradient Clipping | 1.0 |
| Beam Search size | 5 |
| **Adversarial Training** | |
| Number of Epochs | 3 (each time) |
| Loss Function | Label smoothed cross entropy |
| Optimizer | Adafactor ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$) |
| Learning Rate Schedule | Fixed learning rate $3 \times 10^{-5}$ |
| Gradient Clipping | 1.0 |
| Beam Search size | 5 |

Table 8: Hyper-parameter settings for training. Kiyono et al. (2019) conducted an empirical study of pre-training with pseudo data and we follow many settings with them. We conducted our experiments on 4 TITAN Xp GPUs. Each epoch took 20 minutes during pre-training, and 10 minutes during fine-tuning and base model training. The time spent on adversarial training depends on the number of adversarial examples, about 20-30 minutes each epoch.